Studies in Beamforming Tracking Neural Networks Mathematics

R. L. Streit

Scientific and ngineering Studies

Compiled 1991

NAVAL UNDERWATER SYSTEMS CENTER

Studies in Beamforming Tracking Neural Networks Mathematics

R. L. Streit

Scientific and Engineering Studies

Compiled 1991

NAVAL UNDERWATER SYSTEMS CENTER

NEWPORT LABORATORY, NEWPORT, RHODE ISLAND
NEW LONDON LABORATORY, NEW LONDON, CONNECTICUT

Preface

The four parts of this collection of technical articles, reports and memoranda deal with beamforming studies (12 papers), frequency line detector/trackers (9 papers), artificial neural networks (3 papers), and mathematical studies (10 papers). The content of these 34 papers is discussed in the foreword provided for each part of this collection.

Dr. William I. Roderick
Associate Technical Director for Technology
NAVAL UNDERWATER SYSTEMS CENTER

COMPILED 1991

TABLE OF CONTENTS

BEAMFORMING STUDIES

FREQUENCY LINE DETECTOR/TRACKERS

ARTIFICIAL NEURAL NETWORK STUDIES

MATHEMATICS

BEAMFORMING STUDIES

Foreword

Sidelobe suppression in acoustic arrays is an important problem that gives rise to challenging mathematical problems that often cannot be solved analytically. Moreover, the mathematical problems encountered are sometimes new and, although interesting in themselves, not studied in the literature. The novelty and size of these problems make the development of numerical algorithms for their solution very difficult. Papers [1] - [3]* of this compilation make the point clearly for the case of linear arrays with missing elements. Supporting mathematical background is found in papers [26] - [28] of this compilation. Sidelobe optimality is stressed in these papers. A different definition of optimality is presented in paper [4]. The methods of these four papers are applicable to acoustic arrays of arbitrary geometry.

Interchannel crosstalk between acoustic channels can occur if array telemetry systems are imperfect. The potentially debilitating effects of crosstalk on beamforming performance are analyzed in paper [5]. This paper presents the first theoretical study of the effect of crosstalk on beamformer performance. It is shown that crosstalk can be corrected before the channels enter the beamformer, provided the crosstalk levels do not exceed an upper bound derived from the crosstalk transfer function.

Families of weights (shading coefficients) are often used for linear arrays for sidelobe reduction. A two-parameter family of weights is given in paper [6] for discrete and continuous linear arrays. Of particular interest in this paper is the discrete version of the Kaiser-Bessel window for continuous apertures. More general weight families are discussed in [7]. Taylor weights are unrelated to these families and are discussed in [8].

A method for optimal and suboptimal weight design suitable only for linear arrays is presented in papers [9] and [10]. Mathematical results related to this work are found in papers [31] and [32] of this compilation. Paper [11] shows that phased weights are unnecessary for optimal sidelobe suppression in steerable linear arrays. Paper [12] discusses the element location problem for unequally spaced linear arrays.

* *Papers are referred to in the order that they appear in the compilation.*

In Situ Optimal Reshading Of Arrays With Failed Elements

M. S. Sherrill and R. L. Streit

In Situ Optimal Reshading of Arrays with Failed Elements

MICHAEL S. SHERRILL AND ROY L. STREIT, SENIOR MEMBER, IEEE

(Invited Paper)

Abstract—**An algorithm is presented which computes optimal weights for arbitrary linear arrays. The application of this algorithm to *in situ* optimal reshading of arrays with failed elements is discussed. It is shown that optimal reshading can often regain the original sidelobe level by slightly increasing the mainlobe beamwidth. Three examples are presented to illustrate the algorithm's effectiveness. Hardware and software issues are discussed. Execution time for a 25-element array is typically between 1 and 2 min on an HP9836C microcomputer.**

I. Introduction

A linear array of discrete elements (sensors) often experiences element failures *in situ*. These failures can significantly increase the sidelobe levels of the array wavenumber response, depending on how many elements fail and where the elements are located within the array. We discuss here an optimal reshading (reweighting) algorithm which can be applied *in situ* to reduce the sidelobe levels to the original design level. In many common element-failure situations, optimal reshading can regain the original sidelobe level by slightly increasing the mainlobe beamwidth. In arrays which experience significant element failures, optimal reshading is still possible, but may be of limited use. Three examples given below demonstrate a few of the possibilities.

An algorithm for optimal reshading was first proposed in [1] by Streit and Nuttall. Their algorithm utilized the general-purpose subroutine [2] to solve a specially structured "linear programming" problem. Unfortunately, their algorithm required hours of computation time and large amounts of computer storage on a minicomputer (the VAX 11/780) to optimally reshade a 50-element array with five failed elements. Consequently, their algorithm is not useful for *in situ* optimal reshading.

The shading algorithm proposed here differs from Streit and Nuttall's primarily in that we solve their linear programming problem using a new general-purpose subroutine [3], [4], herein referred to as Algorithm 635. Algorithm 635 uses the special structure of the linear programming problem to reduce time and storage requirements by orders of magnitude. Algorithm 635 can be incorporated easily in Streit and Nuttall's original approach. A significant algorithmic improvement was discovered in the course of this study and is described below. The resulting shading algorithm is fast enough and small enough to execute successfully on microcomputers (such as the HP9836C used here) in only a few minutes. Typical execution time for a 25-element array is under 2 min; for a 50-element array, execution time is typically under 10 min. The current algorithm, and the HP9836C with its inherent transportability, comprise an effective system for optimal reshading *in situ*.

Manuscript received March 11, 1986; revised August 11, 1986.

The authors are with the Naval Underwater Systems Center, New London, CT 06320.

IEEE Log Number 8714258.

II. Optimal Array Shading

The wavenumber response of a linear array composed of N discrete omnidirectional elements located at arbitrary fixed positions x_n is given by

$$T(k) = \sum_{n=1}^{N} w_n \exp[-ikx_n] \tag{1}$$

where w_n are the element weights and the independent variable k denotes wavenumber in radians per unit length. The element weights are required to be real, but this entails no loss of generality (see below in Section III). Also, from (1), $T(-k) = T^*(k)$ for real weights (asterisk denotes conjugation), so it is unnecessary to consider negative values for k and we confine our attention to nonnegative k.

The array response as a function of k can be considered to be composed of a mainlobe beamwidth and a sidelobe region. The objective of the optimization process is to make $|T(k)|$ as small as possible on the user-specified sidelobe interval. Array weights which achieve this objective are said to be optimal. The optimization process usually produces equivalued sidelobes in the sidelobe region.

Weights that are optimal for a full array do not remain optimal after the array experiences element failures. To partially compensate for failed elements, the array is optimally reshaded by undertaking the optimization process again and incorporating knowledge of which elements have failed. As the examples below will show, the effectiveness of this strategy depends upon how many elements have failed and the location of these elements in the array.

The sidelobe interval is defined differently depending on the interelement spacing of the array. For an array with periodically spaced elements and no failures, the sidelobe interval is defined to be $[K_0, (2\pi/D) - K_0]$, where K_0 is calculated from the desired sidelobe level and the number N of array elements.[1] D is the physical distance from sensor to sensor.

[1] For an N-element array and $-t$-dB peak sidelobes, we have $K_0 = (2/D) \arccos(1/Z_0)$ where $2Z_0 = [r - (r^2 - 1)^{1/2}]^{1/M} + [r + (r^2 - 1)^{1/2}]^{1/M}$, $r = 10^{t/20}$, and $M = N - 1$. The interelement spacing D is assumed to be half of the so-called design wavenumber, and N is the number of array elements before failures.

Furthermore, the minimization interval can be reduced to $[K_0, \pi/D]$, since the response of this array is symmetric about $k = \pi/D$. K_0 is typically the point on the mainlobe response which is equal in magnitude to the peak of the sidelobes, but this is not always true for seriously degraded and/or aperiodic arrays (see Example 3 below). For arrays with aperiodically spaced elements, the sidelobe-interval, denoted by $[K_0, K_1]$, must be chosen by inspection of a nonoptimal beam pattern or some other means. $|T(k)|$ must be minimized over the full $[K_0, K_1]$ range since, in general, an aperiodic array response is not symmetric about any wavenumber other than $k = 0$. The ability to specify arbitrary K_0 and K_1 is particularly useful for those applications involving aperiodically spaced elements because lower sidelobe levels may be obtained by looking at different minimization regions.

The optimization process deals with element failures in an array in the following way.

Step 1. Maintain mainlobe beamwidth and permit the sidelobe levels to rise.

Step 2. Regain, if possible, the original sidelobe level by broadening the mainlobe.

Broadening the mainlobe by increasing K_0 (step 2) is performed only if the sidelobe level, even after optimal reshading, has risen to an unacceptable value because of element failures. Thus step 1 is normal algorithmic procedure, and step 2 requires some iteration in specifying K_0 and/or K_1 because a compromise has to be made between the mainlobe beamwidth and the level of the sidelobes.

The solution of the array problem in the original formulation [1] is mathematically equivalent to solving an overdetermined system of complex linear equations. Unacceptably high sidelobes result if this system is solved in the usual least squares sense, so it is necessary to solve the system so that the magnitude of the maximum residual error is minimized. There now exists [3] an efficient algorithm and corresponding FORTRAN code [4] for solving problems of this sort to high accuracy.

To obtain the beamformer equation in an appropriate format to utilize this algorithm, we normalize the peak response of $T(k)$ so that $T(0) = 1$. This gives

$$\sum_{n=1}^{N} w_n = 1. \tag{2}$$

We solve (2) for the Nth weight w_N and substitute in (1) to obtain

$$T(k) = \exp[-ikx_N] + \sum_{n=1}^{N-1} w_n[\exp(-ikx_n) - \exp(-ikx_N)]. \tag{3}$$

By sampling $T(k)$ at the M equispaced points

$$k_m = K_0 + \frac{[K_1 - K_0]}{M-1}(m-1), \qquad m = 1, \cdots, M \tag{4}$$

we can write the problem of minimizing the peak sidelobe level of the array response as

$$\min_{\{w_n\}} \max_{1 \le m \le M} \left| f_m - \sum_{n=1}^{N-1} a_{mn} w_n \right| \tag{5}$$

where the complex numbers f_m and a_{mn} are defined by

$$f_m = \exp[-ik_m x_N]$$
$$a_{mn} = \exp[-ik_m x_N] - \exp[-ik_m x_n]. \tag{6}$$

The problem (5) is precisely the form necessary for application of Algorithm 635. For theoretical details of this algorithm, the interested reader is referred to [3].

Sometimes a few of the optimum weights for arrays with failed elements are observed to be negative, particularly those on the end elements. If the weights are applied in hardware, providing a 180° phase factor on the element output may not be desirable or possible. However, Algorithm 635 allows the selection of all nonnegative weights; this is implemented by the addition of constraints to (5). Usually, but not always, an element is zeroed if it would have had a negative weight. From (2) it follows that, if all the element weight values are required to be positive, they must be between 0 and 1. The requirement that weights $w_1, \cdots, w_{N-1}$ be between 0 and 1 can be written mathematically as

$$\left| w_n - \frac{1}{2} \right| \le \frac{1}{2}, \qquad n = 1, \cdots, N-1. \tag{7}$$

Algorithm 635 requires these $N - 1$ constraints. Algorithm 635 can also incorporate any number of general constraints of the form

$$\left| \sum_n b_{mn} w_n - c_m \right| \le d_m, \qquad m = 1, 2, \cdots, L \tag{8}$$

where c_m and d_m are constants. The requirement that w_N also be nonnegative gives

$$\left| \left(1 - \sum_{n=1}^{N-1} w_n\right) - \frac{1}{2} \right| \le \frac{1}{2}$$

or

$$\left| \sum_{n=1}^{N-1} w_n - \frac{1}{2} \right| \le \frac{1}{2} \tag{9}$$

which is clearly a special case of the general constraints (8).

III. Algorithm Improvements

Several changes to the algorithm presented in [1] enable significant reduction in the need for computational intensity. Lewis and Streit [5] have proved that, for a general line array shaded so that it has optimal sidelobe levels when steered through the same number of degrees either side of broadside, there exists a set of optimal weights that are real. Thus complex weights do not need to be considered. This fact allows an approximate eight-fold reduction in computation time and a two-fold reduction in storage requirements.

It is clear that the 50-element example run in Streit and Nuttall [1] was significantly oversampled in wavenumber. Their beam pattern can be reproduced with a four-fold reduction in the sampling of $T(k)$ (see Example 2 below), and this in no way detracts from the practical application of the algorithm. A significant reduction in computation time is realized by decreasing the number M of beam pattern samples in (4).

A significant algorithmic modification made to Algorithm 635 further decreases computation time. We have labeled this modification "fast costing" and it is an important step in making the algorithm feasible on microcomputers such as the HP9836C. In order to describe this modification properly, some familiarity with the simplex method of linear programming and reference [3] is assumed.

Algorithm 635 can be broken into two fundamental computational operations called "costing" and "pivoting." "Costing" determines the so-called minimum reduced cost coefficient and requires $2NM$ multiplications, where N is the number of discrete array elements and M is the number of samples taken of the beam pattern. "Pivoting" is a basis update and requires N^2 real multiplications. It is clear that the speed of the algorithm is intimately related to the number M of samples taken of the beam pattern, as well as the number N of discrete array elements. Since M is larger than N, "costing" requires more multiplications than "pivoting."

"Costing" in the linear array application means that, in each simplex iteration, the "discretized absolute value" of every sidelobe sample of the wavenumber response function $T(k_m)$, $m = 1, \cdots, M$, is computed to determine the "minimum reduced cost coefficient" of the current "basic feasible solution." By proceeding through a finite sequence of such "basic feasible solutions," we arrive at the solution of the "discretized problem." As shown in [3], this implies that the computed optimal wavenumber response function can have sidelobe levels that are theoretically at most 0.04 dB higher than the true optimum sidelobe level.[2] "Fast costing" refers simply to the fact that we first determine which of the sidelobe samples $T(k_m)$, $m = 1, \cdots, M$, has the largest true absolute value, and then compute the "discretized absolute value" of this one complex number. Therefore, only one "discretized absolute value" calculation is performed in each simplex iteration instead of M such calculations. The resulting reduction in computational effort is significant in microcomputing environments. The drawback is that the use of "fast costing" prevents the simplex algorithm from converging to a solution of the "discretized problem." Fortunately, however, it can be proved that we must approximate the solution in a well-defined sense. In the linear array application, "fast costing" results in the computed optimum beam pattern having sidelobe levels that are theoretically at most 0.08 dB higher than the true optimum level.[3] This is a small price to pay for major execution time improvements.

[2] The theoretical error of at most 0.04 dB is derived by taking 20 $\log_{10}$ (sec (π/p)), where $p = 32$. The term sec (π/p) is the error bound discussed in [3].

[3] Fast costing squares the error bound, giving sec^2 (π/p), or 0.08 dB when $p = 32$.

IV. Algorithm Implementation for *In Situ* Use

An algorithm must be reliable, easy to use, and fast when executing on portable microcomputers, to be useful for *in situ* application. The following section details the most important hardware and software issues addressed to enable *in situ* optimal reshading of arrays with failed elements.

The algorithm has been coded in BASIC and is comprised of Algorithm 635 and an array processing driver program. Algorithm 635 solves the linear program for a set of optimal weights, given data supplied by the driver program. The driver performs the initial setup based on several user inputs and provides all program output.

The driver program may be used with linear arrays having either periodic or aperiodically spaced elements. Program output consists of a graph of the optimal beam pattern, a graph of the optimal normalized element weights, and several parameters pertinent to the specific problem. Provision is made for storing the weights in a separate data file for possible use with digital beamformers.

A Hewlett-Packard (HP) specific software modification was made by setting up the input data arrays (equation (6)) in buffers so that they are accessible for a one-dimensional multiply. For large-array dimensions, indexing a doubly subscripted data array and performing a dot product takes more time on the HP9836C than reading in a data array from a buffer, doing a MAT multiply, and performing a summation. (A MAT multiply is simply an element-by-element multiply of two equally dimensioned data arrays.) However, this procedure is more time consuming when the input data arrays are very small (i.e., the number of elements in the line array is small). The break-even point occurs at around 12 or 13 elements, so it was decided to incorporate this speed enhancement for the longer-running larger line arrays and trade off some speed reduction on the smaller line arrays.

To obtain fast execution times for *in situ* applications, we use one hardware speed enhancement, a 12.5-MHz fast CPU card with 16 kbytes of cache memory. This hardware supplement is available from HP for use on the HP9836C. Cache memory is fast memory resident on the CPU card for quick instruction acquisition. The use of the fast CPU board rather than the 8-MHz clock present in the standard computer configuration results in an approximate factor-of-two increase in observed speed.

The complete program is precompiled by use of software and a floating point math card available from the INFOTEK company. Precompilation reduces most computational portions of the BASIC code to machine language, giving an additional three-fold reduction in computation time. It is also desirable to upgrade the operating system for the HP to its latest revision. All work on these problems was run using the BASIC 3.0 operating system and the hardware supplements noted above.

Computation time is defined as time spent in Algorithm 635 and does not include the small amount of set-up time required by the driver program. Computation times are for the compiled BASIC program run on the HP9836C with the special hardware additions mentioned above.

The program described here needs just over 303 kbytes of

internal memory in addition to the memory required by the operating system to execute on the HP9836C. This is the amount of space required by fixing the maximum array size at $N = 50$, and allowing at most $M = 256$ beam pattern samples. Users can change dimensions to suit their specific needs, but storage requirements presently are directly proportional to the product NM. Even for a much larger number of line array elements, it is unlikely that memory restrictions would prove to be a problem on the HP9836C since extra memory boards of 1 Mbyte each are readily available.

Ongoing modifications should further enhance the capability and speed of the BASIC algorithm and driver. The addition of the ability to handle directional sensors is both useful and straightforward to implement. Execution of identical code on the new HP 300 series computers, which have a 16.6-MHz clock rate, should further reduce the computation time. Computation times on the order of 5 min for a 50-element array and 1 min for a 25-element array are anticipated.

It is possible to run the BASIC program in its uncompiled state. The execution of the program with cache memory and the fast CPU board as the only enhancements results in computation times of approximately 25 min for a 50-element array and 4.5 min for a 25-element array.

A copy of the entire program is available from the authors. Our specific implementation in HP BASIC utilizes several hardware and software devices to achieve computational efficiency, some of which may not be pertinent to other BASIC operating systems running on comparable machines. Users will undoubtedly find it necessary to make modifications to the code to allow it to run on other HP equipment or in BASIC on the VAX.

V. Examples

The following examples demonstrate the utility of the current algorithm for application *in situ* and provide insight into different situations that might arise when reshading equispaced arrays with failed elements. If optimal reshading can restore the array's original design sidelobe level by slightly increasing the mainlobe beamwidth, then we say that the optimal reshading has been effective. Optimal reshading is effective in many common element-failure situations. When the array is severely degraded, optimal reshading is less effective but is still useful in reducing the negative impact of element failures. These examples demonstrate that the effectiveness of reshading depends upon the number of element failures, as well as the location of the failed elements within the array.

Missing elements are modeled by zeroing the appropriate weights. In these examples, N refers to the number of intact array elements, M is the number of beam pattern samples, and K_0 is calculated by using the equation in an earlier footnote. We define the mainlobe width to be twice K_0 in all three examples.

A. Example 1: Effective Reshading

This example demonstrates that reshading can restore the original sidelobe level of an array response by slightly increasing the mainlobe beamwidth. In a 25-element equispaced array, originally designed for -30-dB sidelobes, elements 2 and 4 have failed. Therefore, $N = 23$, $M = 128$, and $K_0 = 0.6877$. We first keep the mainlobe width fixed and allow the sidelobe level to rise. See Fig. 1. The peak sidelobe level has risen to -26.86 dB below the mainlobe, and the mainlobe width is unchanged. If the sidelobe level after reshading is too high, an alternative to discarding or repairing the array is to broaden the mainlobe beamwidth. In Fig. 2, K_0 is increased to 0.775 and the peak sidelobe level diminishes to -30.04 dB below the mainlobe. A trade-off must always be made between an enlarged mainlobe beamwidth and an acceptable peak sidelobe level. In this case the mainlobe was increased 12.7 percent in order to recover the original sidelobe level. Execution times on the HP9836C are between 1 and 2 min for Figs. 1 and 2.

B. Example 2: Moderately Effective Reshading

This example is taken from Streit and Nuttall [1]. Because of the improvements detailed in Section III, above, the current algorithm runs faster on the HP9836C than on the VAX 11/780, although the floating point multiply time on the HP in its basic configuration is roughly 200 times slower than on the VAX.

Consider a linear array with 50 equispaced elements, initially designed for peak sidelobes of -30 dB relative to the mainlobe. Fig. 3 shows the classical Dolph–Chebyshev beam pattern with -30-dB sidelobes throughout the minimization range $[K_0, (2\pi/D) - K_0]$. This was computed using the current algorithm in 6.11 min. (This ideal case could have been computed analytically.)

Now we suppose that five elements, 7, 22, 40, 43, 50, of the array have failed. The optimal response after reshading the array is shown in Fig. 4. The peak sidelobe level has risen to -25.51 dB, but we have maintained mainlobe beamwidth and retained full steering capability. In this example $N = 45$ and $M = 128$.

This example (Fig. 4) took 7.47 minutes on the HP9836C and required 292 simplex iterations. The algorithm of Streit and Nuttall required 38.4 min and 402 iterations on the VAX.

Recovery of the original sidelobe level is possible (Fig. 5). The mainlobe beamwidth must be increased by the large factor 257.6 percent ($K_0 = 0.871$) and the execution of this task takes 8.98 min and requires 351 iterations. The constraint that all the weights lie between 0 and 1 is used. It is necessary to use the constraint in this instance because otherwise a dislocation of the maximum response from $k = 0$ results. This dislocation is due to the presence of too many negatively weighted elements.

C. Example 3: A Severely Degraded Array

This example shows that, for severely degraded arrays, recovery of the original sidelobe level may not be possible by increasing the mainlobe beamwidth, even after optimal reshading. Consequently, control of the level of the first sidelobe must be relinquished in order to gain control of the level of the remaining sidelobes.

Consider a 25-element array with elements 11 and 14 failed.

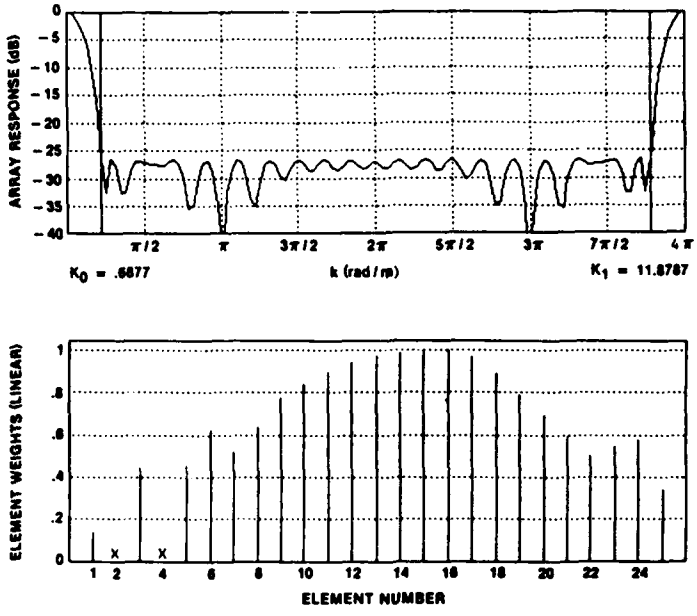


Fig. 1. Optimized array response and normalized weights for 25 elements with elements 2 and 4 missing.

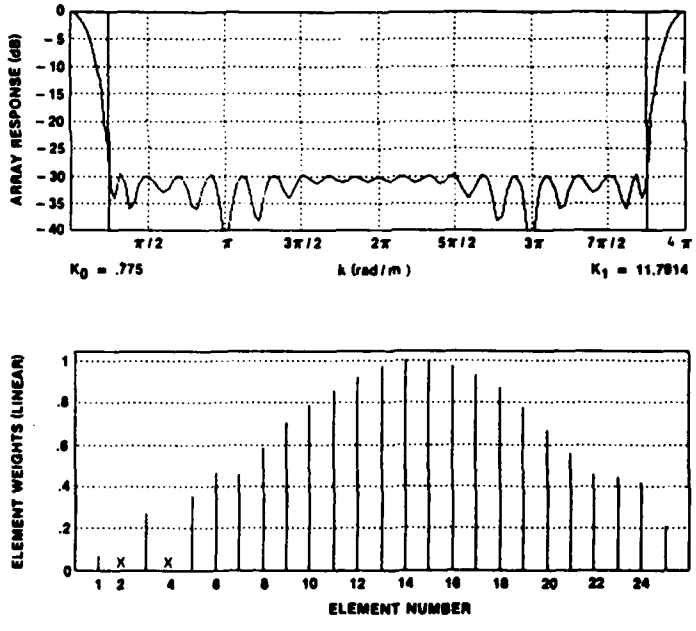


Fig. 2. Array response and normalized weights for Example 1 with $K_0 = 0.775$.

The original sidelobe level is -30 dB. Here $N = 23$, $M = 128$, and $K_0 = 0.6877$. Fig. 6 shows the algorithm's optimal response to this configuration. It is a significant observation that, in this case, small perturbations of K_0 will not affect the level of the sidelobes. Only when the first sidelobe is incorporated into the mainlobe beamwidth ($K_0 = 1.27$) does the level of the remaining sidelobes return to the original desired value (see Fig. 7). It is apparent that decreasing the minimization interval by moving K_0 far enough to the right will improve the approximation, but one must give up control of the first sidelobe to reduce the others to acceptable levels. The net effect of losing two elements so close to the center is that negligible emphasis is placed on the remaining center elements (12 and 13) and the rest of the aperture is reshaded as if it were two separate arrays.

This situation cannot be overcome by using different

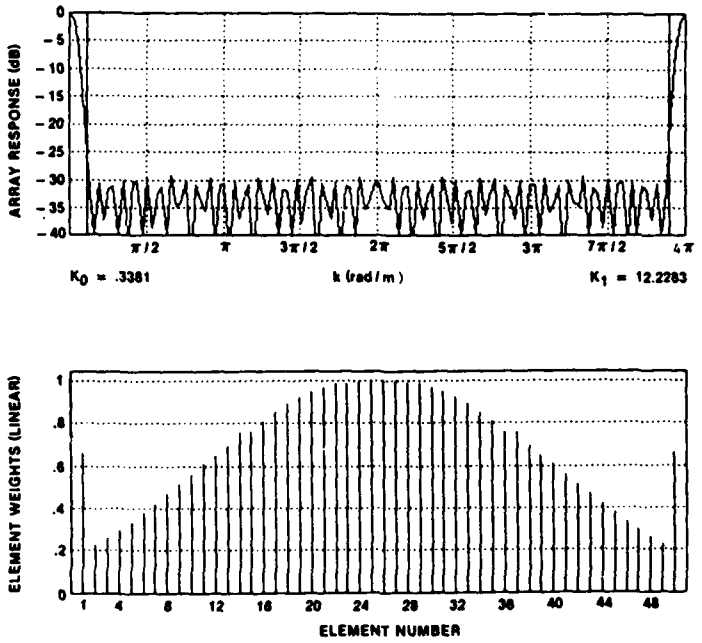


Fig. 3. Classical Dolph-Chebyshev array response and normalized weights for $N = 50$ and -30-dB sidelobes.

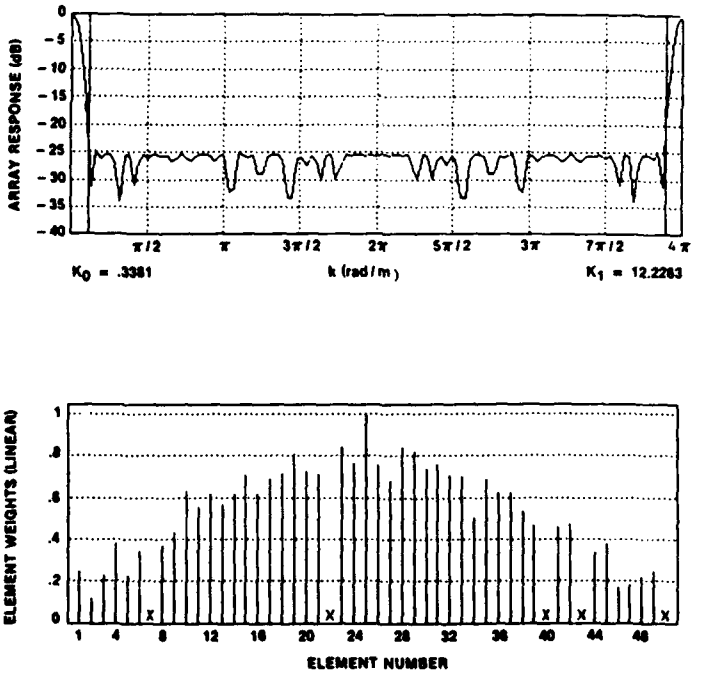


Fig. 4. Optimized array response and normalized weights for 50 elements with elements 7, 22, 40, 43, and 50 failed.

weights. The optimal property of the array problem formulation and solution tells us that no weights exist which can suppress all the sidelobes below a certain level. Thus this array has lost too many elements and performance cannot be restored to its original design levels merely by reshading.

We have chosen to relinquish control of the first sidelobe to gain control of the level of the remaining sidelobes. We pick the first sidelobe merely for ease of implementation; modification of the algorithm to forfeit control of a different sidelobe could also have been done. The need to relinquish control of the first sidelobe level has only appeared in cases of severe array degradation due to element losses.

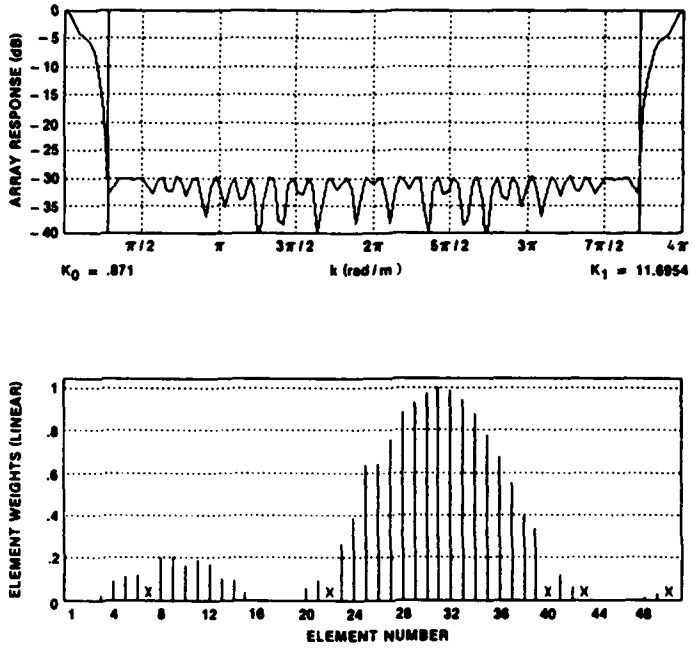


Fig. 5. Recovery of original sidelobe level. Example 2 with $K_0 = 0.871$.

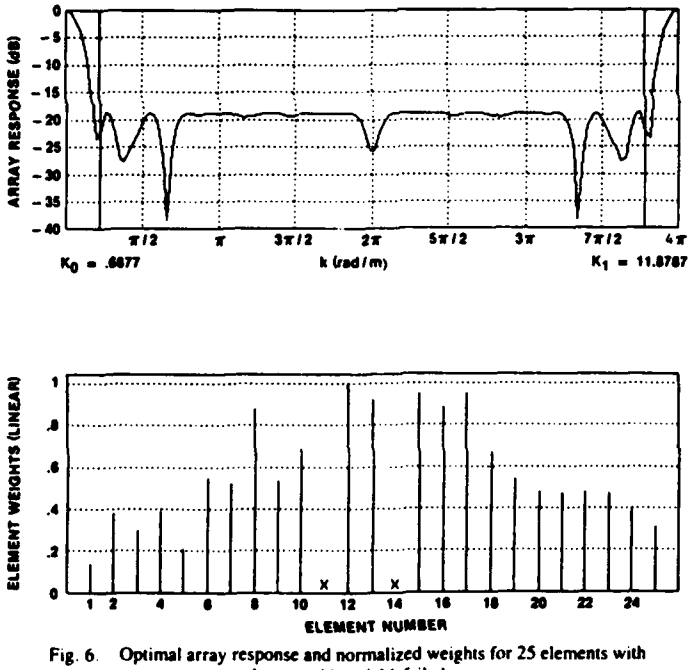


Fig. 6. Optimal array response and normalized weights for 25 elements with elements 11 and 14 failed.

VI. Conclusions

Arrays that have failed elements can be reshaded to obtain optimal array response functions. Optimal reshading is effective in many common element-failure situations. When the array is severely degraded, reshading is less effective, but still can be used to reduce the negative impact of element failures.

Optimal reshading can be accomplished *in situ*, quickly and reliably, on portable microcomputers using the algorithm described here. Arrays with 25 elements routinely run in less than 2 min and computation time for a 50-element array is less than 10 min. The algorithm can be applied to arrays of evenly or unevenly spaced linear geometry.

The above examples (and others) support the generally

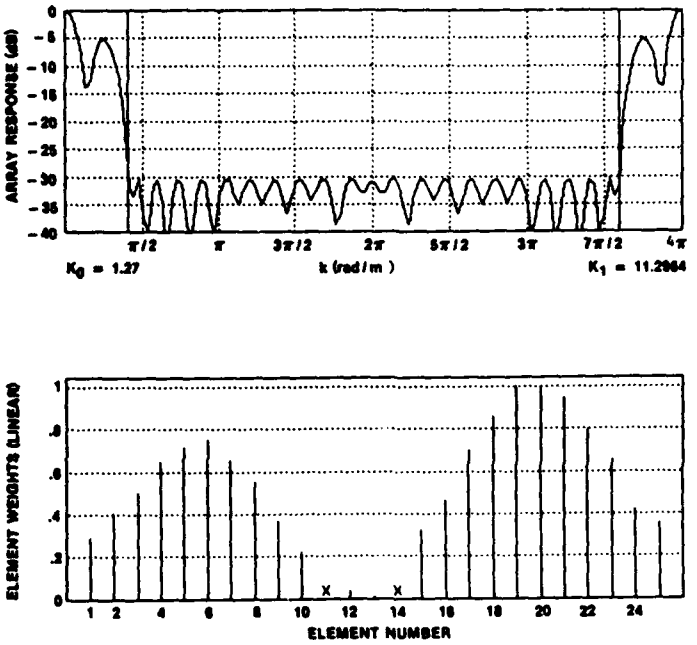


Fig. 7. Recovery of original sidelobe level, Example 3 with $K_0 = 1.27$.

accepted notion that failure of near-center elements is more detrimental to the array response than failure of near-edge elements.

Another application of Algorithm 635 is to arrays of planar and arbitrary three-dimensional geometry. Computation times for these more general arrays probably will depend upon N (number of sensors) and M (number of beam pattern samples) in the same manner as for linear arrays.

References

[1] R. L. Streit and A. H. Nuttall, "A general Chebyshev complex function approximation procedure and an application to beamforming," *J. Acoust. Soc. Amer.*, vol. 72, pp. 181-189, 1982.

[2] I. Barrodale and C. Philips, "Solution of overdetermined systems of linear equations in the Chebyshev norm," *Algorithm 495, ACM Trans. Math. Software*, vol. 1, pp. 264-270, 1975.

[3] R. L. Streit, "Solution of systems of complex linear equations in the L_∞ norm with constraints on the unknowns," *Soc. Indus. Appl. Math.*, vol. 7, no. 1, pp. 132-149, 1986.

[4] R. L. Streit, "An algorithm for the solution of systems of complex linear equations in the L_∞ norm with constraints on the unknowns," *Algorithm 635, ACM Trans. Math. Software*, vol. 11, no. 3, pp. 242-249, 1985.

[5] J. T. Lewis and R. L. Streit, "Real excitation coefficients suffice for sidelobe control in a linear array," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp. 1262-1263, 1982; also Naval Underwater Systems Center, New London, CT, Tech. Memo. 811114, Aug. 17, 1981.

Michael S. Sherrill was born on March 8, 1961, in Dover, DE. He received the B.S. degree in electrical engineering from the University of Delaware, Newark, in 1983.

He joined the staff of the Naval Underwater Systems Center, New London, CT, upon graduation. He is interested in mathematical modeling of physical systems and structured programming. Currently, he is responsible for a data acquisition and processing system for towed arrays.

★

Roy L. Streit (SM'84) was born in Guthrie, OK, on October 14, 1947. He received the B.A. degree (with honors) in mathematics and physics from East Texas State University, Commerce, in 1968, the M.A. degree in mathematics from the University of Missouri, Columbia, in 1970, and the Ph.D. degree in mathematics from the University of Rhode Island, Kingston, in 1978.

He is currently an Adjunct Associate Professor of the Department of Mathematics, University of Rhode Island. He was a Visiting Scholar in the Department of Operations Research, Stanford University, Stanford, CA, during 1981-1982. He joined the staff of the Naval Underwater Systems Center, New London, CT (then the Underwater Sound Laboratory), in 1970. He is an Applied Mathematician and has published work in several areas, including antenna design, complex function approximation theory and methods, and semi-infinite programming. He is currently conducting research in towed array design and hidden Markov models.

In Situ Optimal Reshading Of Arrays With Failed Elements: Algorithm Documentation Package

M. S. Sherrill and R. L. Streit

Abstract

This algorithm documentation package contains three appendices: Appendix A* is a reprint of an article in the IEEE Journal of Oceanic Engineering, volume OE-12, number 1, January 1987, *In Situ Optimal Reshading of Arrays with Failed Elements*; Appendix B is a listing for the driver routine in program "Reshade," and Appendix C,** a 3-1/2-inch floppy disk containing the program "Reshade" that runs on any HP Series 200/300 microcomputer.

* Appendix A is the lead document of this compilation.
** Appendix C is not included here.

IN SITU OPTIMAL RESHADING OF ARRAYS WITH FAILED ELEMENTS

ALGORITHM DOCUMENTATION PACKAGE

This document assembles under one cover information on a NUSC-developed algorithm which computes optimal shading weights for discrete elements (sensors) in linear acoustic arrays. The algorithm has been found especially useful when elements fail and array reshading is required *in situ*. The main attractions of the algorithm are that it loads easily on Hewlett-Packard microcomputers, and that it runs fast enough and is accurate enough to suit most sea trial and engineering development applications. Continuing requests for this information since an invited paper first appeared in the IEEE Journal of Oceanic Engineering in January 1987 motivated the publication of this documentation package.

The information included here is in hard copy and floppy disk form: the IEEE paper, *In Situ Optimal Reshading of Arrays with Failed Elements*, is reprinted in Appendix A; a program listing of the application-specific driver routine is given in Appendix B; and a 3½ inch floppy disk, containing the program "Reshade" which runs on any Hewlett-Packard Series 200 or 300 microcomputer, is pocketed in Appendix C.

GENERAL APPLICATION

When a linear array of discrete acoustic elements is subjected to the rigors of the ocean environment, individual elements can fail. Element failures are usually characterized by noisy channels, or intermittent responses, or no response at all. Depending on the number of failed elements and their specific locations within the array, sidelobe levels of the array wavenumber (k) response can rise significantly to degrade array performance. Because element weighting values determine array wavenumber response, weights that are optimal for a fully populated array have to be recalculated when elements fail. The optimal reshading (reweighting) algorithm described here can be applied *in situ* to compute weighting values that can reduce sidelobe levels to approximately the original design specification. In fact, in the more common situations where "a few" elements fail, optimal reshading does regain original sidelobe levels. Where large numbers of elements fail, optimal reshading is still possible but may be of limited use.

The original approach for optimal reshading of a linear array was proposed by Streit and Nuttall in 1982 (see reference 1, Appendix A). At that time, the algorithm was run on a VAX 11/780 and required hours of computation time and large amounts of mass storage for rudimentary element failure problems.

The 1987 reshading algorithm incorporates several algorithmic improvements that exploit the special structure of the underlying linear programming problem to reduce time and storage requirements by orders of magnitude. The current algorithm is still based on the original theory, but is now fast enough and small enough to execute successfully in minutes instead of hours in the application environment. Execution time for a 50-element array is typically about 10 minutes. Derivation of the optimal reshading algorithm and its implementation are given in the references of the paper reprinted in Appendix A; examples of array reshading are given in the paper itself.

TO RUN "RESHADE"...

- Insert the program disk from Appendix B into device/drive.
- Type the command string **LOAD "RESHADE:*(device specifier)*"**
- Press Enter.
- When the program is loaded, press Run.
- Follow the prompts.

PROGRAM NOTES

"Reshade" comprises a driver routine in HP BASIC which sets up the necessary variables to be optimized and a generic optimization routine. The driver is listed in lines 1 through 481 of the program—the printout is contained in Appendix B.

The driver included in "Reshade" applies to a linear array of acoustic elements, some of which may have failed during the course of a sea trial or similar event. Even with the array intact, "Reshade" allows the user to minimize the sidelobe levels of the array beamformer output, given a certain mainlobe width. If the minimum sidelobe levels remain too high, it is possible to alter the mainlobe beamwidth to reduce sidelobe levels. Note that the weights on each element can be set up as non-negative, if desired.

The program prompts require user inputs, not all of which are self-explanatory. For each user input, values in (parentheses) are those allowed, and values in [brackets] are the defaults. The maximum allowable total number of array elements is 50; the minimum is three. The computation time for a 50-element array is approximately 10 minutes, while a 10-element array runs in less than one minute.

The algorithm is applicable to both equispaced or aperiodically spaced linear arrays. In the equispaced arrangement, the wavenumbers *k0* and *k1*—which delimit the region in which the minimization is performed—are calculated automatically from the desired sidelobe level. The final sidelobe level depends upon the number of failed elements in the array and their location. In this case, only the inter-element spacing must be specified

In the aperiodically spaced arrangement, every element's position referenced to the forward end of the array must be specified. If elements have failed (or are missing), they are treated as if they do not exist. The wavenumbers *k0* and *k1* are not calculated automatically for an aperiodic array, and must be entered manually in units of radians/meter.

If unsatisfactory sidelobe levels are still present after running "Reshade" for an equispaced arrangement, *k0* can be increased to provide a larger beamwidth, thus reducing sidelobe levels. For an aperiodic array, *k0* and *k1* can be altered manually to reduce the sidelobe level in the region of interest.

Resultant weights can be stored in a data file in the following format:

- Equispaced element arrangement—total number of elements, followed by the inter-element spacing, followed by array weights.
- Aperiodic element arrangement—total number of elements, followed by each element's position, followed by array weights.

PROGRAM EXTENSIONS AND IMPROVEMENTS

"Reshade" and its associated algorithm have established the validity of *in situ* computation of linear acoustic array optimal shading weights. Virtually no sea trial is conducted today without reshading to compensate for failed elements. Extensions to larger linear array problems are potentially useful. Improvements and modifications to the original source code are possible in the light of recent advances in signal processing hardware, and are needed to obtain reasonable computation times for these larger arrays. With the advent of single-board array processors, the beam pattern computations done (implicitly) in each iteration in the generic optimization model (KAPROX) may be performed more quickly and accurately using a floating point FFT. This is but one example of software modifications which will enhance the performance of "Reshade."

The generic nature of the optimization routine lends itself to the solution of more general array problems. These arrays may be multiline, planar, or three-dimensional with arbitrary geometry. Each geometry, however, will require a specific driver routine to set up the problem to be optimized. In general, the drivers would need the capability to address complex weights, allocate enough memory for computations, and to take into account any application-specific constraints imposed on the optimization problem. Additional constraints can be useful; for instance, constraints can sometimes be used in active arrays to control adverse effects of acoustic coupling between the array elements.

APPENDIX B

DRIVER PROGRAM LISTING: "Reshade"
Lines 1 through 481

```
1      OPTION BASE 1
2      OUTPUT 2 USING "#,B";255,75          ! CLEAR SCREEN
3      PRINTER IS CRT
4      RAD
5      REAL Estore(50),U(256),Tbeam(256),Grres(256),Hsens(256)
6      INTEGER Ioexit(10),Itlog(10),Icount(50),Ijswt(3,51)
7      INTEGER Ldim,I,J,K,N,C5,Symflag,Cachflag,Floatflag,H
8      COM /Arrss/ Zradii(50),Bradii(4),Cheb(10),Z(50),Zcentr(50)
9      COM /Arrss1/ Ref(256),Imf(256),Reb(4,50) BUFFER,Imb(4,50) BUFFER,Rebcentr(
4),Imbcentr(4)
10     COM /Proj/ Basinv(51,54),Cossin(2,1025),Rea(256,50) BUFFER,Ima(256,50) BUF
FER,Cos45,Space
11     COM /Param/ INTEGER Ndim,M,L,Logp,Ndimp1,Ndimp4,S11,Tme,Misel(10)
12     COM /Buffmult/ Colrea(50),Colima(50),Colreb(50),Colimb(50)
13     COM /Groups/ INTEGER Nogroup,REAL Senslen,Xgroup(25),Dgroup,D(50)
14     COM /Groups1/ Hydsen$[3],Hydro$[3]
15     DIM S11$[3],Equi$[3],Weight$[3],Negwet$[3],Newko$[3],Data_msus$[10],Filena
me$[10],Wgtstore$[3],Group_space$[3]
16   !
17     Data_msus$=":INTERNAL"
18     Cachflag=0
19     ON ERROR GOTO 24          ! POSSIBLE ERRORS IF INTERFACE NOT PRESENT
20     CONTROL 32,1;1            ! IF CACHE MEMORY IS PRESENT IT WILL BE UTILIZED
21     OFF ERROR
22     STATUS 32,1;Stats
23     IF Stats THEN Cachflag=1
24     Floatflag=0
25     ON ERROR GOTO Redo
26     CONTROL 32,2;1       ! IF FLOATING POINT CARD PRESENT IT WILL BE UTILIZED
27     OFF ERROR
28     STATUS 32,2;Stats
29     IF Stats THEN Floatflag=1
30   !
31 Redo: !                                OBTAIN INPUT DATA
32     LOOP
33       IF Ndim=0 THEN
34         Ndim=16
35       ELSE
36         Ndim=Ndim+Tme
37       END IF
38       REPEAT
39         PRINT "ENTER TOTAL NUMBER OF ELEMENTS/GROUPS IN ARRAY: (3-50) ["&VAL$(
Ndim)&"]"
40         INPUT Ndim
41       UNTIL Ndim>2 AND Ndim<51
42   !
43       OUTPUT 2 USING "#,B";255,75        ! CLEAR SCREEN
44       REPEAT
45         PRINT "ENTER NUMBER OF SENSORS IN EACH GROUP: ["&VAL$(Nogroup)&"]"
46         INPUT Nogroup
47       UNTIL Nogroup<26
48       IF Nogroup=0 THEN Nogroup=1
49   !
50       IF Nogroup<>1 THEN
51         REDIM Xgroup(Nogroup)
52         OUTPUT 2 USING "#,B";255,75      ! CLEAR SCREEN
53         REPEAT
54           Group_space$=""
55           INPUT "IS ELEMENT SPACING WITHIN THE GROUP CONSTANT? (Y/N) [Y]",Grou
p_space$
56           IF LEN(Group_space$)=0 THEN
57             Group_space$="Y"
58           ELSE
59             Group_space$=UPC$(Group_space$[1])
60           END IF
61         UNTIL Group_space$="Y" OR Group_space$="N"
```

```
!
    IF Group_space$="N" THEN
      REPEAT
        H=0
        PRINT "ENTER POSITIONS OF SENSORS IN GROUP:"
        FOR I=1 TO Nogroup
          PRINT "SENSOR #"&VAL$(I)&":"
          INPUT Xgroup(I)
          IF I>1 AND Xgroup(I)<Xgroup(I-1) THEN H=H+1
        NEXT I
      UNTIL H=0
    ELSE
      REPEAT
        PRINT "ENTER SPACING BETWEEN SENSORS IN GROUP: ["&VAL$(Dgroup)&"]"
        INPUT Dgroup
      UNTIL Dgroup>0
      FOR I=1 TO Nogroup
        Xgroup(I)=(I-1)*Dgroup
      NEXT I
    END IF
  ELSE
    MAT Grres= (1.)
  END IF
!
  OUTPUT 2 USING "#,B";255,75        ! CLEAR SCREEN
  REPEAT
    Hydsen$=""
    PRINT "DO YOU WISH TO INCORPORATE A HYDROPHONE SENSITIVITY? (Y/N) [N]"
    INPUT Hydsen$
    IF LEN(Hydsen$)=0 THEN
      Hydsen$="N"
    ELSE
      Hydsen$=UPC$(Hydsen$[1])
    END IF
  UNTIL Hydsen$="Y" OR Hydsen$="N"
!
  IF Hydsen$="N" THEN
    MAT Hsens= (1.)
  ELSE
    OUTPUT 2 USING "#,B";255,75      ! CLEAR SCREEN
    REPEAT
      PRINT "ENTER THE PHYSICAL SENSOR LENGTH: (METERS) ["&VAL$(Senslen)&"
]"
      INPUT Senslen
    UNTIL Senslen>0.
!
    REPEAT
      Hydro$=""
      PRINT "IS HYDROPHONE TO BE MODELED AS A DIPOLE OR CONTINUOUS SENSOR?
 (D/C) [C]"
      INPUT Hydro$
      IF LEN(Hydro$)=0 THEN
        Hydro$="C"
      ELSE
        Hydro$=UPC$(Hydro$[1])
      END IF
    UNTIL Hydro$="C" OR Hydro$="D"
  END IF
!
  OUTPUT 2 USING "#,B";255,75        ! CLEAR SCREEN
  REPEAT
    PRINT "ENTER TOTAL NUMBER OF MISSING ELEMENTS/GROUPS [ "&VAL$(Tme)&" ]
:"
    INPUT Tme
  UNTIL Tme>=0 AND Ndim-Tme>2
!
```

B-4

```
125     IF Tme>0 THEN
126       REDIM Misel(Tme)
127       REPEAT
128         OUTPUT 2 USING "#,B";255,75    ! CLEAR SCREEN
129         PRINT "ENTER MISSING ELEMENT/GROUP NUMBERS (SEPARATED BY COMMAS) [ "
;Misel(*);" ]:"
130         INPUT Misel(*)
131         MAT SORT Misel(*)
132         H=0
133         FOR I=1 TO Tme
134           IF Misel(I)<1 OR Misel(I)>Ndim THEN H=H+1
135           IF I>1 THEN
136             IF Misel(I)=Misel(I-1) THEN H=H+1
137           END IF
138         NEXT I
139       UNTIL H=0
140     END IF
141   !
142     OUTPUT 2 USING "#,B";255,75                ! CLEAR SCREEN
143     REPEAT
144       INPUT "ARE ALL ELEMENTS/GROUPS EQUISPACED? (Y/N) [Y]",Equi$
145       IF LEN(Equi$)=0 THEN
146         Equi$="Y"
147       ELSE
148         Equi$=UPC$(Equi$[1])
149       END IF
150     UNTIL Equi$="Y" OR Equi$="N"
151   !
152     REDIM D(Ndim-Tme)
153 New_ko: !
154     Symflag=0                          ! FLAG FOR ARRAY SYMMETRY
155     IF Equi$="Y" THEN                  ! EQUISPACED ARRAY
156       OUTPUT 2 USING "#,B";255,75      ! CLEAR SCREEN
157       IF S11=0 THEN S11=30
158       REPEAT
159         PRINT "ENTER ORIGINAL SIDELOBE LEVEL (DB): (0 TO 50)) [-"&VAL$(S11)&
"]"
160         INPUT S11$
161         IF LEN(S11$)<>0 THEN S11=ABS(VAL(S11$))
162       UNTIL S11>-1 AND S11<51
163   !
164       OUTPUT 2 USING "#,B";255,75      ! CLEAR SCREEN
165       REPEAT
166         PRINT "ENTER ELEMENT/GROUP SPACING (METERS) (0-15) [ "&VAL$(Space)&"
 ]"
167         INPUT Space
168       UNTIL Space>0 AND Space<=15
169   !
170       N=Ndim-1
171       R=10^(S11/20)                ! CALCULATE K0
172       R2=R*R
173       R3=SQR(R2-1.)
174       R5=(R+R3)^(1./N)
175       R6=(R-R3)^(1./N)
176       Zo=(R5+R6)/2.
177       Ko=(2./Space)*ACS(1./Zo)
178       K1=2.*PI/Space-Ko
179   !
180       IF Newko$="Y" THEN
181         OUTPUT 2 USING "#,B";255,75    ! CLEAR SCREEN
182         REPEAT
183           PRINT "ENTER K0: [ "&VAL$(Ko)&" ]"
184           PRINT "SUGGESTED VALUE IS :";PROUND(Ko,-4)
185           INPUT Ko
186           K1=2.*PI/Space-Ko
187           IF Hydsen$="Y" OR Nogroup>1 THEN
```

B-5

```
188             PRINT "ENTER K1: [ "&VAL$(K1)&" ]"
189             INPUT K1
190           END IF
191         UNTIL Ko>0 AND Ko<PI/Space AND Ko<K1
192         Ndim=Ndim+Tme
193       END IF
194 !
195       C5=0
196       FOR I=1 TO Ndim
197         IF Tme>0 THEN
198           FOR J=1 TO Tme
199             IF I=Misel(J) THEN 204
200           NEXT J
201         END IF
202         C5=C5+1
203         D(C5)=Space*(I-1)
204       NEXT I
205       CALL Symd(Ndim-Tme,Symflag,D(*))
206     ELSE
207       PRINT "ENTER ELEMENT/GROUP POSITIONS (METERS FROM END) :"
208       PRINT "SKIP MISSING ELEMENT/GROUP POSITIONS."
209       IF Newko$="Y" THEN Ndim=Ndim+Tme
210       FOR I=1 TO Ndim-Tme
211         REPEAT
212           H=0
213           PRINT "ELEMENT/GROUP "&VAL$(I)&" [ "&VAL$(D(I))&" ] :";
214           INPUT D(I)
215           IF I>1 THEN
216             IF D(I)<D(I-1) THEN H=H+1
217           END IF
218         UNTIL H=0
219         PRINT D(I)
220       NEXT I
221       OUTPUT 2 USING "#,B";255,75      ! CLEAR SCREEN
222       REPEAT
223         INPUT "ENTER KO: (RAD/METER)",Ko
224         INPUT "ENTER K1: (RAD/METER)",K1
225       UNTIL K1>Ko AND Ko>=0
226       CALL Symd(Ndim-Tme,Symflag,D(*))
227     END IF
228 !
229     Ndim=Ndim-Tme
230     Ndimp1=Ndim+1
231     Ndimp4=Ndim+4
232     IF Equi$="Y" THEN
233       M=64
234       C3=(K1-Ko)/(2.*M-1.)
235     ELSE
236       M=128
237       C3=(K1-Ko)/(M-1.)
238     END IF
239     Logp=5
240     P1=(2^Logp)+1
241 !
242     OUTPUT 2 USING "#,B";255,75        ! CLEAR SCREEN
243     REPEAT
244       Negwet$=""
245       INPUT "WILL YOU ALLOW NEGATIVE WEIGHTS ? (Y/N) [Y]",Negwet$
246       IF LEN(Negwet$)=0 THEN
247         Negwet$="Y"
248       ELSE
249         Negwet$=UPC$(Negwet$[1])
250       END IF
251     UNTIL Negwet$="Y" OR Negwet$="N"
252     IF Negwet$="Y" THEN          ! NO CONSTRAINTS: (Wj)<=1
253       L=0
```

B-6

```
254     ELSE                    ! 1 CONSTRAINT: (SUM(Wj)-.5)<=.5,(Wj-.5)<=.5
255       L=1                   ! CHANGE L AND REDIM APPROPRIATE ARRAYS
256     END IF                  ! FOR MORE CONSTRAINTS
257     Ldim=MAX(1,L)
258  !
259  ! REDIMENSION INPUT ARRAYS
260  !
261     REDIM Ioexit(Logp),Ijswt(3,Ndimp1),Itlog(Logp),Icount(Ndim),Zradii(Ndim)
262     REDIM Bradii(Ldim),Cheb(Logp),Estore(Ndim),Z(Ndim),Zcentr(Ndim)
263     REDIM Basinv(Ndimp1,Ndimp4),Cossin(2,P1),Rea(M,Ndim),Ima(M,Ndim)
264     REDIM Ref(M),Imf(M),Reb(Ldim,Ndim),Imb(Ldim,Ndim),Rebcentr(Ldim)
265     REDIM Imbcentr(Ldim),D(Ndim),U(M),Tbeam(2*M),Grres(M),Hsens(M)
266     REDIM Colrea(Ndim),Colima(Ndim),Colreb(Ndim),Colimb(Ndim)
267  !
268     MAT Basinv= (0.)          ! INITIALIZE COMMONS
269     MAT Cossin= (0.)
270     MAT Z= (0.)
271     MAT Cheb= (0.)
272     MAT Colrea= (0.)
273     MAT Colima= (0.)
274     MAT Colreb= (0.)
275     MAT Colimb= (0.)
276  !
277     IF Negwet$="Y" THEN
278       MAT Reb= (0.)
279       MAT Imb= (0.)
280       MAT Rebcentr= (0.)
281       MAT Imbcentr= (0.)
282       MAT Bradii= (0.)
283       MAT Zcentr= (0.)
284       MAT Zradii= (1.)
285     ELSE
286       MAT Reb= (1.)
287       MAT Imb= (0.)
288       MAT Rebcentr= (.5)
289       MAT Imbcentr= (0.)
290       MAT Bradii= (.5)
291       MAT Zcentr= (.5)
292       MAT Zradii= (.5)
293     END IF
294   !
295     FOR I=1 TO M
296       U(I)=Ko+C3*(I-1)              ! GENERATE U ARRAY
297     NEXT I
298   !
299     IF Hydsen$="Y" THEN             ! CALCULATE SENSITIVITY TERM
300       MAT Hsens= (0.)
301       IF Hydro$="D" THEN            ! DIPOLE SENSITIVITY
302         FOR J=1 TO M
303           Const=.5+.5*COS(U(J)*Senslen)
304           Hsens(J)=Const*Const
305           Const=-.5*SIN(U(J)*Senslen)
306           Hsens(J)=SQR(Hsens(J)+Const*Const)
307         NEXT J
308       ELSE                          ! CONTINUOUS SENSITIVITY
309         FOR J=1 TO M
310           Hsens(J)=ABS(SIN(U(J)*Senslen/2.)/(U(J)*Senslen/2.))
311         NEXT J
312       END IF
313     END IF
314   !
315     IF Nogroup<>1 THEN              ! CALCULATE GROUP RESPONSE
316       MAT Grres= (0.)
317       FOR J=1 TO M
318         Grim=0.
319         FOR I=1 TO Nogroup
```

```
320           Grres(J)=Grres(J)+COS(U(J)*Xgroup(I))
321           Grim=Grim-SIN(U(J)*Xgroup(I))
322         NEXT I
323         Grres(J)=SQR(Grres(J)*Grres(J)+Grim*Grim)/Nogroup
324       NEXT J
325     END IF
326   !
327     FOR J=1 TO M
328       Ref(J)=COS(D(Ndim)*U(J))          ! GENERATE F ARRAY
329       Imf(J)=-SIN(D(Ndim)*U(J))
330       FOR I=1 TO Ndim
331         Rea(J,I)=Ref(J)-COS(D(I)*U(J))  ! GENERATE Hk ARRAY
332         Ima(J,I)=Imf(J)+SIN(D(I)*U(J))
333         Rea(J,I)=Rea(J,I)*Grres(J)*Hsens(J)
334         Ima(J,I)=Ima(J,I)*Grres(J)*Hsens(J)
335       NEXT I
336     NEXT J
337   !
338     FOR I=1 TO M
339       Ref(I)=Ref(I)*Grres(I)*Hsens(I)
340       Imf(I)=Imf(I)*Grres(I)*Hsens(I)
341     NEXT I
342   !
343     N=Ndim-1
344     Itlog(1)=20*N                 !MAX ITERATION COUNT
345     Ioexit(1)=0                   !PRINT OPTION
346     Ts=TIMEDATE                   !INITIALIZE TIME
347     CALL Kaprox(N,Itlog(*),Ioexit(*),Ijsut(*))
348     Te=TIMEDATE-Ts                !EXECUTION TIME
349     Estore(N)=Cheb(Logp)
350     Icount(N)=Itlog(Logp)
351   !
352     Zsum=0.                 ! CALCULATE FINAL WEIGHT=1-SUM OF ALL OTHERS
353     Zsum=SUM(Z)
354     Z(Ndim)=1.-Zsum
355   !
356     IF Symflag THEN                     ! SYMMETRIZE WEIGHTS
357       FOR I=1 TO INT((Ndim)/2)
358         P=(Z(I)+Z(Ndim-I+1))/2
359         Z(I)=P
360         Z(Ndim-I+1)=P
361       NEXT I
362     END IF
363   !
364     IF Tme>0 THEN
365       REDIM Z(Ndim+Tme)
366       FOR I=1 TO Tme
367         FOR J=Ndim+I TO Misel(I) STEP -1
368           IF J>Misel(I) THEN Z(J)=Z(J-1)
369         NEXT J
370         Z(Misel(I))=0.
371       NEXT I
372     END IF
373   !
374     IF Equi$="Y" THEN
375       CALL Calcbeam(Ndim,M,Tme,Misel(*),Space,Z(*),Tbeam(*))
376     ELSE
377       CALL Unsymbeam(Ndim,M,Tme,Misel(*),Ko,K1,Z(*),Tbeam(*))
378     END IF
379   !
380     Zmax=MAX(Z(*))
381     IF Zmax<>0 THEN
382       FOR I=1 TO Ndim+Tme             ! NORMALIZE WEIGHTS TO 1
383         Z(I)=Z(I)/Zmax
384       NEXT I
385     END IF
```

B-8

```
386  !
387      CALL Weight_plot(Ndim+Tme,M/2,2,Tme,Misel(*),Space,Ko,K1,C3,Z(*),Tbeam(*
),Equi$)
388      OUTPUT 2 USING "#,B";255,75
389      PRINTER IS PRT
390      PRINT USING "@"
391      DUMP GRAPHICS
392  !
393      CONTROL CRT,12;2
394      FOR I=0 TO 9
395        ON KEY I LABEL "" GOSUB Dummy
396      NEXT I
397      ON KEY 1 LABEL " CONTINUE " GOTO Comp
398      ON KEY 2 LABEL " KEYS OFF/ON " GOSUB Flip_key
399      LOOP
400      END LOOP
401 Dummy: !
402      RETURN
403 Flip_key: !
404      Keflip=(Keflip+1) MOD 2
405      IF Keflip THEN
406        CONTROL CRT,12;1
407      ELSE
408        CONTROL CRT,12;2
409      END IF
410      RETURN
411 Comp: !
412      GRAPHICS OFF
413      OFF KEY
414      Weight$=""
415      REPEAT
416        INPUT "WOULD YOU LIKE A LIST OF THE WEIGHTS? (Y/N) [Y]",Weight$
417        IF LEN(Weight$)=0 THEN
418          Weight$="Y"
419        ELSE
420          Weight$=UPC$(Weight$[1])
421        END IF
422      UNTIL Weight$="Y" OR Weight$="N"
423  !
424      CALL Printinputs(Cachflag,Floatflag,Ndim,Tme,Sll,Itlog(*),Logp,Misel(*),
Space,Ko,K1,Te,Cheb(*),Z(*),Equi$,Weight$,Neguet$)
425  !
426      OUTPUT 2 USING "#,B";255,75        ! CLEAR SCREEN
427      Newko$=""
428      REPEAT
429        INPUT "WOULD YOU LIKE TO CALCULATE A NEW Ko or K1 TO GIVE A DIFF. BEAM
WIDTH (Y/N) [N]",Newko$
430        IF LEN(Newko$)=0 THEN
431          Newko$="N"
432        ELSE
433          Newko$=UPC$(Newko$[1])
434        END IF
435      UNTIL Newko$="Y" OR Newko$="N"
436      IF Newko$="Y" THEN GOTO New_ko
437  !
438      Wgtstore$=""
439      REPEAT
440        INPUT "WOULD YOU LIKE TO STORE THE WEIGHTS IN A DATA FILE? (Y/N) [N]",
Wgtstore$
441        IF LEN(Wgtstore$)=0 THEN
442          Wgtstore$="N"
443        ELSE
444          Wgtstore$=UPC$(Wgtstore$[1])
445        END IF
446      UNTIL Wgtstore$="Y" OR Wgtstore$="N"
447      IF Wgtstore$="Y" THEN
```

```
448       OUTPUT 2 USING "#,B";255,75      ! CLEAR SCREEN
449       INPUT "ENTER FILENAME FOR WEIGHT FILE: (10 CHARACTERS)",Filename$
450       OUTPUT 2 USING "#,B";255,75      ! CLEAR SCREEN
451       INPUT "ENTER MASS STORAGE DEVICE: [ :INTERNAL ]",Data_msus$
452       IF Equi$="Y" THEN
453         IF Ndim>30 THEN
454           CREATE BDAT Filename$&Data_msus$,2,256
455         ELSE
456           CREATE BDAT Filename$&Data_msus$,1,256
457         END IF
458         ASSIGN @Stordat TO Filename$&Data_msus$
459         OUTPUT @Stordat;Ndim+Tme,Space,Z(*)
460       ELSE
461         SELECT Ndim
462         CASE >47
463           CREATE BDAT Filename$&Data_msus$,4,256
464         CASE >31
465           CREATE BDAT Filename$&Data_msus$,3,256
466         CASE >15
467           CREATE BDAT Filename$&Data_msus$,2,256
468         CASE ELSE
469           CREATE BDAT Filename$&Data_msus$,1,256
470         END SELECT
471         ASSIGN @Stordat TO Filename$&Data_msus$
472         OUTPUT @Stordat;Ndim+Tme,D(*),Z(*)
473       END IF
474       ASSIGN @Stordat TO *
475   !
476     END IF
477     GRAPHICS ON
478     PAUSE
479     GRAPHICS OFF
480   END LOOP                  ! RETURNS TO Redo AT PROGRAM BEGINNING
481   END
```

A General Chebyshev Complex Function Approximation Procedure And An Application to Beamforming

R. L. Streit and A. H. Nuttall

A general Chebyshev complex function approximation procedure and an application to beamforming

R. L. Streit and A. H. Nuttall
Naval Underwater Systems Center, New London Laboratory, New London, Connecticut 06320

(Received 22 December 1981; accepted for publication 24 March 1982)

A new computational technique is described for the Chebyshev, or minimax, approximation of a given complex valued function by means of linear combinations of given complex valued basis functions. The domain of definition of all functions can be any finite set whatever. Neither the basis functions nor the function approximated need satisfy any special hypotheses beyond the requirement that they be defined on a common domain. Theoretical upper and lower bounds on the accuracy of the computed Chebyshev error are derived. These bounds permit both *a priori* and *a posteriori* error assessments. Efforts to extend the method to functions whose domain of definition is a continuum are discussed. An application is presented involving "re-shading" a 50-element antenna array to minimize the effects of a 10% element failure rate, while maintaining full steering capability and mainlobe beamwidth.

PACS numbers: 43.60.Gk, 43.30.Vh

LIST OF SYMBOLS

f	the given complex valued function to be approximated
$h_1,\ldots,h_n$	the given basis functions; linear combinations of these functions are used to approximate f
Q_m	the given finite point set; approximations to f are constructed on the m elements of this set. (Ordinarily, Q_m is a set of complex numbers; however, Q_m can be any finite set on which f and h_k are defined.)
$z_1,\ldots,z_m$	the elements of Q_m
$(a_1,\ldots,a_n)=a$	any vector of complex numbers used as coefficients of the basis functions $h_1,\ldots,h_n$
$e_n(z;a)$	the complex error "curve" of the approximation to f afforded by the coefficient vector a; defined by Eq. (A1)
$E_n(f)$	the actual maximum magnitude error committed by the "best" (i.e., Chebyshev, or minimax) approximation to f by linear combinations of $h_1,\ldots,h_n$; defined by Eq. (A2)
$\tilde{a}$	any coefficient vector for which $E_n(f)$ is actually attained; $\tilde{a}$ is itself approximated by $\hat{a}$ (see below)
C^n	the usual vector space of all n tuples of complex numbers
p	a given integer greater than or equal to 2; the larger p, the better will be the approximation of $\tilde{a}$ by $\hat{a}$ (see Theorems A1 and A2)
$\theta_1,\ldots,\theta_{2p}$	angles defined in Lemma A2 and dependent only on p
$R_n(z;a)$	the real part of the error curve $e_n(z;a)$
$I_n(z;a)$	the imaginary part of the error curve $e_n(z;a)$
$G_j(z;a)$	the projection of the error curve $e_n(z;a)$ onto the real axis of the complex plane after a rotation through the angle θ_j; defined by Eq. (A3)
$M_{np}(f)$	essentially an approximation to the number $E_n(f)$; defined by Eq. (A4)
$\hat{a}$	any coefficient vector for which $M_{np}(f)$ is actually attained; $\hat{a}$ is essentially an approximation to the vector $\tilde{a}$ (see above)
$\mathscr{E}_{np}(f)$	the maximum magnitude error committed using the coefficient vector $\hat{a}$; defined in Theorem A2
$u(z),v(z)$	the real and imaginary parts, respectively, of $f(z)$
$r_k(z),s_k(z)$	the real and imaginary parts, respectively, of the basis function $h_k(z)$
R,S	real $m\times n$ matrices whose entries in the jth row and kth column are $r_k(z_j)$ and $s_k(z_j)$, respectively. Used to construct matrix B (see below)
b_k,c_k	the real and imaginary parts, respectively, of the coefficient a_k of basis function h_k
$Bx=g$	the real overdetermined system on mp equations in $2n$ unknowns, whose Chebyshev solution yields a solution $\hat{a}$ to the problem $M_{np}(f)$; see the paragraph containing Eq. (A6) for details of construction
$\hat{B}\hat{x}=g$	analogous to $Bx=g$ when the solution vector $\hat{a}$ is forced to be a vector of real numbers; see Eq. (A7) for details
other	all notations *not* in this glossary are understood to be "local"; that is, they are used only in the context of the particular paragraphs which contain them

INTRODUCTION

The approximation of desired or given functional behavior by finite sets of simpler or specified basis functions is a recurrent problem in many fields. For example, in the mathematical field, we might wish to approximate a (desired) complex integral by a set of (simpler) sinusoidal components. Or in an antenna array processing application, we often want to realize a (given) low side-lobe behavior by means of an array with (specified) element locations which are not under our control.

For the case where the given functional behavior and the specified basis functions are all real valued and defined on a finite discrete data set, and where the approximation is afforded by a real-weighted linear combination of these basis functions, the optimum solution for minimizing the maximum magnitude error, i.e., the Chebyshev norm is in very good shape due to a fine algorithm given in Barrodale and Phillips.[1,2] Specifically, this algorithm solves the following mathematical problem: given real constants $\{f_i\}$, $\{h_{ik}\}$, where $1 \le i \le m$, $1 \le k \le n$, $m > n$, the real quantities $\{a_k\}_1^n$ are determined that minimize the maximum absolute value of the error residuals

$$e_i \equiv f_i - \sum_{k=1}^{n} a_k h_{ik}, \quad \text{for } 1 \le i \le m. \tag{1}$$

This algorithm has recently been used to good advantage in an array processing application to design some real symmetric weighting functions with very good side-lobe behavior, subject to constraints on the rate of decay of the distant side lobes.[3]

Here we wish to employ the algorithm, as described above for real variables in Eq. (1), for the minimization of the Chebyshev norm of

$$e_n(z;a) \equiv f(z) - \sum_{k=1}^{n} a_k h_k(z), \tag{2}$$

when $f(z)$ and $\{h_k(z)\}_1^n$ are complex, and z can take values in an arbitrary finite discrete point set. The weighting coefficients $\{a_k\}_1^n$ may be complex, or alternatively, they may be restricted to be real. Applications are afforded by an antenna array with arbitrarily specified element locations, but employing weights that are restricted to be real, or alternatively by array weights that are also allowed to be phased (complex). Numerical examples and applications of the technique, some efforts attempted for extending the method to a continuum of values of z, and a discussion constitute the rest of the main body of the paper. In the Appendix the basic mathematical theory and algorithm for the minimization of Eq. (2) is developed. Streit and Nuttall[4] present a FORTRAN program in a form which should be useful to readers interested in applying the technique to their own particular applications; unfortunately the listing is too long to include here. [A brief study of the appendix, especially with regard to Eq. (A6), should enable interested readers to write their own program.]

Although the above algorithm[1] is limited to a discrete set of points, it has been used fruitfully to minimize the continuous error [Eq. (2)] over a real variable z in the interval $[z_a, z_b]$, when f and $\{h_k\}$ are real, in the following manner. First, an initial set of $m > n$ real points $\{z_i^1\}_1^m$ was specified and the Chebyshev norm minimized in the usual fashion, resulting in the coefficient set $\{a_k^1\}_1^n$. For this set of optimum coefficients, the locations $\{z_i^2\}_1^l$ of the largest peaks of $|e_n(z;a)|$ were located, by setting the derivative $e_n'(z;a)$ to zero and solving numerically for $\{z_i^2\}_1^l$; the number l of such peaks will generally be less than m, but larger than n. [This approach presumes the availability of computable expressions for $f'(z)$ and $\{h_k'(z)\}_1^n$.] Then the modified set of points $\{z_i^2\}_1^l$ were used for another Chebyshev minimization, resulting in coefficient set $\{a_k^2\}_1^n$. Repetition of this procedure stabilized after a few trials with a unique set of $\{z_i\}_1^l$ at which the maximum errors were equal and irreducible. In the examples tried in Nuttall,[3] the number of peaks l at which the magnitude error $|e_n(z;a)|$ was largest and equal turned out to be $n+1$. Further discussion of this recursive approach is given in Sec. II.

Our method, as presented in the Appendix, is not inherently restricted to arrays of any particular geometry, but does assume that interelement effects (mutual coupling) can be ignored. In the most general case of a spatial or volumetric array, the method proposed here can still be applied. All the functions in Eq. (1) are then functions of spherical coordinates (θ,φ), so the finite domain of approximation becomes an appropriately chosen finite set $\{(\theta_k,\varphi_k)\}$ instead of a set of complex numbers. This difference does not in any way affect the mathematical properties of our method; rather it affects the size of the numerical problem to be solved and consequently, the computer effort required for its solution. For large enough arrays, such effort ultimately becomes prohibitive; where that point lies depends upon the designer and the application.

Although our method is applied only to single-frequency design problems for arrays, it can also be applied to broadband frequency design by sampling in frequency space as well. This again adds to the computer effort of solution, but does not affect the basic mathematical method.

We use no weighting function in Eq. (2), and so the resulting farfield beam patterns have a level side-lobe structure. For example, the classical Dolph–Chebyshev array design can be reproduced by our method. If such a level side-lobe structure is not desired, then use of an appropriate weighting function in Eq. (2) is easily incorporated into our method without altering the algorithm in any essential way.

I. APPLICATION TO ARRAY DESIGN WITH A CONSTRAINT

Consider a linear antenna array with N elements, located at arbitrary fixed positions $\{x_k\}_1^N$, receiving a plane-wave arrival of wavelength λ from direction θ_a, $-\pi/2 < \theta_a < \pi/2$, relative to a normal to the array. If the array is steered to look in direction θ_l, $-\pi/2 < \theta_l < \pi/2$, then the complex transfer function of the beamformer is given by

$$T(u) = \sum_{k=1}^{N} w_k \exp(-id_k u), \tag{3}$$

where $\{w_k\}_1^N$ are the element weights, and

$$d_k = 2\pi x_k/\lambda, \quad \text{for } 1 \le k \le N,$$

$$u = \sin\theta_o - \sin\theta_i.$$

Observe that the total range of u depends on the look direction θ_i; for example, if $\theta_i = 0$, then the range of u is the closed interval $[-1,1]$. The peak response of $T(u)$ should occur at $u = 0$, so we normalize (without loss of generality) according to

$$T(0) = 1 = \sum_{k=1}^{N} w_k.$$

To realize small side lobes, we must minimize $|T(u)|$ for all u values in some subset U of the total range of u. For example, if $\theta_i = 0$, the total range of u is $[-1,1]$, and U could be the union of intervals $[-1, -u_0]$ and $[u_0,1]$, where $u_0 > 0$ is chosen small relative to 1. For the special case of real weights, since from Eq. (3), $T(-u) = T^*(u)$, we could confine attention to $U = [u_0,1]$. The normalization constraint is most easily accounted for by solving for w_N and eliminating it; we obtain then

$$T(u) = \exp(-id_N u) - \sum_{k=1}^{N-1} w_k [\exp(-id_N u) - \exp(-id_k u)]. \quad (4)$$

This problem now fits the framework of Eq. (A1) in the appendix if we identify

$$z = u, \quad n = N-1, \quad e_n(z;a) = T(u),$$

$$f(z) = \exp(-id_N u), \quad a_k = w_k,$$

$$h_k(z) = \exp(-id_N u) - \exp(-id_k u),$$

$$Q_m = \text{finite subset of } U. \quad (5)$$

There has been no statement thus far as to the real or complex nature of the weights $\{w_k\}$. This distinction depends upon the application and the capability of the beamformer. Both cases fit the above framework; the only difference is that the number of unknowns to be solved for will be twice as large for the complex weights as for the real weights.

If the array is half-wavelength equispaced, then the computed element weights will be identical to the classical Dolph–Chebyshev weights and can, in this instance, be computed analytically. The general case of arbitrary spacings, however, cannot be computed analytically; yet the algorithm presented in this paper can always be applied.

In the remainder of this section, we presume that the elements are equispaced at half-wavelength. Then $x_k = k\lambda/2$ and Eq. (3) becomes

$$T(u) = \sum_{k=1}^{N} w_k \exp(-i\pi k u). \quad (6)$$

Observe now that $T(u)$ in Eq. (6) has period 2 in u, regardless of whether the weights $\{w_k\}$ are real or complex, or whether some elements have failed, i.e., zero weight values. This means that we can study and control $T(u)$ in Eq. (6) over any convenient u interval of length 2, and need not confine our investigation to $[-1,1]$. In particular, we concentrate on the u interval $[0,2]$ in the following.

As an illustration of the capability of the minimization technique of this paper, a 50-element, half-wavelength, equispaced linear array was initially designed for peak side lobes

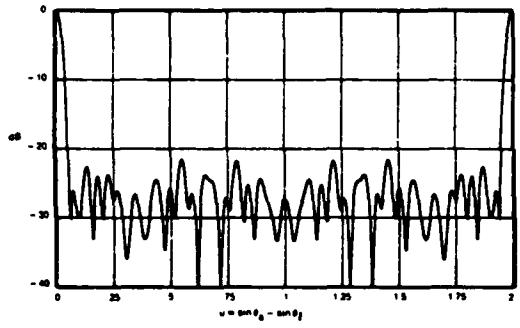


FIG. 1. Relative pattern for five elements failed.

of -30 dB relative to the main peak. This is of course a standard Dolph–Chebyshev case, and gives -30-dB side lobes throughout the u range $[u_0, 2-u_0]$, where $u_0 = 0.0538117$.[5] Then 10% of the elements were randomly eliminated from the array, but the remaining weights were unchanged; this corresponds to five elements failing in the array. The relative response of this particular array, with elements 7, 22, 40, 43, 50 failed, is illustrated in Fig. 1. The peak side lobe has increased from -30 to -21.58 dB, a degradation of 8.4 dB, and there is a large variety of different size peaks.

When our method with $p = 2$ and $m = 251$ equispaced points in $[u_0, 2-u_0]$ is applied to this defective array and the remaining 45 elements are weighted with real coefficients, subject to the constraints that the mainlobe width be the *same* as the ideal 50-element array and that the steering range in u be the *same*, the resultant array pattern is as displayed in Fig. 2. The peak side lobe is now -23.62 dB, an improvement of 2.04 dB over Fig. 1; however, there is still a significant variation in the values of the side lobes due to an insufficient number of phase controls, namely only $p = 2$.

The best real weights resulting from an increase in the parameter values to $p = 8$, $m = 501$ are displayed graphically in Fig. 3, and the corresponding array pattern is given in Fig. 4. The gaps in Fig. 3 at locations 7, 22, 40, 43, and 50 correspond to zero weighting at the failed elements. The general character of the weights is a bell-shaped one of all posi-

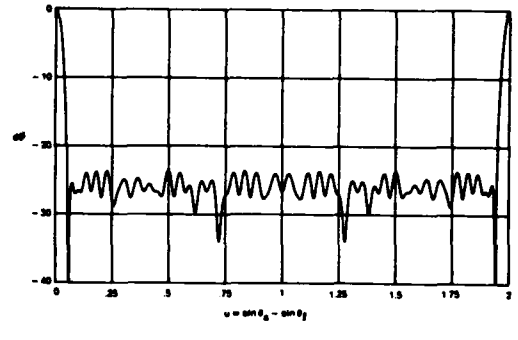


FIG. 2. Relative pattern for $p = 2$, $m = 251$, real weights.

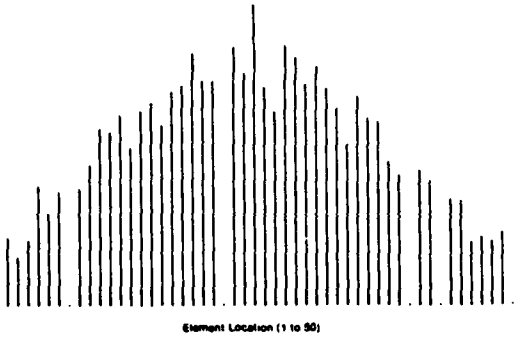


FIG. 3. Best real weights for $p = 8$, $m = 501$.

tive numbers, but there is significant fluctuation in the actual weight values, of the order of 10%. The pattern in Fig. 4 has a peak side lobe of -25.20 dB, an improvement of 3.62 dB over Fig. 1 but still 4.80 dB poorer than the ideal 50-element array.

When the weights were allowed to be complex and the maximum side lobe minimized in the *same* steering range $[u_0, 2 - u_0]$ for $p = 2$ and $m = 501$ equispaced points in $[u_0, 2 - u_0]$, the best complex weights turned out to be virtually pure real, and the corresponding pattern was almost identical to Fig. 2. A much improved pattern for complex weights was achieved when we took $p = 8$, $m = 501$; in fact, the best complex weights were real (within 10^{-6} relative error) and the pattern was the same as Fig. 4. Although we had anticipated a better pattern for the complex weight case than for the real weights, that did not materialize; the best complex weights for this equispaced linear array with five missing elements were real. The reason for this behavior is unknown, but it is an encouraging result from the array design viewpoint, for it indicates that there is no need to allow phasing at the individual elements; gain alone will achieve all the side-lobe reduction that can be achieved. This conclusion is drawn only for the half-wavelength equispaced line array with omnidirectional element response. (Recently, Lewis and Streit[6] proved, for a general line array steered through the same number of degrees either side of broadside, that within the collection of all sets of best complex weights there always exists a set of real weights. Thus it is not necessary to use complex weights in the case of line arrays to achieve best possible side-lobe levels.)

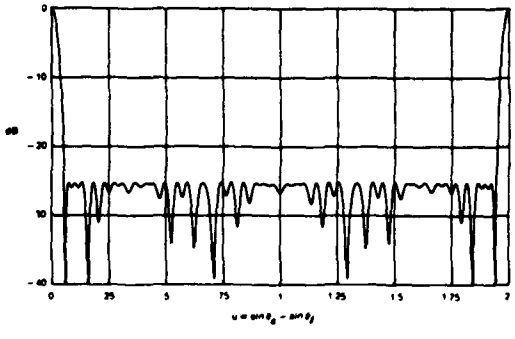


FIG. 4. Relative pattern for $p = 8$, $m = 501$.

The use of linear programming to design antenna arrays is not entirely new. In McMahon *et al.*[7] and Wilson,[8] linear programming was used to synthesize desired complex transfer functions to within 3 dB of the best possible side-lobe level. Their method corresponds identically to taking $p = 2$ in the method presented in this paper, i.e., treating only the real and imaginary parts of Eq. (2).

The computation of the real weights of Fig. 2 (where $p = 2$, $m = 251$, and $n = 44$) and of Fig. 4 (where $p = 8$, $m = 501$, and $n = 44$) required 1.2 min/205 simplex iterations and 38.4 min/402 simplex iterations, respectively. On the other hand, when the weights were allowed to be complex (replacing $n = 44$ by $n = 88$, but leaving p and m unchanged in both cases), the computations required 7.0 min/657 simplex iterations and 179 min/1262 simplex iterations, respectively. The two of these four cases requiring the smallest CPU times encountered almost no system overhead due to program size. However, the two cases requiring the largest CPU times encountered very significant system overhead because their large memory requirements caused significant usage of the virtual memory feature of the DEC VAX 11/780. The 38.4-min case required over 3.6 million page faults, while the 179-min case required over 11 million page faults. It is important to bear in mind that the DEC VAX 11/780 is essentially a minicomputer, and that without virtual memory, only the largest mainframe computers could have solved either of these two problems.

II. EFFORTS TO EXTEND THE METHOD

Our basic problem is to minimize the maximum magnitude of complex error

$$e_n(z;a) = f(z) - \sum_{k=1}^{n} a_k h_k(z) \tag{7}$$

over a continuum of values of z, when f, $\{h_k\}$, and $\{a_k\}$ are complex. We immediately approximate this desired problem by discretizing the z variable to a finite number of values, in order to make the problem computable. Furthermore, at any z value of interest, we additionally discretize the number of phase errors we are willing to consider. To be specific, since the algorithm in Barrodale and Phillips[1,2] applies only to real quantities, we consider the "projection" of a rotated version of the complex error:

$$P(z,\psi) = \mathrm{Re}\{\exp(i\psi)e_n(z;a)\}. \tag{8}$$

Then, since the argument of complex error [Eq. (7)] is unknown *a priori*, we let ψ take on a finite set of values spread over any π radian interval, and minimize the *magnitude* of projection [Eq. (8)] over all these selected ψ values. This is equivalent to the method of the Appendix.

In an effort to eliminate this second discretization process in ψ, a perturbation method was put forth[9] that claimed guaranteed convergence to the optimum weights for any given finite discrete set of z values. When applied to the examples in Barrodale *et al.*,[9] the proposed perturbation technique did indeed converge. However, when applied to the

following example, of approximation of $\exp(i3x)$ by the three basis functions 1, $\exp(ix)$, $\exp(i2x)$, over 100 equispaced points in the domain $[0,\pi/4]$ in x, it sometimes failed to converge, depending on the initial weights employed. The reason for this failure is that the "direction of the minimum" furnished by the perturbation is often totally irrelevant, and the best scale factor to apply to this perturbation is very small. Thus there occurs a small random meander in the coefficient space, and occasional convergence to a nonoptimum point. A modification of this technique was attempted wherein the magnitude of the perturbation was bounded. Although this improved the situation somewhat, convergence to the optimum was not always obtained.

It was thought that this meander in coefficient space might be eliminated by tracking the exact z values at which Eq. (7) is a maximum. Recall that in the real case discussed in the Introduction, convergence to the absolute optimum over a continuum of real z values was achieved in a practical example by re-evaluating the z points of maximum error and using these in a recursive approach. When this idea was extended to the two continuous variables z, ψ in Eq. (8), and only the $2n+1$ largest error points were retained, convergence was not obtained. When, however, the single "point" of a maximum, i.e., a pair of values (z_k,ψ_k), was replaced by a "patch", i.e., a set of values $\{(z_{kp},\psi_{kp})\}$ covering the maximum point (z_k,ψ_k), the convergence to the absolute optimum for the examples considered was apparently achieved. The patch width in ψ was of the order of a degree in most cases. The problem with this latter modification is that a large number of computations of the error function and its derivative must be evaluated, and the improvement over the method of the appendix is insignificant when p there is large.

If the final error in Eq. (7), after application of the method of the Appendix, is inadequate due to inadequate sampling in z and/or ψ, it is possible, for a given coefficient set $\{a_k\}$, to locate the point (z_m,ψ_m) at which Eq. (8) is largest, and then use a gradient approach to decrease this maximum error at (z_m,ψ_m). Of course, the particular point of maximum will jump around as the set $\{a_k\}$ is perturbed; nevertheless, the technique does converge (although slowly) and does lead to smaller errors at the maximum of Eq. (8) in a continuum for z and ψ.

III. DISCUSSION AND SUMMARY

It has been observed that two of the locations of maximum magnitude error often occur at the endpoints, if the specified domain in Eq. (2) is a real interval. (For example, see Figs. A1 and A2. The example of real coefficients in Fig. A1 had one of the maximum error points at one endpoint, but not the other. However, if we had specified domain $[-\pi/4,\pi/4]$ in that example, we would have observed four peak-error points, two of which would have been at endpoints, due to the conjugate property of the desired function and the basis functions.) Since the endpoints may be the only ones we can anticipate *a priori* and specify as locations of maximum error, an obviously useful procedure is to use more values of phase shift ψ in Eq. (8) [alternatively, the angles $\{\theta_j\}$ in Lemma A2] at the endpoints than in the interior, so as to better control these very likely locations of maximum error. For example, we might use $p=6$ in the interior of a specified real interval domain of z and use $p=12$ or 20 at the two endpoints. This does not add greatly to the total computation, since there are generally far more interior points than (two) endpoints. The program in Streit and Nuttall[4] may be readily used with different values of p at different data points.

The p different phase shifts ψ selected in Eq. (8) have been chosen here to be equally spaced over a 180° span (along with their 180° mates). This is the most reasonable selection in the absence of *a priori* knowledge of the complex error magnitude and phase because it gives the best upper bound in Lemma A2 of any set of phases. However, one could select any value of ψ to investigate the error; for example, different sets of values of ψ could be used at various values of abscissa z. The program in Streit and Nuttall[4] may be used with any desired set of phases at any, or all, of the data points.

The potential for significant round-off error accumulation is always present in linear Chebyshev complex function approximation. For example, in approximating $f(x)=\cos(12x)+i\sin(3x)$ by a complex linear combination of the 12 basis functions 1, $\exp(ix),\ldots,\exp(i11x)$ on the interval $[0,\pi/4]$, the complex coefficients of best approximation were observed to be large in magnitude and to lie in all quadrants of the complex plane; therefore significant numerical round-off error occurred during computation of the residuals within algorithm ACM495. Even if the coefficients of best approximation had happened to be better behaved, serious cancellation error may still occur in some problems because of the very nature of complex arithmetic. It might, therefore, be wise to use a double precision version of algorithm ACM495 routinely in complex Chebyshev approximation problems to alleviate such cancellation errors.

A sensitivity analysis on the optimum coefficients may be in order in some applications to determine their utility. This consideration is completely independent of their numerical accuracy. For example, in an antenna array design problem where some elements are spaced significantly less than a half-wavelength apart, it might well turn out that the optimum coefficients need to be specified with a relative error of better than 10^{-6}. Then, although the mathematical results may be correct and accurate, practical usage is precluded. This sensitivity can be determined by perturbing the optimum weights a few percent and observing if a drastic change occurs on the desired side-lobe behavior. (Such arrays are referred to as super-directive arrays.)

APPENDIX: MATHEMATICAL THEORY AND ALGORITHM

Let f and $h_1,\ldots,h_n$ be complex valued functions defined on the finite discrete point set $Q_m=\{z_1,\ldots,z_m\}$. For a complex vector $a=(a_1,\ldots,a_n)\in C^n$, define the complex error

$$f(z)-\sum_{k=1}^{n}a_k h_k(z)\equiv e_n(z;a),\quad z\in Q_m. \tag{A1}$$

The discrete linear Chebyshev approximation problem is to find a complex vector $\tilde{a}=(\tilde{a}_1,\ldots,\tilde{a}_n)\in C^n$ so that

$$E_n(f) \equiv \min_{a \in C^n} \max_{z \in Q_m} |e_n(z;a)| = \max_{z \in Q_m} |e_n(z;\tilde{a})|. \quad \text{(A2)}$$

The quantity $E_n(f)$ is called the discrete Chebyshev, or minimax, error of the approximation on the point set Q_m. (The restriction of $\tilde{a}$ to real values is discussed below.)

We do not solve this problem exactly. An algorithm presented in Barrodale *et al.*[9] for its solution is erroneous; we have discovered examples (see Sec. II) such that the recursive procedure described there need not converge to a solution of Eq. (A2). We will show that problem (A2) can be replaced by a related approximate problem solvable by available linear programming techniques. The exact solution of this related problem yields approximate solutions of Eq. (A2). The error in these approximate solutions to Eq. (A2) can be determined and, in fact, made arbitrarily small, using the results we prove below; see Theorems A1 and A2.

It can be shown by standard mathematical methods[10] that a vector $\tilde{a}$ satisfying Eq. (A2) exists, although it may not be unique. Sufficient conditions are known that result in unique $\tilde{a}$, but we do not need these conditions here. Therefore no further assumptions on $f, h_1,...,h_n$ or the point set Q_m are made. In order to proceed, we need the following results. Proofs of all these results are given in Streit and Nuttall.[4]

Lemma A1. If $z = x + iy$, where x and y are real, then

$$|z| = \max_{-\pi < \theta \leq \pi} (x \cos\theta + y \sin\theta).$$

Lemma A2. Let $\theta_j = \pi(j-1)/p$, $j = 1,2,...,2p$, where the integer $p \geq 2$. Let $z = x + iy$, and let

$$M = \max_{j=1,...,2p} (x \cos\theta_j + y \sin\theta_j).$$

Then

$$M \leq |z| \leq M \sec[\pi/(2p)].$$

We are now in a position to describe a problem that we can solve exactly and that is related to the given discrete linear Chebyshev approximation problem (A2). Let the real and imaginary parts of the complex error $e_n(z;a)$ be denoted by $R_n(z;a)$ and $I_n(z;a)$, respectively. For notational convenience, we define, for any complex vector $a \in C^n$,

$$G_j(z;a) = R_n(z;a)\cos\theta_j + I_n(z;a)\sin\theta_j, \quad j = 1,...,2p, \quad \text{(A3)}$$

where $\theta_1,...,\theta_{2p}$ are the angles given explicitly in Lemma A2. We seek a complex vector $\hat{a} = (\hat{a}_1,...,\hat{a}_n) \in C^n$ satisfying

$$M_{np}(f) \equiv \min_{a \in C^n} \max_{z \in Q_m} \max_{j=1,...,2p} G_j(z;a),$$

$$= \max_{z \in Q_m} \max_{j=1,...,2p} G_j(z;\hat{a}). \quad \text{(A4)}$$

With standard mathematical methods, it is easy to see that at least one such vector $\hat{a} \in C^n$ exists. The connection between the problem (A4) and the problem (A2) is explored in the next few results.

Theorem A1. Let $p \geq 2$ be an integer, and let $\theta_j = \pi(j-1)/p$, $j = 1,2,...,2p$. Then

$$M_{np}(f) \leq E_n(f) \leq M_{np}(f)\sec[\pi/(2p)].$$

Theorem A2. Let $p \geq 2$ be an integer, and let $\theta_j = \pi(j-1)/p$, $j = 1,2,...,2p$. Let

$$\mathscr{E}_{np}(f) = \max_{z \in Q_m} |e_n(z;\hat{a})|,$$

where the complex vector $\hat{a} \in C^n$ is any vector satisfying (A4). Then

$$E_n(f) \leq \mathscr{E}_{np}(f) \leq E_n(f)\sec[\pi/(2p)].$$

Corollary A2.1. Under the conditions of Theorem A2,

$$M_{np}(f) \leq E_n(f) \leq \mathscr{E}_{np}(f).$$

The preceding corollary evidently gives excellent upper and lower bounds on the discrete linear Chebyshev approximation error $E_n(f)$, and these bounds are readily available after the numerical computation of $\hat{a} \in C^n$ and $M_{np}(f)$ has been completed. We point out that the above two theorems substantially generalize results in Barrodale *et al.*,[9] p. 854.

Using the Maclaurin series for $\sec x$ in Theorem A2 gives the relative discrepancy

$$0 < \frac{\mathscr{E}_{np}(f) - E_n(f)}{E_n(f)} < \sec[\pi/(2p)] - 1$$

$$= \frac{\pi^2}{8p^2} + O\left(\frac{1}{p^4}\right), \quad \text{as } p \to \infty.$$

Note that this upper bound on the relative error is independent of f, the point set Q_m, the basis functions $\{h_k\}$, and n.

We will now explicitly formulate an overdetermined system of real linear equations to be solved in the Chebyshev norm (to be defined) which is equivalent to solving the problem (A4). Referring to the choice of θ_j's in Lemma A2, we observe that $\theta_{p+j} = \pi + \theta_j$, $j = 1,...,p$, and so, from Eq. (A3), we have

$$G_{p+j}(z;a) = -G_j(z;a), \quad j = 1,...,p.$$

Therefore, we may rewrite Eq. (A4) as

$$M_{np}(f) = \min_{a \in C^n} \max_{\substack{1 \leq i \leq m \\ 1 \leq j \leq p}} |G_j(z_i;a)|. \quad \text{(A5)}$$

Now, breaking the following quantities into their real and imaginary components

$$f(z) = u(z) + iv(z),$$

$$h_k(z) = r_k(z) + is_k(z), \quad k = 1,...,n,$$

$$a_k = b_k + ic_k, \quad k = 1,...,n,$$

we may write

$$R_n(z;a) = u(z) - \sum_{k=1}^{n} b_k r_k(z) + \sum_{k=1}^{n} c_k s_k(z),$$

$$I_n(z;a) = v(z) - \sum_{k=1}^{n} b_k s_k(z) - \sum_{k=1}^{n} c_k r_k(z),$$

$$G_j(z_i;a) = u(z_i)\cos\theta_j + v(z_i)\sin\theta_j$$

$$- \sum_{k=1}^{n} b_k [r_k(z_i)\cos\theta_j + s_k(z_i)\sin\theta_j]$$

$$- \sum_{k=1}^{n} c_k [r_k(z_i)\sin\theta_j - s_k(z_i)\cos\theta_j].$$

Note that $G_j(z_i;a)$ is a real linear equation in the $2n$ variables $\{b_k\}$ and $\{c_k\}$, and that all the coefficients of this equation are computable directly from known data.

Define the $mp \times 2n$ real matrix B in the partitioned form

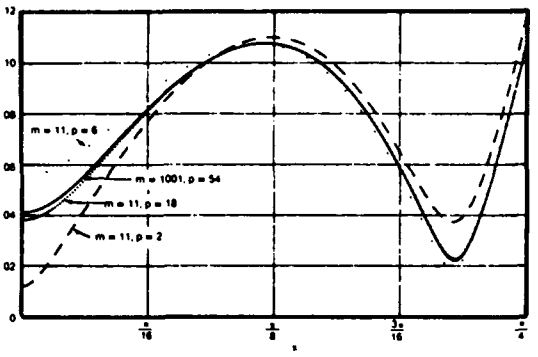


FIG. A1. Error curves for real coefficients; $m = 11$.

$$B = \begin{bmatrix} B_1 & D_1 \\ B_2 & D_2 \\ \vdots & \vdots \\ B_p & D_p \end{bmatrix},$$

with the $m \times n$ submatrices

$$B_t = R \cos\theta_t + S \sin\theta_t,$$
$$D_t = R \sin\theta_t - S \cos\theta_t, \qquad t = 1,\ldots,p,$$

where R and S are real $m \times n$ matrices defined by

$$R = [r_k(z_j)], \quad S = [s_k(z_j)].$$

Also, define the real vector

$$g \equiv [g_{11},\ldots,g_{1m},g_{21},\ldots,g_{2m},\ldots,g_{p1},\ldots,g_{pm}]^T$$

of length mp, where

$$g_{jt} = u(z_t)\cos\theta_j + v(z_t)\sin\theta_j, \quad t = 1,\ldots,m; \quad j = 1,\ldots,p.$$

Finally, define the real vector

$$x = [b_1,\ldots,b_n,c_1,\ldots,c_n]^T$$

of length $2n$. With this notation in hand, it is easily seen that the overdetermined system of mp equations in $2n$ unknowns

$$Bx = g \tag{A6}$$

has a residual error vector, defined by

$$g - Bx,$$

whose mp components are precisely the mp real numbers

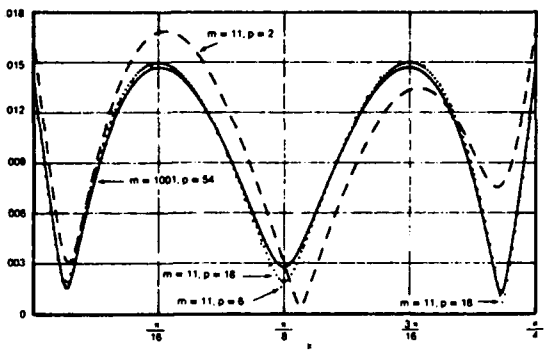


FIG A2. Error curves for complex coefficients; $m = 11$.

$G_j(z_t;a)$ arranged in a special order. Therefore the problem (A5) can be solved by computing a solution to the overdetermined linear system (A6) in the Chebyshev norm; i.e., the largest magnitude component of the residual vector $g - Bx$ is minimized over all choices of the vector x.

This equivalent problem in linear algebra can, in principle, be solved exactly and in a finite number of steps using linear programming methods.[1,2] Solutions of Eq. (A6) are not required to be unique; every solution of Eq. (A6) is a solution of Eq. (A5).

An excellent algorithm, which we will refer to as ACM 495, is available in the literature[1,2] for solving the overdetermined system of equations $Ax = b$. A linear program is set up and solved by the algorithm, so that knowledge of linear programming techniques is not necessary to use the algorithm in practice. The computational procedure, internal to the algorithm, actually solves the dual of the primal linear program using a modification of the simplex method. The dual formulation of this problem is available.[2,11] We will not discuss the details of the linear programming technique in this paper.

A very simple modification[9] of ACM 495 yields an algorithm for solving any real overdetermined system of linear equations in the Chebyshev norm subject to the additional constraints that all the residuals be non-negative. For a general system $Ax = b$, this problem takes the form

$$\underset{x_1,\ldots,x_s}{\text{minimize}} \max_{1 \le j \le r} \left(b_j - \sum_{k=1}^{s} a_{jk} x_k \right),$$

subject to the r constraints

$$b_j - \sum_{k=1}^{s} a_{jk} x_k \ge 0, \quad j = 1,\ldots,r.$$

The solution $x_1,\ldots,x_s$ returned by this modified algorithm is correct, even though the residuals returned may be in error. The correct residuals, if desired, must be calculated directly from the solution. Alternatively, if the residuals are required to be non-positive, then the same modified algorithm will work with A and b replaced by $-A$ and $-b$, respectively.

Requiring non-negative residuals in the overdetermined system (A6) has interesting geometrical interpretations. For example, if we take $p = 2$ in Lemma A2, then $\theta_1 = 0$ and $\theta_2 = \pi/2$. Thus $G_1(z;a)$ and $G_2(z;a)$ are merely the real and imaginary parts of the complex error $e_n(z;a)$, and the $2m$ components of the residual vector $g = Bx$ are precisely the real and imaginary parts of $e_n(z;a)$ evaluated in all m data points. Therefore, if the system (A6) is required to have non-negative residuals, we have forced the error curve to lie entirely in the first quadrant of the complex plane. More generally, we may always constrain $e_n(z;a)$ to lie in a given convex wedge-shaped sector of the complex plane with vertex at the origin, by making different, but appropriate, choices of the angles θ_1 and θ_2.

Suppose, finally, that the complex solution vector $a \in C^n$ of problem (A4) is required to be strictly real, while f and $\{h_k\}$ are complex. Then, in the vector x of Eq. (A6), $c_1 = \ldots = c_n = 0$. Thus the overdetermined system $Bx = g$ of mp equations in $2n$ unknowns can be replaced by a smaller system $\hat{B}\hat{x} = g$ of mp equations in only n unknowns,

TABLE AI. Coefficients for the real weight case.[12]

m	p	a_1	a_2	a_3
11	2	0.936738	− 2.443144	2.518388
	6	0.828404	− 2.280319	2.396455
	18	0.858547	− 2.321885	2.425096
	54	0.844146	− 2.301461	2.410611
101	2	0.936781	− 2.443223	2.518458
	6	0.831314	− 2.284548	2.399525
	18	0.865131	− 2.331446	2.432033
	54	0.853823	− 2.315301	2.420506
1001	2	0.936785	− 2.443232	2.518466
	6	0.831237	− 2.284448	2.399461
	18	0.865213	− 2.331571	2.432127
	54	0.853443	− 2.314772	2.420138

where the $m\,p \times n$ real matrix $\hat{B}$ is defined in partitioned form by

$$\hat{B} = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_p \end{pmatrix} \tag{A7}$$

where the $m \times n$ submatrices $B_1,\ldots,B_p$, are unchanged from (A6), and the real vector $\hat{x} = [b_1,\ldots,b_n]^T$. A solution of $\hat{B}\hat{x} = g$ in the Chebyshev norm can be computed using linear programming and algorithm ACM 495 as before.

We illustrate the procedure by approximating the complex function $f(x) = \exp(i3x)$ by a weighted sum of the basis functions $1, \exp(ix), \exp(i2x)$. That is, we seek to minimize the magnitude of the complex error curve

$$e_3(x) \equiv \exp(i3x) - \sum_{k=1}^{3} a_k \exp[i(k-1)x] \tag{A8}$$

over interval $[0,\pi/4]$, by choice of a_1,a_2,a_3, by solving the problem $M_{n,p}(f)$ of Eq. (A4). Two cases are of interest; in the first, the coefficients $\{a_k\}_1^3$ are restricted to be real, whereas in the second, these coefficients can be complex. The number m, of equispaced x values at which Eq. (A8) is sampled, is taken to be either 11, 101, or 1001, thereby ensuring that the smaller sample sizes are subsets of the larger sizes. The value of p, which is half the number of phase-shifted values of Eq. (A8) employed in the error minimization, is taken to be 2, 6, 18, 54, again ensuring the subset behavior of the smaller size cases. Note that p and the phase shifts $\{\theta_j\}$ are as given in Theorem A1.

The optimum real coefficients in Eq. (A8) for the problem $M_{n,p}(f)$ are given in Table AI for these choices of m and p, and a plot of the magnitude of the error for several representative cases is given in Fig. A1. The best approximation of all cases considered is afforded by $m = 1001, p = 54$, and its error curve is plotted as a solid line; its maximum error is 0.1078, which is realized at two points in the interval $[0,\pi/4]$. The cases for smaller m (less sampling of the abscissa) and smaller p (less sampling of the phase of the complex error) are poorer; for example, the maximum error for $m = 11, p = 2$ is 0.1184, realized at only one point, namely $x = \pi/4$.

We have not plotted the other error curves with real coefficients for $m = 101$ and 1001, because they are indistinguishable from Fig. A1, as a perusal of Table AI shows. For example, the coefficients for $m = 11, p = 2$ are very close to those for $m = 101$, $p = 2$ and $m = 1001$, $p = 2$. Thus our sampling in x is already "fine enough" at $m = 11$. However, there is a significant change in the coefficients as p is varied, for a fixed value of m; that is, $p = 2$ yields very coarse phase sampling of the error curve and should definitely be made larger.

The Chebyshev error curve ($m = 1001, p = 54$) in Fig. A1 realizes its maximum value at only $n - 1$ points, rather than at $n + 1$ points, where $n = 3$ is the number of coefficients for this example. This is probably related to the fact that we have minimized both the real and imaginary parts of the complex error, but have allowed ourselves to use only real coefficients.

The solution of the problem $M_{n,p}(f)$ for complex weights is given in Table AII for the same choices of m and p as above. Again, the change in coefficient values is more marked with p than with m. Magnitude-error curves for $m = 11$ and 101 are given in Figs. A2 and A3, respectively; the curves for $m = 1001$ are indistinguishable from those for $m = 101$ and are not presented.

The Chebyshev error curve ($m = 1001, p = 54$) is now symmetric about the midpoint of the interval of interest and has four equal error peaks of value 0.0147. This error is 7.3 times smaller than that for the real coefficient case. Also, the number of equal error peaks now equals 1 plus the number of coefficients; whether this property holds generally is not known.

TABLE AII. Coefficients for the complex weight case.

m	p	$\mathrm{Re}(a_1)$	$\mathrm{Im}(a_1)$	$\mathrm{Re}(a_2)$	$\mathrm{Im}(a_2)$	$\mathrm{Re}(a_3)$	$\mathrm{Im}(a_3)$
11	2	0.364737	0.954343	− 2.021670	− 2.119639	2.669023	1.153207
	6	0.378045	0.907888	− 2.016657	− 2.018598	2.648834	1.100488
	18	0.373079	0.898715	− 2.003032	− 2.003205	2.639992	1.094451
	54	0.371586	0.896504	− 1.999352	− 1.999473	2.637788	1.092947
101	2	0.362962	0.953469	− 2.018255	− 2.119960	2.667544	1.154238
	6	0.376532	0.904026	− 2.012095	− 2.014055	2.646131	1.099461
	18	0.370549	0.893500	− 1.995913	− 1.997062	2.635782	1.093144
	54	0.368950	0.890017	− 1.991172	− 1.991196	2.632622	1.090777
1001	2	0.362947	0.953499	− 2.018253	− 2.120028	2.667560	1.154275
	6	0.376502	0.903926	− 2.011979	− 2.013914	2.646047	1.099417
	18	0.370711	0.893848	− 1.996440	− 1.997545	2.636145	1.093278
	54	0.369179	0.890566	− 1.991954	− 1.991974	2.633175	1.091006

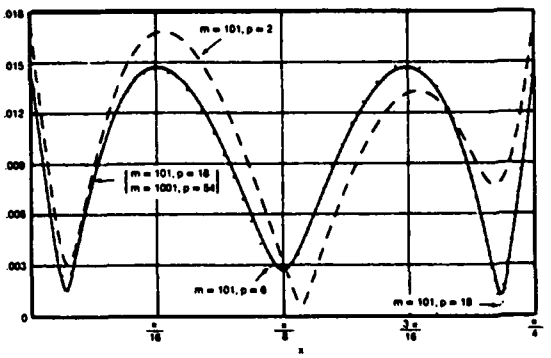


FIG. A3. Error curves for complex coefficients; $m = 101$.

TABLE AIV. Maximum magnitude error, computed over 2001 equispaced points in $[0,\pi/4]$.

m	p	Real coefficients	Complex coefficients
11	2	0.118396	0.017097
	6	0.108780	0.015142
	18	0.107890	0.015004
	54	0.107983	0.015005
101	2	0.118415	0.017329
	6	0.108893	0.014946
	18	0.107967	0.014733
	54	0.107813	0.014711
1001	2	0.118417	0.017331
	6	0.108902	0.014950
	18	0.107976	0.014735
	54	0.107821	0.014712

Upper and lower bounds on the discrete Chebyshev error $E_n(f)$ for the real and complex coefficient cases are given in Table AIII. These bounds are precisely those presented in Corollary A2.1. They correspond to sampling the complex error (A8) both in the abscissa x and in the phase of $e_3(x)$. The lower bounds monotonically increase with increasing m or p. The upper bounds decrease with increasing p, but increase with increasing m. All these trends follow from the fact that smaller sample sizes are subsets of the larger sizes.

However, the maximum magnitude error, evaluated over the continuum of x values in the interval $[0,\pi/4]$ (actually computed on a dense discrete sampling space), obeys none of the these monotonic relations, as Table AIV demonstrates. For example, the maximum error in the real case for $m = 11$, $p = 18$ is less than that for $m = 11$, $p = 54$. Also, the maximum error in the complex case for $m = 11$, $p = 6$ is greater than that for $m = 101$, $p = 6$. The reason for this behavior is that we have minimized a discrete approximation to our problem of interest, sampling both in the abscissa x and in the phase values of the complex error. However, the numerical discrepancies are small, as they must be for reasonably fine sampling in both variables. (A recursive gradient procedure could be used with any of these coefficient sets to improve the final maximum magnitude error if desired.)

Efficiency and timing estimates for actual calculation of complex Chebyshev approximations by the method of this paper is an important consideration in some applications. If we define an operation as consisting of a multiplication followed by an addition, then it is known[13] that the number of operations per simplex iteration required by algorithm ACM 495 is exactly the number of equations times the number of unknowns. In our case, the number of equations is $m\,p$, and the number of unknowns is $2n$ if the coefficients are complex, or n if the coefficients are required to be real. Thus the operation count per iteration is either $2nmp$ or nmp. The number of iterations required is difficult to estimate, since it depends on the particular problem. However, in randomly generated problems, it has been observed[13] that the number of iterations, I, is approximately the number of unknowns times some small constant c, where usually $1 \leqslant c \leqslant 3$. (Similar estimates have been observed[14,15] in more general linear programs as well.) Thus, in our case, $I = 2cn$ if the coefficients are complex and $I = cn$ if they are real.

The CPU time should be proportional to the total operation count, which equals the product of the number of iterations and the number of operations per iteration. That is, we expect the CPU time to be proportional to $n^2 m\,p$. For the particular example here, however, we obtain an excellent fit to the limited data in Table AV with the equation

TABLE AIII. Bounds on the discrete Chebyshev error $E_n(f)$.

m	p	Real coefficients		Complex coefficients	
		Lower bound	Upper bound	Lower bound	Upper bound
11	2	0.083718	0.118396	0.012089	0.017097
	6	0.105074	0.108780	0.013963	0.014456
	18	0.107307	0.107717	0.014143	0.014197
	54	0.107612	0.107658	0.014168	0.014174
101	2	0.083731	0.118414	0.012252	0.017328
	6	0.105192	0.108893	0.014436	0.014946
	18	0.107556	0.107967	0.014677	0.014733
	54	0.107767	0.107813	0.014703	0.014709
1001	2	0.083734	0.113418	0.012255	0.017331
	6	0.105191	0.108901	0.014440	0.014950
	18	0.107565	0.107976	0.014679	0.014735
	54	0.107775	0.107821	0.014704	0.014712

TABLE AV. Number of simplex iterations and CPU time.

		Real coefficients		Complex coefficients	
m	p	Simplex	CPU(s)	Simplex	CPU(s)
11	2	6	0.02	10	0.05
	6	8	0.08	15	0.16
	18	11	0.23	21	0.58
	54	13	0.81	27	2.25
101	2	7	0.25	10	0.40
	6	9	0.73	17	1.60
	18	13	2.65	21	5.78
	54	15	11.39	28	24.27
1001	2	9	3.05	13	5.00
	6	10	10.34	17	19.38
	18	13	48.16	24	105.47
	54	16	170.52	28	359.20

$$\text{CPU time(ms)} = 0.128\ n^{1.13} m^{1.18} p^{1.18},$$

where $n = 6$ if the coefficients are complex, and $n = 3$ if they are real. This fit was obtained by letting the exponents of n, m, and p vary separately. Other examples, however, lead us to anticipate that, more generally,

$$\text{CPU time} \propto n^2 (m\,p)^{1.2},$$

with a proportionality factor of the order of 0.01–0.03 ms, where n is either twice the number of approximation coefficients if the coefficients are complex, or exactly the number of coefficients if they are required to be real.

The CPU time estimates apply, of course, only to the DEC VAX 11/780 computer on which the calculations were performed. The virtual memory feature of this system allows very large problems to be solved; however, for sufficiently large problems, the system overhead incurred (page faulting, and so on) may significantly and adversely affect these estimates.

One method of detecting the presence of significant round-off errors is supplied by the nature of the approximation problem itself. That is, it can be proven that

$$M_{np}(f) \leqslant \mathscr{E}_{np}(f) \leqslant M_{np}(f) \sec[\pi/(2p)].$$

Once $M_{np}(f)$ and the coefficients have been computed in algorithm ACM 495, these bounds may be checked to see if significant numerical round-off error has occurred. In example (A8) above ($p = 6$, $m = 101$, complex coefficients), these inequalities were observed numerically to hold to five (but not six) significant digits. We conclude that the effects of round-off errors, although visible in the results, are not significant in this example. (Single precision numbers on the DEC VAX 11/780 have approximately seven significant decimal digits.)

[1] I. Barrodale and C. Phillips, "Solution of an Overdetermined System of Linear Equations in the Chebyshev Norm," Algorithm 495, ACM Trans. Math. Software 1, 264–270 (1975).

[2] I. Barrodale and C. Phillips, "An Improved Algorithm for Discrete Chebyshev Linear Approximation," *Proceedings of the Fourth Manitoba Conference on Numerical Mathematics*, edited by B. L. Hartnell and H. C. Williams (Utilitas Math., 1975).

[3] A. H. Nuttall, "Some Windows with Very Good Sidelobe Behavior," IEEE Trans. Acoust. Speech Signal Process. ASSP-29 (1) (1981); [also in NUSC Tech. Rep. 6239 (9 April 1980)].

[4] R. L. Streit and A. H. Nuttall, "Linear, Chebyshev Complex Function Approximation," NUSC Tech. Rep. 6403, Nav. Underwater Syst. Cent., New London, CT (26 February 1981).

[5] For an N-element array and $-t$ dB peak side lobes, we have $u_0 = (2/\pi)\arccos(1/z_0)$ where $2z_0 = [r + (r^2 - 1)^{1/2}]^{1/M} + [r - (r^2 - 1)^{1/2}]^{1/M}$, $r = 10^{t/20}$, and $M = N - 1$.

[6] J. T. Lewis and R. L. Streit, "Real Excitation Coefficients Suffice for Sidelobe Control in a Linear Array," IEEE Trans. Antennas Propag. (to appear); [also in NUSC Tech. Memo. No. 811114 (17 August, 1981)].

[7] G. W. McMahon, B. Hubley, and A. Mohammed, "Design of Optimum Directional Arrays Using Linear Programming Techniques," J. Acoust. Soc. Am. 51, 304–309 (1972).

[8] G. L. Wilson, "Computer Optimization of Transducer-Array Patterns," J. Acoust. Soc. Am. 59, 195–203 (1976).

[9] I. Barrodale, L. M. Delves, and J. C. Mason, "Linear Chebyshev Approximation of Complex-Valued Functions," Math. Computation 32, 853–863 (1978).

[10] G. Meinardus, *Approximation of Functions: Theory and Numerical Methods* (Springer-Verlag, New York, 1967), p. 1.

[11] I. Barrodale and A. Young, "Algorithms for Best L_1 and L_∞ Linear Approximations on a Discrete Set," Num. Math. 8, 295–306 (1966).

[12] In this case, we observe without proof that $a_1 + \sqrt{2}a_2 + a_3 \equiv 0$.

[13] I. Barrodale, private communication (18 December 1980).

[14] G. B. Dantzig, *Linear Programming and Extensions* (Princeton U. P., Princeton, NJ, 1963), p. 160.

[15] E. H. McCall, "Performance Results of the Simplex Algorithm for a Set of Real-World Linear Programming Models," Tech. Rep. 80-4, Comput. Sci. Dep., Univ. Minnesota, Minneapolis, MN (January 1980).

Optimization Of Discrete Arrays Of Arbitrary Geometry

R. L. Streit

Optimization of discrete arrays of arbitrary geometry

Roy L. Streit

New London Laboratory, Naval Underwater Systems Center, New London, Connecticut 06320
(Received 20 November 1979; accepted for publication 4 October 1980)

The concept of Directivity Index with Beamwidth Control (DIBC) leads to a practical method for the optimization of element excitations to control the tradeoff between beamwidth and sidelobe level in a discrete array of arbitrary configuration. This optimization procedure depends on the design frequency, specified element positions, individual element field patterns, and ambient noise field. Each of these factors can be specified in a completely general manner. In addition, the optimization procedure can be adapted to computers of modest memory size by using subarrays of the full array. Examples are included to show the versatility of this approach to the optimization problem, as well as its limitations. One of these examples is a 105-element cylindrical array.

PACS numbers: 43.60.Gk, 43.30.Vh, 43.28.Tc

I. THE CONCEPT

A. Introduction

Optimization of the element excitations of discrete antenna arrays is a matter of definition for three reasons. First, the definition of optimality will dictate the appropriate mathematical approach. Seemingly subtle changes in the definition of optimality can alter radically the applicable mathematical methods. Second, element excitations that are optimal in one sense are unlikely to be optimal in another sense. Two sets of excitations, each set optimal in its own sense, can be completely different. Third, the definition of optimality must reflect directly on the primary design goals for the array. It is pointless to optimize the Directivity Index (DI) and then complain that the sidelobes are too high, because the design goal of low sidelobes and the definition of optimality (maximum DI) are not directly related.

This article defines and uses exclusively the concept of Directivity Index with Beamwidth Control (DIBC). Several advantages, as well as difficulties, inherent in this definition are discussed. The primary difficulty in this definition is the requirement of large computer memories for large arrays. A technique employing subarrays of the full array in a systematic manner is shown to overcome this problem. The same technique can be used to solve the following problem as well: Given an array with known element positions and excitations, and given that new elements are to be introduced at known locations, how does one excite (or drive) these new elements to improve performance of the total array without changing the excitations of any of the elements of the original array?

The optimization procedure in this article is applicable when the following premises obtain:

(1) The wavelength, λ, of the design frequency is given and fixed.

(2) The number of elements, n, in the array is fixed and all the element positions (x_k, y_k, z_k), $k = 1, \ldots, n$ are known and fixed.

(3) Individual element field patterns at the design frequency are completely known.

(4) The ambient noise field at the design frequency is completely known.

(5) Element interactions can be ignored.

(6) Element excitations can be phased (i.e., complex).

The premise that the element excitations must be allowed to be phased is not necessary. As is pointed out later, we can just as easily require them to be strictly real, i.e., either positive or negative. However, except where noted, we assume that the excitations are phased because this is the more general situation and allows for better performance.

The concept of DIBC has been defined and used earlier by Butler and Unz.[1,2] In these papers, DIBC is called beam efficiency and is defined by them only for line arrays. This article is new in three regards. First, we apply the concept of DIBC to arbitrary spatial arrays and, thereby demonstrate its usefulness in very general situations. Second, we exhibit viable numerical procedures and techniques for overcoming a variety of mathematical difficulties inherent in the concept of maximizing DIBC. Third, the above-mentioned method of optimizing DIBC for general spatial arrays of any number of elements, while using only small amounts of core storage (and no peripheral storage devices), appears to be completely novel to this article.

All the examples in this article were computed on the Univac 1108 under EXEC 8. A listing of the computer program is available in Streit.[3] It is written in FORTRAN V for the general three-dimensional array of arbitrary configuration.

B. Field patterns and coordinate system

The spherical coordinate system of Fig. 1 is used throughout this article; however, a particular direction (θ, ϕ) will be specified by the direction cosines

$$\cos\alpha = \sin\phi\cos\theta, \quad \cos\beta = \sin\phi\sin\theta, \quad \cos\gamma = \cos\phi\,. \tag{1}$$

The most general field pattern treated here is

$$V(\theta,\phi) = \sum_{k=1}^{n} a_k R_k(\theta,\phi)\exp\left(\frac{2\pi i}{\lambda} d_k(\theta,\phi)\right), \tag{2}$$

where $R_k(\theta,\phi)$ is the phased (complex) response of the

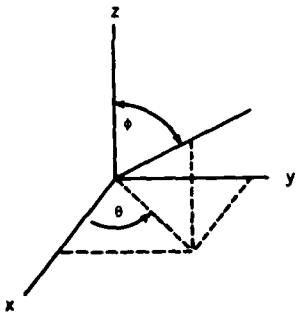


FIG. 1. The coordinate system.

kth element, and

$$d_k(\theta,\phi)=x_k\cos\alpha+y_k\cos\beta+z_k\cos\gamma\,. \tag{3}$$

Because of assumptions (1) to (6), the field pattern $V(\theta,\phi)$ depends solely on the phased (complex) excitations $a_1,\ldots,a_n$. The ambient noise field $N(\theta,\phi)$ will enter in the definition of optimal excitations. [Alternately, one may think of $N(\theta,\phi)$ as a given non-negative weighting function of the two angles.]

C. Directivity index with beamwidth control (DIBC)

The antenna designer is required to divide the set of all directions, denoted Ω, into three disjoint regions:

$\mathfrak{M}$ = mainlobe region,

$\mathcal{S}$ = sidelobe region,

$\mathcal{G}$ = ignored region $=\Omega-(\mathfrak{M}\cup\mathcal{S})$.

This division of directional space is completely arbitrary, except that neither $\mathfrak{M}$ nor $\mathcal{S}$ can be empty sets whereas $\mathcal{G}$ can be empty if desired. Once a particular choice of $\mathfrak{M}$, $\mathcal{S}$, and $\mathcal{G}$ has been made, the following definition of optimality is used.

Definition 1 The element excitations $a_1,\ldots,a_n$ are optimal excitations for a given choice of regions $\mathfrak{M}$, $\mathcal{S}$, and $\mathcal{G}$ if and only if the ratio

$$\mathrm{DIBC}=\frac{\iint_{\mathfrak{M}}N(\theta,\phi)|V^2(\theta,\phi)|\sin\phi\,d\phi\,d\theta}{\iint_{\mathfrak{M}\cup\mathcal{S}}N(\theta,\phi)|V^2(\theta,\phi)|\sin\phi\,d\phi\,d\theta} \tag{4}$$

is maximized. Any ratio of this form will be referred to as a directivity index with beamwidth control.

We point out that any excitations $a_1,\ldots,a_n$ that maximize the DIBC ratio (4) also maximize the ratio

$$\frac{\iint_{\mathfrak{M}}N(\theta,\phi)|V^2(\theta,\phi)|\sin\phi\,d\phi\,d\theta}{\iint_{\mathcal{S}}N(\theta,\phi)|V^2(\theta,\phi)|\sin\phi\,d\phi\,d\theta}. \tag{4a}$$

To see this, note that

$$\frac{1}{\mathrm{DIBC}}=\frac{\iint_{\mathfrak{M}\cup\mathcal{S}}}{\iint_{\mathfrak{M}}}=1+\frac{\iint_{\mathcal{S}}}{\iint_{\mathfrak{M}}},$$

so that any excitations minimizing the reciprocal of DIBC are also excitations that minimize the reciprocal of the ratio (4a), and this proves our assertion. This is *not to say*, of course, that the maximum value of (4) and the maximum value of (4a) are equal, only that excitations that maximize the one also maximize the other.

Maximizing DI is a limiting case of maximizing DIBC. To see this, recall that for a specified direction (θ_0,ϕ_0), DI is a maximum if the ratio

$$\mathrm{DI}=\frac{N(\theta_0,\phi_0)|V^2(\theta_0,\phi_0)|}{\iint_{\Omega}N(\theta,\phi)|V^2(\theta,\phi)|\sin\phi\,d\phi\,d\theta}, \tag{5}$$

is maximized. Now let the ignored region $\mathcal{G}$ be empty, let the mainlobe region, $\mathfrak{M}$, contain (θ_0,ϕ_0), and let $\mathcal{S}=\Omega-\mathfrak{M}$. Then, excitations maximizing DIBC converge to excitations that maximize DI as the mainlobe region, $\mathfrak{M}$, shrinks down on the point (θ_0,ϕ_0).

We have defined optimal excitations as those for which DIBC is maximized for some choice of regions $\mathfrak{M}$, $\mathcal{S}$, and $\mathcal{G}$. This allows a measure of control over the beamwidth and sidelobe level. By varying systematically the choice of $\mathfrak{M}$ and $\mathcal{S}$ and maximizing the DIBC for each choice, we can examine directly the tradeoff between beamwidth and sidelobe level for the particular array at hand. The engineer can, then, select those excitations that best suit his needs. Generally, the larger the mainlobe region, $\mathfrak{M}$, and the smaller the sidelobe region, $\mathcal{S}$ (for fixed ignored region, $\mathcal{G}$), the lower the overall sidelobe level and the greater the beamwidth. However, this may not always be the case, since sidelobe level does not enter directly into the DIBC ratio of (4). Nothing prevents the field pattern from having narrow high amplitude sidelobes, since such sidelobes contribute little to the integral in the denominator of the DIBC.

Another reason for maximizing DIBC is simply that it is conceptually easy to do so. All that is required is the solution of an eigenvalue/eigenvector problem (see Theorem 1), and problems of this type have been studied extensively in the literature.[4] Numerically, such problems require considerable care. Fortunately, well-designed computer programs are available for the solution of eigenproblems.[5,6] With the use of these routines, the solutions of the eigenproblems encountered in the antenna problem seem to be numerically stable. This is not to say that there may not be arrays that yield numerically unstable eigenproblems.

A final reason for maximizing DIBC is more esoteric. In the process of solving the required eigenproblem, all the eigenvalue/eigenvector pairs are computed, not merely the largest one. It happens that the field patterns corresponding to the lower order eigenvalues have some interesting features [see the figures in example (2)]. In addition, it often happens that some of the larger eigenvalues are close together; i.e., several linearly independent sets of excitations exist which give DIBC values that lie close together. (For an analogous situation, see Slepian and Pollak.[7]) What this means in the antennna problem is that, without sacrificing antenna performance (as measured solely

by the DIBC), it becomes a simple matter to examine numerous different sets of excitations with the aim of improving some completely different design goal of the array. [See (19) below.] This will not be discussed further in this article.

It must be mentioned that this approach to the array optimization problem does not attempt to address several issues that are of practical interest. First, this approach does not guarantee that the array performance is insensitive to perturbations in the optimum excitations. The question of sensitivity to excitation perturbation can be examined only after the optimum excitations are found. Second, this approach does not attempt to control the efficiency of the array. In other words, it can happen that the optimal excitations for a particular array may drive certain elements at their maximum allowed levels while the remaining elements are hardly driven at all, so that the total output power of the array is too low for the application. This problem is common to all amplitude shaded arrays and can be examined after the optimum excitations are found. Finally, this approach to array optimization ignores element interactions, so that it is possible for optimum excitations derived by this method (or by any other method for that matter) to have undesirable characteristics in this regard. This possibility, as well as the other two possibilities mentioned above, should be investigated after optimal excitations are found.

D. Computer storage problem

The primary drawback to maximizing DIBC is that the number of computer storage locations required (using the program in Streit[3]) is approximately

$$N_T = 6n^2 + 16n + 12\,000 \text{ words}, \tag{6}$$

for the case of constant ambient noise field and omnidirectional elements. Since the total requirement will grow as the ambient noise field and/or element field patterns require more storage to compute, it appears that the direct computation of optimal excitations for any array of 100 or more elements requires either large main-frame computers or computers with virtual memory. However, the storage requirements for maximizing DIBC can be avoided. A technique known as group coordinate relaxation[8] gives a method that can be tailored to the computer memory available. The technique is an excellent example of how to trade off computer memory for computational speed. The more memory available, the faster the DIBC can be maximized.

Group coordinate relaxation, in the context of maximizing DIBC, is simply stated. Suppose there are 300 elements in the array. Make any initial guess at the optimal excitations. Define distinct subarrays of, say, 50 elements each. By working with the first of these subarrays, new element excitations are computed for these 50 elements, so that the DIBC of the entire 300 element array is increased. Next, new excitations are computed for the second subarray. Cycling through all six subarrays in turn, until DIBC for the entire 300 element array cannot be increased further by changing the excitations in any of the subarrays, is the essence of group coordinate relaxation. The method can be proved to be convergent. It yields the globally best excitations, not merely locally best. A careful statement of the algorithm and further remarks are given in the subsection on numerical solution of the eigenproblem by the indirect method.

The rate of convergence of the group coordinate relaxation method depends heavily on the size of the subarrays used. The larger the subarrays, the faster the convergence, and the more core storage required. Thus, core storage is traded off in a direct manner for the convergence rate and, hence, for computation time. In addition, each step of the group coordinate relaxation method produces new excitations that increase the DIBC, so that if the computations are interrupted for any reason: (1) The last computed excitations are better than any of the excitations previously computed and (2) by saving the last computed excitations, the computations can be resumed without significant loss.

If n_s is the number of elements in a subarray used by the group coordinate relaxation process, the total storage required (using the program in Streit[3]) is approximately

$$N_R = 6n_s^2 + 8(n + n_s) + 12\,000 \text{ words}, \tag{7}$$

for the case of constant ambient noise field and omnidirectional elements. Thus, memory requirements grow as the square of the subarray size no matter how large the full array may be. By choosing the subarray size sufficiently small, the designer can maximize DIBC for large arrays on computers of modest size. The cost, however, is computer time. On the other hand, if the designer has a dedicated minicomputer of reasonable size, the cost of computer time is nil.

II. ELABORATION OF THE CONCEPT

A. DIBC and the eigenproblem

Let the vector $a = \langle a_1, \ldots, a_n \rangle^T$ be the vector of element excitations for the field pattern $V(\theta, \phi)$ given by (2). Then

$$\iint_{3\pi} N(\theta,\phi)|V^2(\theta,\phi)|\sin\phi\, d\phi\, d\theta = \iint_{3\pi} N(\theta,\phi)|V(\theta,\phi)|^2 \sin\phi\, d\phi\, d\theta$$

$$= \iint_{3\pi} N(\theta,\phi)\left[\overline{\sum_{k=1}^{n} a_k R_k(\theta,\phi)\exp\left(\frac{2\pi i}{\lambda} d_k(\theta,\phi)\right)}\right]\left[\sum_{j=1}^{n} a_j R_j(\theta,\phi)\exp\left(\frac{2\pi i}{\lambda} d_j(\theta,\phi)\right)\right]\sin\phi\, d\phi\, d\theta$$

$$= \sum_{k=1}^{n}\sum_{j=1}^{n} \bar{a}_k a_j \iint_{3\pi} N(\theta,\phi)\overline{R_k(\theta,\phi)} R_j(\theta,\phi)\exp\left(\frac{2\pi i}{\lambda}[d_j(\theta,\phi) - d_k(\theta,\phi)]\right)\sin\phi\, d\phi\, d\theta = \bar{a}^T U a, \tag{8}$$

where U is an $n \times n$ complex matrix. If $U = [u_{kj}]$, with k denoting the row number and j denoting the column number, then

$$u_{kj} = \iint_{\mathfrak{M}} N(\theta,\phi) R_j(\theta,\phi) \overline{R_k(\theta,\phi)} \times \exp\left(\frac{2\pi i}{\lambda}[d_j(\theta,\phi) - d_k(\theta,\phi)]\right) \sin\phi \, d\phi \, d\theta . \quad (9)$$

Clearly, U is a Hermitian matrix (i.e., $U = \bar{U}^T$), since it is obvious that $u_{kj} = \bar{u}_{jk}$. Also, U is positive definite, since

$$\bar{a}^T U a = \iint_{\mathfrak{M}} N(\theta,\phi) |V^2(\theta,\phi)| \sin\phi \, d\phi \, d\theta > 0 , \quad (10)$$

whenever the excitation vector $a \neq 0$ (and provided the mainlobe region, $\mathfrak{M}$, is not a set of measure zero, a pathological condition that is not encountered in this application). Therefore, for every mainlobe region, $\mathfrak{M}$, the matrix U defined in (9) is an $n \times n$ positive definite Hermitian matrix. Similarly,

$$\iint_{\mathfrak{M} \cup \mathfrak{S}} N(\theta,\phi) |V^2(\theta,\phi)| \sin\phi \, d\phi \, d\theta = \bar{a}^T W a , \quad (11)$$

where $W = [w_{kj}]$ is an $n \times n$ positive definite Hermitian matrix whose general entry is

$$w_{kj} = \iint_{\mathfrak{M} \cup \mathfrak{S}} N(\theta,\phi) R_j(\theta,\phi) \overline{R_k(\theta,\phi)} \times \exp\left(\frac{2\pi i}{\lambda}[d_j(\theta,\phi) - d_k(\theta,\phi)]\right) \sin\phi \, d\phi \, d\theta . \quad (12)$$

Thus, for a given choice of $\mathfrak{M}$, $\mathfrak{S}$, and θ, we have

$$\text{DIBC} = \bar{a}^T U a / \bar{a}^T W a , \quad (13)$$

which is a ratio of positive definite Hermitian forms. Therefore, optimal excitations are those that maximize this ratio of Hermitian forms.

The mathematical tools for handling ratios of the form (13) have been known for at least a century. We have the following general mathematical result.

Theorem 1: If U and W are $n \times n$ Hermitian matrices and W is positive definite, then the eigenvalues of the generalized eigenproblem

$$Uz = \mu W z \quad (14)$$

are all real. Let $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$ denote these eigenvalues. Then, linearly independent vectors $z_1, \ldots, z_n$ can be found that satisfy

$$U z_k = \mu_k W z_k , \quad k = 1, \ldots, n , \quad (15)$$

and

$$\bar{z}_k^T W z_j = \begin{cases} 1, & \text{if } k = j \\ 0, & \text{if } k \neq j \end{cases} . \quad (16)$$

The vectors $z_1, \ldots, z_n$ are called the eigenvectors of the eigenproblem (14). Also, we have

$$\max_{z \neq 0}\left(\frac{\bar{z}^T U z}{\bar{z}^T W z}\right) = \mu_1 , \quad (17)$$

and this maximum is attained for every eigenvector corresponding to μ_1, and

$$\min_{z \neq 0}\left(\frac{\bar{z}^T U z}{\bar{z}^T W z}\right) = \mu_n , \quad (18)$$

and this minimum is attained for every eigenvector corresponding to μ_n. Finally, if $1 \leq k \leq n$, then, for any constants $\alpha_1, \ldots, \alpha_k$ not all zero, we have

$$\mu_1 \geq \bar{z}^T U z / \bar{z}^T W z \geq \mu_k , \quad (19)$$

where $z = \alpha_1 z_1 + \cdots + \alpha_k z_k$.

The proofs of the various parts of this theorem can be found in numerous sources, e.g., Gantmacher.[4]

For the immediate purposes, the most important part of this theorem is (17). It states that optimal excitations are precisely the components of any eigenvector corresponding to the largest eigenvalue of the generalized eigenproblem $Uz = \mu W z$, where U and W are defined by (9) and (12).

Theoretically, Theorem 1 solves the problem of maximizing DIBC in the case where all element excitations can be phased. But what is the solution if all the excitations are required to be real (positive or negative)? In this case, the ratio (13) still holds, but the excitation vector, a, is real, i.e., $a = \bar{a}$. Since U and W are Hermitian, we have the algebraic identity

$$\text{DIBC} = \frac{\bar{a}^T U a}{\bar{a}^T W a} = \frac{a^T (\text{Re}U) a}{a^T (\text{Re}W) a} . \quad (20)$$

Now, ReU and ReW are both real symmetric matrices, and all the properties of Theorem 1 hold for the real generalized eigenproblem $(\text{Re}U)z = \mu(\text{Re}W)z$. The only difference is that now the eigenvectors have all real components. Therefore, if the excitations are required to be real, the optimal real excitations are precisely the components of any eigenvector corresponding to the largest eigenvalue of the generalized real eigenproblem

$$(\text{Re}U)z = \mu(\text{Re}W)z , \quad (21)$$

where U and W are the matrices defined by (9) and (12).

In the remainder of this article, we concern ourselves only with phased excitations. Everything that we do, however, can be recast for real excitations simply by using the real parts of the matrices involved.

A discrete reformulation of DIBC is discussed in the next section. By way of analogy only, this discrete version of the DIBC ratio is to DIBC as the discrete Fourier transform is to the Fourier transform. Following this is a discussion of the numerical methods for the solution of the kind of eigenproblems encountered in this article.

B. A discrete version of DIBC

Maximizing the DIBC ratio (4) is mathematically tractable, but it is not practical. It requires the solution of an eigenproblem, which in turn requires the evaluation of approximately n^2 double integrals (9) and (12) over subsets of the unit sphere. Since it is essential that the mainlobe region, $\mathfrak{M}$, and the sidelobe region, $\mathfrak{S}$, be quite general in nature (i.e., be defined to suit the particular application), these double integrals are in general impossible to evaluate explicitly and are

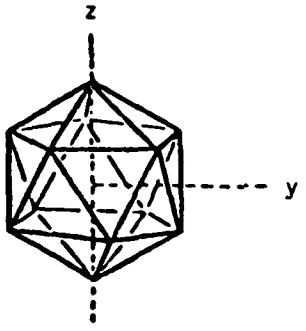


FIG. 2. The icosahedron.

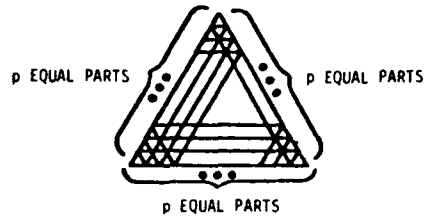


FIG. 3. One face of icosahedron subdivided into p parts.

also difficult and time consuming to evaluate accurately by numerical methods. For these reasons, DIBC itself is not optimized. What is optimized is a discrete version (DIBCF) of DIBC that is not only numerically practical to use, but is also conceptually simple.

The discrete DIBC definition replaces the surface integrals in ratio (4) by discrete sums over points chosen in $\mathfrak{M}$ and $\mathcal{S}$. Since $\mathfrak{M}$ and $\mathcal{S}$ are not known *a priori*, these points are distributed uniformly over the surface of the sphere, with each point contributing one term to the discrete sum and all terms entering with equal weight. Ideally, then, these points must show no directional bias and must be easy to compute. Furthermore, it must be possible to choose these points with any desired density on the sphere.

A natural choice for points fulfilling these conditions is easy to describe, but difficult to compute. Choose as points the equilibrium positions of a finite number of positive charges constrained to lie on the surface of the unit sphere. When the number of positive charges is 4, 6, 8, 12, or 20, it is intuitively clear that stable points for these charges are at the vertices of the five regular Platonic bodies: the tetrahedron, the octahedron, the cube, the icosahedron, and the dodecahedron, respectively. Unfortunately, these are the only easy cases (see Melnyk *et al.*[9]).

The discrete points chosen to define discrete DIBC are the vertices of a geodesic dome. Consider the icosahedron shown in Fig. 2. Note that in this figure the y axis is in the plane of the paper and the z axis is tilted slightly to show off the configuration. (The x axis is not shown, but is, of course, orthogonal to the yz plane.) This regular figure has 12 vertices, 20 faces, and 30 edges. Geodesic domes with (almost) any number of faces are constructed from the icosahedron by subdividing its equilateral triangular faces in a systematic manner.[10] First, subdivide each face into congruent equilateral subtriangles, as shown in Fig. 3; i.e., for each positive integer $p \geq 1$, find $p+1$ equispaced points along each edge and pass lines through each of these points parallel to the other two edges. Next, take all the vertices of the equilateral subtriangles so generated and project them on the unit sphere. By doing this for each face of the icosahedron for a fixed integer $p \geq 1$, we construct the vertices of a geodesic dome of order p. We define the Fuller points, $\mathfrak{F}_p$, to be the totality of these points.

The Fuller points, $\mathfrak{F}_p$, are uniquely oriented in Cartesian space once the vertices of the icosahedron are defined. With some simple trigonometry, it can be seen that the 12 vertices of an icosahedron inscribed in a sphere of unit radius can be taken to be the 2 points $(0,0,\pm 1)$, together with the 10 points

$$[2b(1-b^2)^{1/2}\cos 2\pi k/5,\ 2b(1-b^2)^{1/2}\sin 2\pi k/5,\ 2b^2-1]$$

$$[2b(1-b^2)^{1/2}\cos 2\pi(k+\tfrac{1}{2})/5,$$

$$2b(1-b^2)^{1/2}\sin 2\pi(k+\tfrac{1}{2})/5,\ 1-2b^2], \qquad (22)$$

where $k=0,1,\ldots,4$ and $b=1/(2\cos 3\pi/10)=2/\mathrm{SQRT}[10-2\mathrm{SQRT}(5)]$. The edge length of this icosahedron is $2\mathrm{SQRT}(1-b^2)\approx 1.0515$.

How many points are there in $\mathfrak{F}_p$? By inspecting an unfolded paper model of the icosahedron on which the Fuller points have been marked, it is easy to see that $\mathfrak{F}_p$ contains exactly $10p^2+2$ points. Thus, the number of steradians per point is approximately $4\pi/10p^2 \approx 1.25/p^2$.

Notice that the Fuller points, $\mathfrak{F}_p$, are not quite ideal. Those points chosen near the center of a face of the original icosahedron will be less finely spaced when projected on the sphere than will those points that were chosen nearer an edge. This defect in $\mathfrak{F}_p$ does not seem to be significant in this application. With the Fuller points defined, we state the following.

Definition 2: For a given integer $p \geq 1$, and regions $\mathfrak{M}$, $\mathcal{S}$, and $\mathcal{B}$, the element excitations $a_1,\ldots,a_n$ are optimal if and only if the ratio

$$\mathrm{DIBC\,F}=\frac{\left(\sum_{(\theta,\phi)\in\mathfrak{F}_p\cap\mathfrak{M}} N(\theta,\phi)|V^2(\theta,\phi)|\right)}{\left(\sum_{(\theta,\phi)\in\mathfrak{F}_p\cap(\mathcal{N}\mathfrak{M}\cup\mathcal{S})} N(\theta,\phi)|V^2(\theta,\phi)|\right)}, \qquad (23)$$

is maximized. Any ratio of the form of (23) will be referred to as a directivity index with beamwidth control over the Fuller points, $\mathfrak{F}_p$.

Note that, as $p\to\infty$, we do not have DIBCF $\to$ DIBC because the distribution at the Fuller points does not approach the uniform distribution as p gets large. In addition, we point out that for every $p \geq 1$, we have the inequalities

$$0 \leq \mathrm{DIBCF} \leq 1,$$

provided only that the denominator sum in (23) is non-

zero. The proof of the lower bound is trivial, and the proof of the upper bound follows from the observation that every summand in the numerator of (23) appears also in the denominator.

The formulation of DIBCF as an eigenproblem parallels that for DIBC. Specifically, we have

$$Mz = \mu Sz , \tag{24}$$

where $M = [m_{kj}]$ and $S = [s_{kj}]$ are $n \times n$ positive definite Hermitian matrices with

$$m_{kj} = \sum_{(\theta,\phi)\in\mathfrak{F}_p\cap\mathfrak{R}} N(\theta,\phi)R_j(\theta,\phi)\overline{R_k(\theta,\phi)} \times \exp\left\{(2\pi i/\lambda)[d_j(\theta,\phi) - d_k(\theta,\phi)]\right\} , \tag{25}$$

$$s_{kj} = \sum_{(\theta,\phi)\in\mathfrak{F}_p\cap(\mathfrak{R}\cup\mathfrak{S})} N(\theta,\phi)R_j(\theta,\phi)\overline{R_k(\theta,\phi)} \times \exp\left\{(2\pi i/\lambda)[d_j(\theta,\phi) - d_k(\theta,\phi)]\right\} . \tag{26}$$

By Theorem 1, maximizing DIBCF requires the computation of any eigenvector corresponding to the largest eigenvalue for the eigenproblem (24). The numerical solution of (24) is discussed fully in the next section.

There are two considerations that should enter into the particular choice of p for the Fuller points $\mathfrak{F}_p$. First, the Fuller points should be numerous enough to sample adequately the worst behavior of any realizable field pattern. In other words, p should be large enough that even the narrowest sidelobe achievable in the field pattern will contain points in $\mathfrak{F}_p$. Second, Theorem 1 requires that the denominator matrix S of the DIBCF ratio be positive definite. Normally, the sampling criterion will effect this automatically.

C. Numerical solution of the eigenproblem: Direct method

The eigenproblem (24) is equivalent to the eigenproblem

$$(S^{-1}M)z = \mu z . \tag{27}$$

In other words, the eigenvalues and eigenvectors of (27) are precisely the same as those of (24). There are two difficulties in using (27) for numerical computation. First, it requires the inverse of the matrix S, whose only special structure is that it is positive definite and Hermitian. In general, numerical computation of the inverse of matrices should be avoided if possible. Second, (27) is not a Hermitian eigenproblem; i.e., $S^{-1}M$ is not necessarily Hermitian even though S and M are both Hermitian. This means that the eigenvalues and eigenvectors of (27) must be computed by a routine designed for a general complex matrix, and this means that the eigenvalues can (and do) turn out to be complex numbers because of numerical roundoff. Since Theorem 1 requires that all the eigenvalues be strictly real numbers, there is numerical error in using (27) caused by destruction of the natural Hermitian symmetry in (24). For these reasons, it is desirable to solve the eigenproblem (24) directly.

Martin and Wilkinson give a method and a routine for solving this eigenproblem when M and S are real symmetric. Both the technique and the routine can be adapted to the Hermitian case. Every Hermitian positive definite matrix S has the Cholesky decomposition

$$S = L\bar{L}^T , \tag{28}$$

where L is a lower triangular matrix. Thus,

$$Mz = \mu L\bar{L}^T z , \quad L^{-1}Mz = \mu\bar{L}^T z , L^{-1}M(\bar{L}^{-T}\bar{L}^T)z = \mu\bar{L}^T z ,$$

$$(L^{-1}M\bar{L}^{-T})x = \mu x , \tag{29}$$

where

$$x = \bar{L}^T z . \tag{30}$$

Therefore, the eigenvalues of $L^{-1}M\bar{L}^{-T}$ are precisely the eigenvalues of (24), and the eigenvectors x of $L^{-1}M\bar{L}^{-T}$ and the eigenvectors z of $S^{-1}M$ are related by (30). Note, also, that (29) is a Hermitian eigenproblem, since $L^{-1}M\bar{L}^{-T}$ is a Hermitian matrix. It is, therefore, possible to solve (29) by numerical methods designed for Hermitian eigenproblems that explicitly use the fact that the eigenvalues are real.[5] Therefore, the eigenvalues computed by using (29) will always be real, as required.

This computational procedure seems to require a prohibitively large number of arithmetic operations; however, the computations may be done very efficiently because of the special structure of the matrices involved. For example, the matrix $L^{-1}M\bar{L}^{-T}$ can be computed (without inverting the matrix L) by using only $\frac{2}{3}n^3$ complex multiplications. This compares to $\frac{1}{2}n^3$ complex multiplications in the computation of S^{-1} alone in (27). In terms of storage required, computation time, and numerical accuracy, the use of (29) and (30) is preferable to the use of (27).

The routine in Martin and Wilkinson[6] was adapted to the Hermitian case, using routines in Ref. 5 to solve the ordinary eigenproblem. This routine is called PENCLH, and its listing is available in Streit.[3] (The listings of the routines used from Ref. 5 are not available; they are proprietary information under terms of the lease arrangements made with International Mathematical and Statistical Libraries, Incorporated.) Finally, it is pointed out that the routine PENCLH computes all the eigenvalues and eigenvectors of (24), and not merely the largest eigenvalue and corresponding eigenvector(s).

D. Numerical solution of the eigenproblem: Indirect method

As discussed in the section on the computer storage problem, the drawback to the direct method is excessive computer storage for large arrays. The group coordinate relaxation (or indirect) method overcomes this drawback, but at the cost of computer time and the loss of ability to compute the lower order eigenvalues eigenvectors. The group coordinate relaxation method is detailed by Faddeev and Faddeeva[8] for the real symmetric eigenproblem $Ax = \mu x$. This method can be extended easily to the Hermitian eigenproblem

$$Mz = \mu Sz . \tag{31}$$

Although the method can be extended to arbitrary Her-

mitian matrices M and S, with S positive definite, it is important here to retain the structure of M and S as given by (25) and (26). The reason is that the Hermitian forms of M and S can be evaluated directly without knowledge of any of the entries of either matrix. This is the fact that allows the computer storage problem to be overcome.

The following notation will be very useful. Define the basis vectors

$$\begin{aligned} e_1 &= \langle 1\ 0\ 0\ \ldots\ 0\ 0\rangle^T, \\ e_2 &= \langle 0\ 1\ 0\ \ldots\ 0\ 0\rangle^T, \\ &\vdots \\ e_n &= \langle 0\ 0\ 0\ \ldots\ 0\ 1\rangle^T. \end{aligned} \tag{32a}$$

Note that each of these vectors is of dimension n. To define vectors e_m for $m \geq n+1$, we first set

$$l(m) = \begin{cases} n, & \text{if } m \text{ is an integral multiple of } n \\ m - [m/n]n, & \text{if not} \end{cases} \tag{32b}$$

where $[\ ]$ denotes the greatest integer function. Since (32b) requires that $1 \leq l(m) \leq n$, we can now define

$$e_m = e_{l(m)}, m \geq n+1. \tag{32c}$$

In other words, we have defined

$$\begin{aligned} e_1 &= e_{n+1} = e_{2n+1} = \ldots, \\ e_2 &= e_{n+2} = e_{2n+2} = \ldots, \\ &\vdots \\ e_n &= e_{2n} = e_{3n} = \ldots. \end{aligned} \tag{33}$$

Before the group coordinate relaxation algorithm can begin, two items must be specified. First, an initial guess

$$a_{(0)} = \langle a_1^{(0)}, a_2^{(0)}, \ldots, a_n^{(0)}\rangle^T, \tag{34}$$

for the optimal element excitation vector is required. The vector $a_{(0)}$ should not contain all zero entries, but it is completely arbitrary otherwise. Second, it must be decided in some manner to work with subarrays of the full array of size $r \geq 1$. It will be shown that choosing to work with subarrays of size r will mean that generalized eigenproblems of size $r+1$ will have to be solved, so computer storage plays an important role in the choice of r. Another important consideration is computation time. In general, the larger r is taken to be, the faster optimum excitations of the full array can be computed.

The group coordinate relaxation algorithm is most easily described by exhibiting the first two steps of the algorithm. From these steps it is easy to see the general procedure. In the first step, we seek to

$$\underset{x \in Q_0}{\text{maximize}} \frac{\bar{x}^T M x}{\bar{x}^T S x}, \tag{35}$$

where Q_0 is the vector space of dimension $r+1$ whose general element, x, can be written in the form

$$x = c_0 a_{(0)} + c_1 e_1 + \ldots + c_r e_r, \tag{36}$$

for some complex constants $c_0, c_1, \ldots, c_r$. It is shown that (35) is a ratio of Hermitian forms in the parameters $c_0, c_1, \ldots, c_r$. Therefore, by Theorem 1, the solution of (35) requires solving an eigenproblem of size $r+1$. Let

$$a_{(1)} = \tilde{c}_0 a_{(0)} + \tilde{c}_1 e_1 + \ldots + \tilde{c}_r e_r, \tag{37}$$

be a vector for which the maximum (35) is attained. This completes the first step. In the second step, we seek to

$$\underset{x \in Q_1}{\text{maximize}} \frac{\bar{x}^T M x}{\bar{x}^T S x}, \tag{38}$$

where Q_1 is the vector space of dimension $r+1$ whose general element, x, can be written in the form

$$x = c_0 a_{(1)} + c_1 e_{r+1} + \ldots + c_r e_{2r}, \tag{39}$$

for some complex constants $c_0, c_1, \ldots, c_r$. Since (38) is, again, a ratio of Hermitian forms in the parameters $c_0, c_1, \ldots, c_r$, we solve an eigenproblem of size $r+1$ to compute a vector

$$a_{(2)} = \tilde{\tilde{c}}_0 a_{(1)} + \tilde{\tilde{c}}_1 e_{r+1} + \ldots + \tilde{\tilde{c}}_r e_{2r}, \tag{40}$$

for which the maximum (38) is attained. This completes the second step. Continuing in this fashion defines the group coordinate relaxation algorithm.

We see that this algorithm cycles through the entire array using subarrays of size r. This is because the basis vectors $\{e_k\}$ are defined to cycle regularly through the vectors $\{e_1, e_2, \ldots, e_n\}$. Also, if r does not divide n evenly, each individual element belongs to a number of different subarrays as the computation proceeds. In other words, if r does not divide n, the entire array is not subdivided into disjoint subarrays.

The group coordinate relaxation algorithm generates a sequence of vectors $a_{(0)}, a_{(1)}, a_{(2)}, \ldots$ that converges to an eigenvector corresponding to the largest eigenvalue of (24). Convergence is assured regardless of the starting vector, with some highly unlikely exceptions. These exceptions are easy to state. If any of the computed vectors $\{a_{(0)}, a_{(1)}, a_{(2)}, \ldots\}$ is precisely an eigenvector of (24) that corresponds to an eigenvalue which is not the largest eigenvalue of the equation, the group coordinate relaxation method will not move from this eigenvector. Numerical roundoff error probably will prevent this in practice. For further discussion and for a convergence theorem whose proof can be extended to the present situation, see Faddeev and Faddeeva.[8] For possible applications of these mathematical methods to other problems, see Lee.[11]

An important feature is that the last computed vector, $a_{(k)}$, gives a larger DIBCF than the previous vector, $a_{(k-1)}$. This is easy to see by observing the ratios (35) and (38).

Another very useful observation is that the algorithm requires knowledge only of $a_{(k)}$ to compute $a_{(k+1)}$. This means that if computation must be interrupted for any reason, it is necessary to store only the last computed vector in order to restart computations.

It is now easy to see how to solve the problem mentioned in the introduction, namely, how to excite

(drive) new elements being added to an existing array without changing the excitations of any of the original array elements. Let N_S be the number of elements in the existing array, and let N_A be the number of elements to be added to this array. Now, number the $n = N_S + N_A$ elements in the full array so that the new elements are numbered $1, 2, \ldots, N_A$, and the elements of the original array are numbered $N_A + 1, N_A + 2, \ldots, N_A + N_S$. The solution of this problem is to perform precisely one iteration of the group coordinate relaxation algorithm with the number of elements relaxed equal to N_A. In other words, set $r = N_A$ in (36) and compute that x in $\mathcal{Q}_0$ for which the maximum in (35) is attained. The required excitations for the additional elements are given explicitly by $a_k = \bar{c}_k / \bar{c}_0, k = 1, \ldots, N_A$, where we have used the notation of (37).

We conclude this section by an examination of the maximum (35). Everything that is said of (35) is easily translated to the maximum (38), as well as all the other maxima required in the group coordinate relaxation algorithm. Note, first, that putting (36) into (35) gives the identity

$$\max_{x \in Q_0} \frac{\bar{x}^T M x}{\bar{x}^T S x} = \max_{z \neq 0} \frac{\bar{z}^T G z}{\bar{z}^T B z}, \tag{41}$$

where $z = \langle c_0, c_1, \ldots, c_r \rangle^T$, and $G = [g_{kj}]$ and $B = [b_{kj}]$ are $(r+1) \times (r+1)$ Hermitian matrices whose general entries are given by

$$\begin{aligned} g_{00} &= \bar{a}_{(0)}^T M a_{(0)}, \\ g_{0k} &= \bar{g}_{k0} = \bar{a}_{(0)}^T M e_k, \quad k = 1, \ldots, r, \\ g_{kj} &= \bar{e}_k^T M e_j, \quad k, j = 1, \ldots, r, \end{aligned} \tag{42}$$

and

$$\begin{aligned} b_{00} &= \bar{a}_{(0)}^T S a_{(0)}, \\ b_{0k} &= \bar{b}_{k0} = \bar{a}_{(0)}^T S e_k, \quad k = 1, \ldots, r, \\ b_{kj} &= \bar{e}_k^T S e_j, \quad k, j = 1, \ldots, r. \end{aligned} \tag{43}$$

Thus, the entries of G and B are computable from the Hermitian forms of M and S, respectively. Let $V_0(\theta, \phi)$ be the field pattern of the entire array for the excitations $a_{(0)}$. Then, we have, explicitly,

$$g_{00} = \sum_{(\theta,\phi) \in \mathcal{F}_p \cap \mathcal{M}} N(\theta, \phi) |V_0^2(\theta, \phi)|, \tag{44}$$

$$g_{k0} = \bar{g}_{0k} = \sum_{(\theta,\phi) \in \mathcal{F}_p \cap \mathcal{M}} N(\theta, \phi) V_0(\theta, \phi) \overline{R_k(\theta, \phi)} \times \exp[-(2\pi i/\lambda) d_k(\theta, \phi)], \quad k = 1, \ldots, r, \tag{45}$$

$$g_{kj} = m_{kj}, \quad k, j = 1, \ldots, r, \tag{46}$$

where m_{kj} is given by (25), and similarly,

$$b_{00} = \sum_{(\theta,\phi) \in \mathcal{F}_p \cap (\mathcal{M} \cup \mathcal{S})} N(\theta, \phi) |V_0^2(\theta, \phi)|, \tag{47}$$

$$b_{k0} = \bar{b}_{0k} = \sum_{(\theta,\phi) \in \mathcal{F}_p \cap (\mathcal{M} \cup \mathcal{S})} N(\theta, \phi) \times V_0(\theta, \phi) \overline{R_k(\theta, \phi)} \exp[-(2\pi i/\lambda) d_k(\theta, \phi)], \quad k = 1, \ldots, r, \tag{48}$$

$$b_{kj} = s_{kj}, \quad k, j = 1, \ldots, r \tag{49}$$

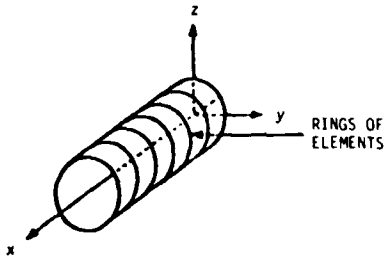


FIG. 4. Arrangement of elements in example 1.

where s_{kj} is given by (26). Because $V_0(\theta, \phi)$ can be computed easily for each (θ, ϕ), we see that (44) through (49) can be computed efficiently in terms of time and core-storage requirements. Now, by using Theorem 1, we see that the maximum of

$$\bar{z}^T G z / \bar{z}^T B z, \tag{50}$$

is achieved by any vector

$$\bar{z} = \langle \bar{c}_0, \bar{c}_1, \ldots, \bar{c}_r \rangle^T \tag{51}$$

which is an eigenvector of the largest eigenvalue of $Gz = \mu B z$. Thus, from (41), we see that

$$a_{(1)} = \bar{c}_0 a_{(0)} + \bar{c}_1 e_1 + \ldots + \bar{c}_r e_r, \tag{52}$$

is a vector for which the maximum (35) is attained.

III. EXAMPLES

A. Example 1: A 105 element cylindrical array

This example illustrates the use of subarrays (i.e., the group coordinate relaxation method) for computing optimum DIBCF with limited computer storage. We select an array with 105 elements arranged around a cylinder. Specifically, we first construct 7 rings of 15 elements each and then place the axis of each of these rings along the x axis (see Fig. 4). The exact positions (and element numbers) are given in Table I, where the units of length are such that the wavelength $\lambda = 1$.

Each element of this array has a hemispherical field pattern defined in the following manner. We conceive of the array as being supported by a (transparent) cylinder. Through each element, we pass a tangent plane parallel to the cylinder axis. The field pattern of an element has unit response on the side of the plane that does not contain the cylinder and has zero response on the side that does contain the cylinder. We assume that the ambient noise field is flat. Also, we choose $p = 32$ in the definition of the Fuller points $\mathcal{F}_p$.

The mainlobe region, $\mathcal{M}$, is defined as a half cone lying above the positive x axis. Specifically, consider the solid cone with axis lying along the positive x axis, with its vertex at the origin, and with a vertex angle of 40°. The xy plane slices this cone into two equal parts, and the mainlobe region, $\mathcal{M}$, is defined to be that part of the cone that lies above the xy plane (i.e., points having positive z coordinates). The sidelobe region, $\mathcal{S}$, is defined to be the set of all directions that are not in the mainlobe region, $\mathcal{M}$. There is no

TABLE I. Coordinates of elements in example 1.

Element no.	Coordinates x	y	z
1	0.0000	0.7642	0.0000
2	0.3337	0.7642	0.0000
3	0.6674	0.7642	0.0000
4	1.0011	0.7642	0.0000
5	1.3348	0.7642	0.0000
6	1.6685	0.7642	0.0000
7	2.0022	0.7642	0.0000
8–14	As above	0.6982	0.3108
15–21		0.5114	0.5679
22–28		0.2362	0.7268
29–35		−0.0799	0.7600
36–42		−0.3821	0.6618
43–49		−0.6183	0.4492
50–56		−0.7475	0.1589
57–63		−0.7475	−0.1589
64–70		−0.6183	−0.4492
71–77		−0.3821	−0.6618
78–84		−0.0799	−0.7600
85–91		0.2362	−0.7268
92–98		0.5114	−0.5679
99–105		0.6982	−0.3108

ignored region, $\mathcal{S}$, in this example.

With the above choices, the DIBCF array problem is completely specified. In this case, we use subarrays to optimize the full array because the direct method of optimization requires more core storage on the Univac 1108 than is available. (If the Univac 1108 had virtual memory, the use of subarrays would not be required. On the other hand, one might still use subarrays on a machine with virtual memory for a variety of other reasons.) It seems best to use as many array elements as can be handled easily in the available computer storage, so in this case we choose 69 elements, i.e., roughly two-thirds of the full array. By using the program in Streit,[3] we require only 45 000 words of main memory.

The group coordinate relaxation scheme required roughly 1650 s per iteration, and 5 iterations in all. Thus, total computation time was roughly 2.25 h. Table II gives the final (optimal) set of element excitations. The vertical field pattern is given in Fig. 5, and Fig. 6 gives the horizontal field pattern for these excitations. We point out that the field patterns in these two figures have abrupt jumps because the individual element field

TABLE II. Optimum excitations for example 1.

Element no.	Magnitude	Phase	Element no.	Magnitude	Phase	Element no.	Magnitude	Phase
1	0.014 97	−2.041 50	36	0.001 97	1.820 93	71	0.036 56	2.707 18
2	0.044 56	1.554 78	37	0.003 07	−1.479 02	72	0.085 37	−0.055 12
3	0.076 01	−1.273 18	38	0.008 57	1.588 49	73	0.129 45	−2.858 12
4	0.091 75	2.152 64	39	0.014 55	−1.390 23	74	0.142 65	0.617 88
5	0.082 06	−0.724 47	40	0.016 16	1.955 39	75	0.117 32	−2.178 31
6	0.052 16	2.686 48	41	0.012 82	−0.942 79	76	0.069 27	1.340 10
7	0.021 26	−0.153 55	42	0.006 53	2.490 85	77	0.025 74	−1.306 99
8	0.011 60	−0.728 50	43	0.008 11	0.104 99	78	0.070 23	2.827 35
9	0.035 88	2.552 05	44	0.023 91	−2.953 28	79	0.188 62	0.017 20
10	0.061 33	−0.396 98	45	0.041 58	0.424 16	80	0.308 92	−2.834 28
11	0.073 42	2.934 76	46	0.050 11	−2.473 88	81	0.357 55	0.590 43
12	0.064 03	0.000 79	47	0.044 22	0.926 78	82	0.304 68	−2.261 94
13	0.038 93	−2.923 34	48	0.027 16	−1.925 83	83	0.184 37	1.185 12
14	0.013 98	0.469 31	49	0.010 48	1.501 98	84	0.067 83	−1.595 34
15	0.003 48	−0.013 23	50	0.012 66	−1.160 74	85	0.056 00	2.520 14
16	0.012 78	3.131 21	51	0.039 73	2.244 29	86	0.139 59	−0.311 04
17	0.022 96	0.298 55	52	0.067 34	−0.654 77	87	0.219 23	3.115 25
18	0.027 17	−2.506 53	53	0.079 80	2.707 91	88	0.245 87	0.254 34
19	0.023 19	0.987 64	54	0.069 17	−0.208 62	89	0.203 37	−2.595 81
20	0.014 09	−1.732 54	55	0.041 97	−3.121 57	90	0.119 10	0.863 11
21	0.005 70	1.789 55	56	0.015 13	0.260 34	91	0.041 79	−1.865 68
22	0.010 32	1.683 25	57	0.016 76	−2.085 31	92	0.026 26	2.889 25
23	0.026 24	−1.204 92	58	0.052 36	1.523 29	93	0.064 38	0.229 40
24	0.046 16	2.097 37	59	0.090 72	−1.299 56	94	0.102 32	−2.514 17
25	0.057 83	−0.904 29	60	0.109 97	2.134 97	95	0.118 89	1.006 83
26	0.053 88	2.376 97	61	0.097 76	−0.740 37	96	0.103 93	−1.759 55
27	0.036 79	−0.617 24	62	0.061 42	2.676 18	97	0.066 23	1.767 56
28	0.016 19	2.703 56	63	0.025 15	−0.164 89	98	0.026 74	−0.955 27
29	0.032 97	1.326 46	64	0.018 74	−2.678 88	99	0.014 31	−2.548 78
30	0.087 31	−1.599 98	65	0.058 33	0.974 72	100	0.045 66	1.152 51
31	0.146 26	1.741 47	66	0.105 12	−1.807 74	101	0.081 78	−1.633 09
32	0.175 53	−1.207 41	67	0.132 27	1.655 09	102	0.103 00	1.829 50
33	0.157 43	2.124 65	68	0.121 80	−1.182 57	103	0.094 45	−1.016 12
34	0.101 96	−0.819 50	69	0.079 79	2.258 32	104	0.061 24	2.421 74
35	0.040 99	2.524 82	70	0.032 12	−0.589 40	105	0.025 21	−0.418 90

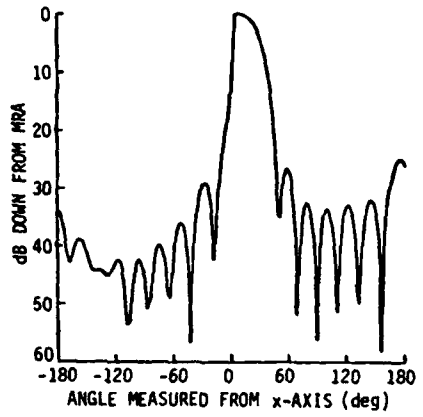


FIG. 5. Vertical field pattern for example 1 with excitations given in Table I.

patterns have sharp jumps, due to their assumed hemispherical shape. (These field patterns were computed by the program described by Lee and Leibiger.[12]) Also, we point out that the geometry of the array and of the mainlobe region, $\mathfrak{M}$, implies that the optimum field pattern be symmetric about endfire in the horizontal plane. That is, in Fig. 6, the field pattern should be symmetric about 0°. The fact that it is not is due entirely to ending the computations after the fifth iteration. Further iterations, presumably, would yield increasingly symmetric horizontal field patterns.

This method creates a steadily increasing sequence of estimates for the largest eigenvalue. Since there were five iterations, there were five estimates and these are given in Table III. Based on this table and on the field patterns of Figs. 5 and 6, it would seem that additional iterations of the algorithm would be only marginally worthwhile. In other words, to all intents and purposes, the array excitations have been optimized successfully.

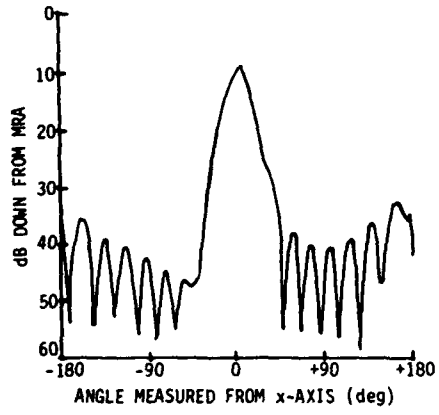


FIG. 6. Horizontal field pattern for example 1 with excitations given in Table II.

TABLE III. Group coordinate relaxation estimates of largest eigenvalue for example 1.

Iteration no.	Estimate of largest eigenvalue
1	0.914 27
2	0.961 00
3	0.963 74
4	0.965 32
5	0.965 59

B. Example 2: A comparison with Dolph-Chebyshev design

This example serves two purposes. First, it provides a comparison with the Dolph–Chebyshev line array design. Second, it gives some insight into the nature of the lower order eigenvalues/eigenvectors.

Suppose we have a line array of 15 elements that lies along the y axis (see Fig. 1) with equal spacings of 0.5 wavelength, where the wavelength $\lambda = 1$. The units of length are irrelevant. Thus, if the first element lies at the origin with coordinates (0., 0., 0.), the 15th element has the coordinates (0., 7., 0.). It is well known that any line array has a field pattern with cylindrical symmetry about the array axis. Therefore, we define $\mathfrak{M}$ to be the set of all directions that lie within 8° of a normal to the y axis, and we define $\mathcal{S}$ to be the collection of all other directions. Hence, $\mathfrak{M}$ is a 16° wide annulus and both $\mathfrak{M}$ and $\mathcal{S}$ are cylindrically symmetric. The ambient noise field is assumed to be flat, and the individual elements are assumed to be omnidirectional. Finally, considering the construction of the Fuller points, $\mathfrak{F}_p$, we choose $p = 24$.

The above data completely define the DIBCF array problem. In Streit,[3] a listing of the entire computer program required for exactly this example is given. The results of the execution are given in Table IV. Computation time on the Univac 1108 (under EXEC 8) was about 41 s. The field pattern in the xy plane is given in Fig. 7.

TABLE IV. Excitations for 15-element equispaced line array: Dolph–Chebyshev versus DIBCF.

Element no.	Dolph–Chebyshev	DIBCF ($p = 24$)
1	0.343 71	0.256 87
2	0.357 75	0.395 20
3	0.504 03	0.540 24
4	0.653 38	0.682 90
5	0.791 08	0.812 42
6	0.902 42	0.911 88
7	0.974 87	0.979 20
8	1.000 00	1.000 00
9	0.974 87	0.979 20
10	0.902 42	0.911 88
11	0.791 08	0.812 42
12	0.653 38	0.682 90
13	0.504 03	0.540 24
14	0.357 75	0.395 20
15	0.343 71	0.256 87

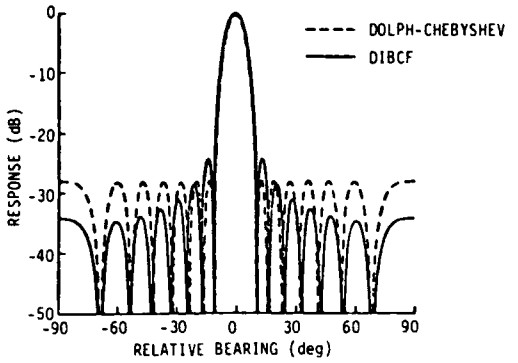


FIG. 7. Field patterns for excitations in Table IV.

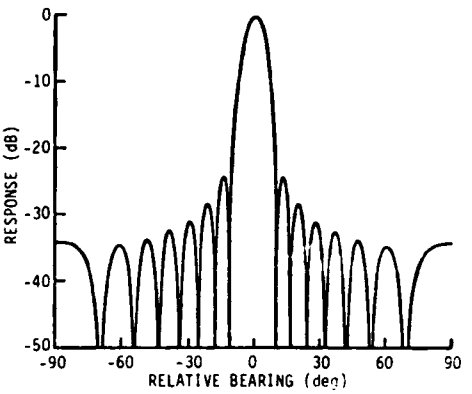


FIG. 8. Field pattern for eigenvector 1 of example 2.

The Dolph–Chebyshev excitations are designed exclusively for half-wavelength equispaced line arrays with omnidirectional elements. For a given number of elements, the Dolph–Chebyshev excitations depend only on the steered direction and on the specified sidelobe level. For a broadside (i.e., steered normal to the line of the array) 15 element array, the Dolph–Chebyshev excitations for a 28 dB sidelobe level field pattern are given in Table IV. The corresponding field pattern is shown in Fig. 7.

We note that the mainlobe shape of the Dolph–Chebyshev array and the DIBCF array are indistinguishable. The only difference lies in sidelobe structure. We see that by sacrificing approximately 3 dB in the sidelobe nearest the mainlobe, all the remaining sidelobes can be made smaller than the overall 28 dB sidelobe level of the Dolph–Chebyshev array.

What about the lower order eigenvalues? The first four eigenvalues/eigenvectors are listed in Table V. (Note that the eigenvector of μ_1 in Table V is the same as DIBCF in Table IV, but is normalized differently.) Also, the corresponding field patterns are given in Figs. 8–11. We remark only that the field pattern for the largest eigenvalue μ_1 has no nulls in the mainlobe region, $\mathfrak{M}$, whereas the field pattern for μ_2 has one null in $\mathfrak{M}$, two nulls for μ_3, and three nulls for μ_4.

TABLE V. The four largest eigenvalues/eigenvectors of example 2.

Element no.	$\mu_1 = 0.9894$	$\mu_2 = 0.8206$	$\mu_3 = 0.3231$	$\mu_4 = 0.0383$
1	0.0916	0.2755	0.4486	0.5106
2	0.1409	0.3158	0.3666	0.2141
3	0.1927	0.3289	0.2442	−0.0356
4	0.2436	0.3112	0.1066	−0.2042
5	0.2898	0.2675	−0.0242	−0.2664
6	0.3252	0.1929	−0.1413	−0.2454
7	0.3492	0.1030	−0.2097	−0.1391
8	0.3567	0.0000	−0.2403	0.0000
9	0.3492	−0.1030	−0.2097	0.1391
10	0.3252	−0.1929	−0.1413	0.2454
11	0.2898	−0.2675	−0.0242	0.2664
12	0.2436	−0.3112	0.1066	0.2042
13	0.1927	−0.3289	0.2442	0.0356
14	0.1409	−0.3158	0.3666	−0.2141
15	0.0916	−0.2755	0.4486	−0.5106

C. Example 3: Effects of sampling

The first two examples did not mention the effects of sampling on the field patterns. Specifically, the parameter p in the definition of the Fuller points, $\mathfrak{F}_p$, determines how finely we have sampled all spatial directions. Hence, the parameter p influences the resulting field patterns. In particular, if p is not sufficiently large it is possible for the optimal DIBCF field pattern to have a split beam.

We illustrate this effect by systematically varying p in the array of example 2, but for a different choice of $\mathfrak{M}$ and $\mathcal{S}$. Here, we define $\mathfrak{M}$ to be the collection of all directions whose projection on the xz plane lies within $\pm 8°$ of the z axis. Specifically, the direction corresponding to direction cosines (α,β,γ) lies in $\mathfrak{M}$ only if $|\alpha/(\alpha^2+\gamma^2)^{1/2}| \leq \sin 8°$. In other words, $\mathfrak{M}$ consists of all directions contained between the two planes intersecting the yz plane at the angles of $+8°$ and $-8°$. The sidelobe region, $\mathcal{S}$, consists of all remaining directions, so there is no ignored region, $\mathcal{I}$. Optimal excitations for several choices of p are given in Table VI. The field patterns for $p = 24$ and $p = 16$ are given in Fig. 12.

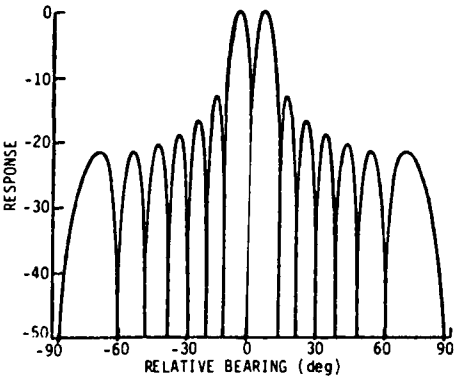


FIG. 9. Field pattern for eigenvector 2 of example 2.

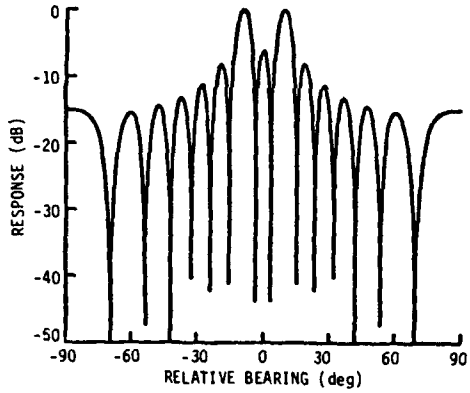


FIG. 10. Field pattern for eigenvector 3 of example 2.

We do not present the field patterns for $p = 32$ and $p = 40$, because they are so similar to $p = 24$.

Table VI also shows the effects of oversampling. Note that the optimal excitations for $p \geq 24$ are all similar, but they do not seem to be converging to an optimal set. This is probably due to the buildup of numerical roundoff error in the required sums [i.e., (25) and (26)], but it could also be that p must be chosen even larger than 40 before the optimal excitations give the appearance of convergence. In any event, the importance of sampling sufficiently finely is clear, but evidently oversampling wastes time and increases the numerical roundoff error in the computed optimal excitations.

D. Example 4: Time and accuracy in the indirect method

It is clear from the definition of the indirect, or group coordinate relaxation method, that the size of the subarrays used and the stopping criteria for the iteration procedure both have significant effects on numerical accuracy of the computed excitations and on the time required to compute them. The following example illustrates how numerical accuracy and computation time depend on both these parameters.

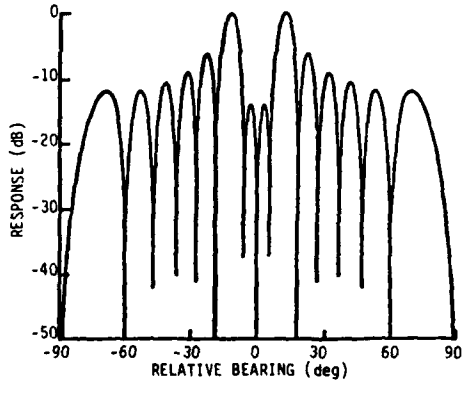


FIG. 11. Field pattern for eigenvector 4 of example 2.

TABLE VI. Effects of sampling on excitations for example 3.

Element no.	p = 16	24	32	40
1	0.2953	0.1275	0.1322	0.1177
2	−0.0064	0.1676	0.1792	0.1662
3	0.3846	0.2207	0.2406	0.2232
4	−0.1045	0.2497	0.2613	0.2581
5	0.2990	0.2884	0.2946	0.2934
6	−0.2830	0.3067	0.2991	0.3099
7	0.1753	0.3337	0.3154	0.3259
8	−0.3280	0.3346	0.3117	0.3279
9	0.1753	0.3337	0.3154	0.3259
10	−0.2830	0.3067	0.2991	0.3099
11	0.2990	0.2884	0.2946	0.2934
12	−0.1045	0.2497	0.2613	0.2581
13	0.3846	0.2207	0.2406	0.2232
14	−0.0064	0.1676	0.1792	0.1662
15	0.2953	0.1275	0.1322	0.1177
Largest eigenvalue, μ_1	0.1149	0.1243	0.1165	0.1225
No. of Fuller points $\mathfrak{F}_p$	2562	5762	10242	16002

We consider a line array of 25 elements that lies along the y axis with equal spacings of 0.5 wavelength, where the wavelength $\lambda = 1$. Thus, the coordinates of the first and last elements are (0.,0.,0.) and (0.,12.,0.), respectively. We select the mainlobe region, $\mathfrak{M}$, to be the set of all directions that lie within 5° of a normal to the y axis, and we define $\mathcal{S}$ to be the set of all other directions. There is no ignored region, $\mathcal{I}$. The ambient noise field is flat and the individual elements are assumed omnidirectional. Finally, we select the Fuller points, $\mathfrak{F}_{28}$. This completely defines our problem.

Table VII shows the number of iterations required for various choices of subarray size, n_s, and stopping criterion, EPSI, defined by

$$\left|1 - \frac{\text{old eigenvalue estimate}}{\text{new eigenvalue estimate}}\right| \leq \text{EPSI}.$$

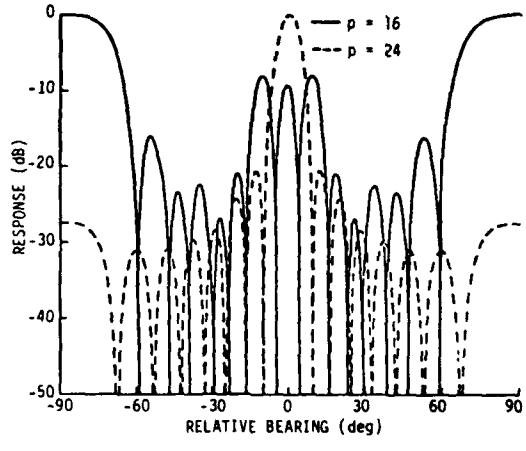


FIG. 12. Field pattern for example 3.

TABLE VII. Number of iterations required in example 4.

n_s	EPSI 10^{-3}	10^{-4}	10^{-5}	Time per iteration (s)
5	9	13	24	83
10	5	10	13	95
15	3	4	6	115
20	3	3	3	140
25	1	1	1	172

As can be expected, the number of iterations required increases with decreasing ESPI and decreases with increasing n_s. Also the computation time per iteration increases with n_s.

An important concern is the numerical accuracy of the computed excitations. This is particularly important in light of the fact that numerical computation of eigenvectors by any method is less stable than the numerical computation of eigenvalues. Table VIII shows the results obtained for $n_s = 5$ by stopping after the first four complete passes through the array, i.e., for iterations 5, 10, 15, and 20, respectively. The exact results are included, also. The field patterns corresponding to excitations of iteration 5 and the exact excitations are shown in Fig. 13. Note that, at the end of iteration 5, the field pattern already possesses sidelobes in the correct positions although they are about 3 dB higher than in the field pattern of the exact excitations. Thus, the effect of later iterations is to beat down the sidelobes while maintaining the mainlobe beamwidth.

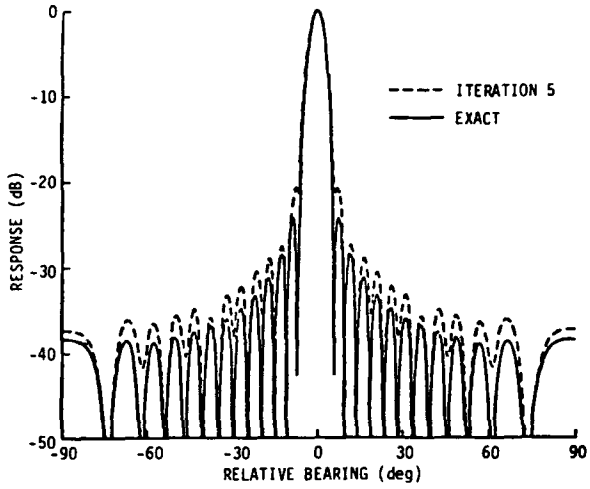


FIG. 13. Comparison in example 4, exact versus iteration 5.

IV. SUMMARY

The concept of Directivity Index with Beamwidth Control (DIBC) has been defined as the ratio of power in the mainlobe region to the total power in both the mainlobe and the sidelobe regions. A mathematically and numerically tractable method for the computation of optimum element excitations (i.e., excitations that maximize DIBC) is presented. A technique known as group coordinate relaxation is shown to be an effective

TABLE VIII. Example 4 with subarrays of five elements.

Element no.	Iteration 5	Iteration 10	Iteration 15	Iteration 20	Exact
1	0.632	0.410	0.623	0.793	1.000
2	0.813	0.622	0.887	1.090	1.339
3	0.993	0.860	1.173	1.406	1.692
4	1.183	1.117	1.428	1.736	2.057
5	1.394	1.398	1.800	2.082	2.436
6	1.446	1.788	2.218	2.461	2.792
7	1.648	2.079	2.539	2.795	3.145
8	1.840	2.355	2.832	3.095	3.456
9	2.023	2.611	3.100	3.366	3.732
10	2.188	2.842	3.328	3.592	3.955
11	2.541	3.143	3.522	3.783	4.120
12	2.664	3.298	3.654	3.904	4.223
13	2.749	3.392	3.722	3.956	4.254
14	2.806	3.439	3.737	3.951	4.223
15	2.809	3.421	3.677	3.877	4.120
16	3.011	3.271	3.559	3.729	3.955
17	2.951	3.145	3.396	3.541	3.732
18	2.834	2.967	3.174	3.298	3.456
19	2.696	2.756	2.929	3.022	3.145
20	2.489	2.494	2.631	2.700	2.792
21	2.012	2.190	2.290	2.353	2.436
22	1.762	1.887	1.957	2.000	2.057
23	1.511	1.567	1.632	1.658	1.692
24	1.260	1.294	1.313	1.324	1.339
25	1.000	1.000	1.000	1.000	1.000
μ_{max}	0.979 831 3	0.985 735 7	0.988 001 9	0.988 699 8	0.988 989 0

means of computing optimum element excitations for arrays of arbitrary numbers of elements, yet it requires only nominal core storage. Conceptually, the group coordinate relaxation technique employs subarrays of the full array in a systematic manner to optimize excitations of the full array. Four examples have been included, one of which demonstrates the effectiveness of group coordinate relaxation for a cylindrical array of 105 elements.

ACKNOWLEDGMENTS

The author would like to thank Mr. Barry G. Buehler and Dr. Albert H. Nuttall, both of the New London Laboratory, Naval Underwater Systems Center, for their helpful comments on the various drafts of this article. Mr. Buehler also supplied the author with many examples using the approach of this article, one of which is included here as example 1.

[1]J. K. Butler and H. Unz, "Beam efficiency and gain optimization of antenna arrays with nonuniform spacings," Radio Sci. 2 (7) (new series), 711–720 (July 1967).

[2]J. K. Butler and H. Unz, "Optimization of beam efficiency and synthesis of nonuniformly spaced arrays," Proc. IEEE (Letters) **54**, 2007–2008 (December 1966).

[3]R. L. Streit, *Array Optimization Using Subarrays*, NUSC Technical Report 5889 (Naval Underwater Systems Center, New London, CT, 23 March 1979).

[4]F. R. Gantmacher, *The Theory of Matrices* (Chelsea, New York, 1960).

[5]*The IMSL Library, Volume 2*, International Mathematical and Statistical Libraries (IMSL, Inc., Houston, TX, 1977), 6th ed.

[6]R. S. Martin and J. H. Wilkinson, "Reduction of the symmetric eigenproblem $Ax = \lambda Bx$ and related problems to standard form," Numerische Mathematik **11**, 99–110 (1968).

[7]D. Slepian and H. O. Pollak, "Prolate spheroidal wave functions, Fourier analysis and uncertainty— I," Bell Syst. Tech. J. **40**, 43–63 (1961).

[8]D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra* (Freeman, San Francisco, 1963).

[9]T. W. Melnyk, O. Knop, and W. R. Smith, "External arrangements of points and unit charges on a sphere: Equilibrium configurations revisited," Can. J. Chem. **55**, 1745–1761 (1977).

[10]J. Prenis, "An Introduction to Domes," in *The Dome Builder's Handbook* (Running Press, Philadelphia, 1973).

[11]D. Lee, *Maximization of Reverberation Index*, NUSC Technical Report 5375 (Naval Underwater Systems Center, New London, CT, 2 October 1976).

[12]D. Lee and G. A. Leibiger, *Computation of Beam Patterns and Directivity Indices for Three-Dimensional Arrays with Arbitrary Element Spacings*, NUSC Technical Report 4687 (Naval Underwater Systems Center, New London, CT, 22 February 1974).

The Effect Of Interchannel Crosstalk On Array Performance

R. L. Streit

The effect of interchannel crosstalk on array performance

Roy L. Streit
Naval Underwater Systems Center, New London, Connecticut 06320

(Received 5 May 1987; accepted for publication 16 July 1989)

It is shown that interchannel crosstalk can always be eliminated before the channel signals enter the beamformer, provided crosstalk levels do not exceed a maximum permissible upper bound and are known exactly. The crosstalk upper bound (in decibels) is shown to be $-20 \log N$, where N is the number of channels in the array. The beam pattern of a general array with arbitrary crosstalk levels, steered in any direction, is derived. Sample beam patterns are presented for arrays of 50 and 100 elements. Expected, or average, beam patterns are derived for interchannel crosstalk coefficients modeled as statistically independent random variables. It is shown that pointing error can occur when crosstalk has nonzero mean. It is also shown how to correct for nonzero mean crosstalk before the signals enter the beamformer, provided these means are known.

PACS numbers: 43.60.Gk, 43.30.Yj, 43.88.Hz

INTRODUCTION

Interchannel crosstalk is modeled throughout this paper as a multiple-input–multiple-output linear system. The inputs to this linear system are the array's sensor outputs, while the system outputs are the inputs to the array beamformer. In Secs. I and II it is assumed that the transfer function matrix H of the linear crosstalk system is known exactly. The discussion in these two sections is simplified by presenting explicitly only the idealized case when crosstalk is both instantaneous and frequency independent; however, this simplified presentation in no way is a restriction on the methods and results presented in these sections. A very different crosstalk model is assumed for Sec. III. In this section the components of the crosstalk transfer function matrix H are treated as random variables. Such a model is employed not to suggest that crosstalk is truly a random system, but rather to gain some rudimentary insight into the consequences of the lack of exact knowledge of the crosstalk transfer function matrix H. Whether or not such a model is satisfactory for the intended purpose depends on the particular application.

Section I of this paper shows that if the interchannel crosstalk levels do not exceed $\mathrm{XB} = -20 \log N$ (in decibels), then crosstalk can always be eliminated before the channel signals enter the beamformer. The crosstalk bound XB should be interpreted as a theoretical worst case upper bound on crosstalk levels. This bound is especially important if adaptive beamforming is undertaken; that is, given statistically independent array sensor outputs, statistical independence of the beamformer input signals can be guaranteed if the crosstalk bound XB is satisfied.

Section II of this paper shows that, for a general array, crosstalk is theoretically equivalent to a channel shading perturbation. An explicit expression for the beam pattern of a general array, arbitrarily steered, with crosstalk is given. As will be seen, satisfying the crosstalk bound XB does not necessarily mean that sidelobe levels are not degraded. In addition, crosstalk almost always causes pointing error; that is, the maximum response of the beam pattern need not occur in the steered direction. Pointing error thus contributes to target bearing estimation error. Examples indicate, however, that pointing error is probably not significant if the bound XB is satisfied.

Section III of this paper derives the expected, or average, beam pattern of an array with the individual crosstalk coefficients (i.e., the components of the transfer function matrix H) modeled as statistically independent random variables. It is shown that pointing error cannot occur in the expected beam pattern, provided the crosstalk between distinct pairs of channels is zero mean. Pointing error can occur only when crosstalk is nonzero mean. It is shown how to correct for nonzero mean crosstalk before the signals enter the beamformer.

Section IV gives several example beam patterns, and Sec. V briefly recapitulates the paper's conclusions. Appendix A states Gershgorin's theorem that is used to derive the crosstalk bound XB. Finally, using the crosstalk model of Sec. III, Appendix B derives the maximum crosstalk variance allowed for a specified increase in the beam pattern sidelobe level.

I. CROSSTALK BOUND

Let $V_n(t)$ denote the output voltage signal of the nth sensor of an N channel array. Let $U_n(t)$ denote this signal, contaminated by crosstalk, at the input to the beamformer. Ideally, $U_n(t) = V_n(t)$ when the effect of crosstalk between channels is negligible. If crosstalk cannot be ignored, then $\{U_n(t)\}$ is related to the voltage signals $V_1(t),\ldots,V_N(t)$ by the linear relationships

$$U_n(t) = V_n(t) + \sum_{\substack{k=1 \\ k \neq n}}^{N} H_{nk} V_k(t), \quad \text{for } 1 \leq n \leq N, \quad (1)$$

where $\{H_{nk}\}$ are real constants, independent of both time t and the signals $\{V_n(t)\}$. For convenience, we define $H_{kk} = 1$ for all k. Then, for all n and k, $H_{nk} V_k(t)$ is the contribution to the nth beamformer input from the k th sen-

sor output. The units of the beamformer inputs are not stated in (1). It is convenient to suppose that $U_n(t)$ is a voltage, in which case the coefficients $\{H_{nk}\}$ are dimensionless.

Model (1) assumes that no time delay exists between the sensor outputs and their communication to the beamformer. This assumption is very reasonable in practice, and it has the important consequence that the crosstalk coefficients $\{H_{nk}\}$ are all real. If some application requires nonzero time delays, then it is more appropriate to develop a model in the frequency domain. In that case, the coefficients $\{H_{nk}\}$ are complex and, possibly, frequency dependent. None of the results developed in this section depend on $\{H_{nk}\}$ being real constants; hence, they apply on a frequency-by-frequency basis in the frequency domain when crosstalk is modeled as a multiple-input–multiple-output linear system.

Model (1) does not require that the system which communicates the sensor outputs to the beamformer conserve power. It also does not require reciprocity; that is, H_{nk} need not equal H_{kn}. However, the model does assume that the sensors are properly calibrated and have the same gain.

Complete crosstalk is defined to be the special case where the output voltage of each sensor makes equal contributions to every beamformer input channel. Thus, when complete crosstalk occurs, all the coefficients H_{nk} are equal to 1. From (1) it follows that

$$U_n(t) = \sum_{k=1}^{N} V_k(t), \quad \text{all } n.$$

In other words, regardless of the nature of the sensor outputs, all beamformer input channels are identical. Complete crosstalk is highly undesirable.

In matrix form, system (1) can be written

$$U(t) = HV(t), \tag{2}$$

where

$$U(t) = \begin{bmatrix} U_1(t) \\ U_2(t) \\ \vdots \\ U_N(t) \end{bmatrix}, \quad V(t) = \begin{bmatrix} V_1(t) \\ V_2(t) \\ \vdots \\ V_N(t) \end{bmatrix},$$

$$H = \begin{bmatrix} 1 & H_{12} & \cdots & H_{1N} \\ H_{21} & 1 & \cdots & H_{2N} \\ \vdots & \vdots & & \vdots \\ H_{N1} & H_{N2} & \cdots & 1 \end{bmatrix}.$$

If the matrix H is invertible, then we have

$$V(t) = H^{-1}U(t). \tag{3}$$

Therefore, crosstalk can be eliminated before the signals enter the beamformer by using (3) to recover the sensor output voltages if H is known and the sensors are properly calibrated. The requirement that H^{-1} exists is a critical assumption. In the case of complete crosstalk, for example, the entries of H are identically 1, and so H^{-1} does not exist. It is important to emphasize that eliminating crosstalk using (3) requires a thorough understanding of the crosstalk mechanism because every entry of the matrix H must be known with considerable accuracy to evaluate the inverse of H accurately.

In practice, efforts are usually made to minimize crosstalk by appropriate engineering means. We therefore assume that the level of crosstalk between any pair of channels is bounded; that is,

$$|H_{nk}| \leq \epsilon, \quad \text{all } k \neq n, \tag{4}$$

for some constant ϵ. As the complete crosstalk example shows, linear independence of the beamformer inputs, given linearly independent sensor outputs, is a very important consideration to keep in mind (especially in adaptive beamforming) when specifying the size of ϵ. We now derive a theoretical upper bound, denoted XB, which guarantees linear independence of the beamformer inputs whenever crosstalk levels fall below XB. Equivalently, we derive a crosstalk level XB, which guarantees that the inverse of H exists no matter what the actual crosstalk coefficients H_{nk} are, so long as they are not larger in magnitude than XB.

Gershgorin's theorem applied to H (see Appendix A) together with the inequalities (4) imply that all the eigenvalues λ of H satisfy the inequality

$$|\lambda - 1| \leq (N-1)\epsilon. \tag{5}$$

Since H is invertible if and only if H has no zero eigenvalues, it follows that H is certainly invertible if

$$(N-1)\epsilon < 1,$$

or

$$\epsilon < \epsilon_{max} = 1/(N-1). \tag{6}$$

This is a sufficient, but not necessary, condition for the existence of H^{-1}. Inequality (6) is the crosstalk upper bound mentioned above. The beamformer inputs $\{U_n(t)\}$ are linearly independent if and only if the matrix H is invertible (and, of course, the sensor outputs $\{V_n(t)\}$ are linearly independent). Taking $20\log(\epsilon_{max})$ gives $-20\log(N-1) \doteq -20\log N = \text{XB}$ as the maximum allowed crosstalk level (in decibels) between any two channels.

In some applications, the crosstalk matrix H may have special structure, e.g., H may be banded, or block diagonal, or sparse, etc. If such structure is present, it may well mean that a less restrictive crosstalk bound than the bound XB derived above is appropriate for the given array. Incorporation of any such special structure for H into the above derivation of XB is straightforward because of the generality of Gershgorin's theorem; however, we do not pursue this issue further here because of the many different possible structures for H.

Define $E = H - I$, where I is the identify matrix. Thus E has a zero diagonal, but is otherwise identical to H. Gershgorin's theorem and assumptions (4) and (6) guarantee that all the eigenvalues of E lie inside the unit circle in the complex plane. Since $H = I + E$ and the eigenvalues of E are less than 1 in magnitude, we can write

$$H^{-1} = (I+E)^{-1} = \sum_{n=0}^{\infty} (-1)^n E^n$$

$$= I - E + E^2 - E^3 + \cdots \tag{7}$$

$$\doteq I - E. \tag{8}$$

Substituting (8) into (3) gives the potentially very useful approximation

$$V(t) \doteq (I - E)U(t). \tag{9}$$

Eliminating crosstalk using (9) is much more computationally efficient than using (3). The drawback is that (9) is only an approximation, whereas (3) holds exactly.

Using standard results concerning matrices, (9) gives the following estimate for the total squared difference between the true sensor output vector $V(t)$ and the crosstalk contaminated beamformer input vector $U(t)$:

$$\sum_{n=1}^{N} [U_n(t) - V_n(t)]^2 \le \sum_{n=1}^{N} \sum_{\substack{k=1 \\ k \neq n}}^{N} |H_{nk}|^2 \sum_{n=1}^{N} U_n^2(t).$$

This estimate is useful only when most of the crosstalk coefficients are zero.

When is the calculation of H^{-1} numerically accurate? If the computer uses $T > 1$ significant digits, then a sufficient condition for numerical stability is that the "condition number" of H not exceed 10^t. The condition number of H is defined to be the ratio of its largest to its smallest singular value. Assuming that H is diagonalizable (a not very restrictive assumption in this application), then the singular values of H are precisely the eigenvalues of H in absolute value (Ref. 1, 4.12.3). Since the eigenvalues of H satisfy the inequality (5), the singular values lie in the closed interval $[1 - (N-1)\epsilon, 1 + (N-1)\epsilon]$. To ensure numerical stability of H^{-1}, we require

$$\frac{1 + (N-1)\epsilon}{1 - (N-1)\epsilon} < 10^t.$$

Solving for ϵ gives

$$\epsilon < \frac{1}{N-1}\left(\frac{10^t - 1}{10^t + 1}\right) \doteq \frac{1}{N-1} = \epsilon_{max}.$$

This is the same condition (6) which guaranteed the existence of H^{-1}. Thus numerical inversion of H is reliable if $\epsilon < \epsilon_{max}$, that is, if the crosstalk bound XB is satisfied.

It is still necessary, however, to estimate the effect of crosstalk on the beam patterns. It is possible for the crosstalk bound XB to be satisfied and still have poor beam patterns. Satisfying the bound XB does not guarantee satisfactory beam patterns, as will be seen; it only guarantees that H^{-1} exists.

II. BEAM PATTERNS

The beamformer output is the weighted delayed sum of the channel outputs

$$B(t) = \sum_{n=1}^{N} a_n U_n(t - \tau_n),$$

where $\{a_n\}$ are the channel weights, and $\{\tau_n\}$ are time delays corresponding to a particular steered beam direction. Substituting the crosstalk effects (1) gives

$$B(t) = \sum_{n=1}^{N} a_n \sum_{k=1}^{N} H_{nk} V_k(t - \tau_n)$$

$$= \sum_{k=1}^{N} \sum_{n=1}^{N} a_n H_{nk} V_k(t - \tau_n). \tag{10}$$

For a line array steered broadside, $\tau_n = 0$ for all n. For this particular case, the summation on n can be done separately, giving

$$B(t) = \sum_{k=1}^{N} b_k V_k(t), \tag{11}$$

where

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} 1 & H_{21} & \cdots & H_{N1} \\ H_{12} & 1 & \cdots & H_{N2} \\ \vdots & \vdots & & \vdots \\ H_{1N} & H_{2N} & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = H^t a. \tag{12}$$

Thus the broadside beam pattern of a line array with crosstalk is an ordinary beam pattern with weights $\{b_n\}$ which are perturbations of the original weights $\{a_n\}$.

In general, however, the time delays $\{\tau_n\}$ are not zero. Nonetheless, a similar result can be shown to hold. The time delay used for a channel located at position $\mathbf{p}_n = (x_n, y_n, z_n)$, when steered to form a beam in the direction of the unit vector $\mathbf{u}$, is

$$\tau_n = (\mathbf{p}_n \cdot \mathbf{u})/c, \tag{13}$$

where c is the wave propagation velocity. To find the beam pattern of this steered beam, we proceed as follows. Suppose a unit amplitude plane wave of radian frequency ω is propagating from direction $\mathbf{v}$. Note that $\mathbf{v}$ is unrelated to the steered direction $\mathbf{u}$. Then,

$$V_k(t) = \exp[i\omega(t + \mu_k)], \quad \text{all } k, \tag{14}$$

where the time delay μ_k is given by

$$\mu_k = (\mathbf{p}_k \cdot \mathbf{v})/c. \tag{15}$$

The time required for a plane wave to propagate from sensor k to sensor j is therefore $|\mu_k - \mu_j|$. Because of (14), we have

$$V_k(t - \tau_n) = \exp[i\omega(t + \mu_k - \tau_n)]. \tag{16}$$

Substituting (16) into (10) and taking the magnitude squared of both sides gives the squared amplitude of the steered beamformed output:

$$|B(t)|^2 = \left| \sum_{k=1}^{N} \sum_{n=1}^{N} a_n H_{nk} \exp[i\omega(\mu_k - \tau_n)] \right|^2.$$

The right-hand side of this equation is independent of time, but it does depend on the plane-wave direction $\mathbf{v}$ via (15). Because of the assumption of a unit amplitude plane wave, this function is the directional beam pattern, denoted here $F(\mathbf{v})$. Consequently,

$$F(\mathbf{v}) = \left| \sum_{k=1}^{N} \exp(i\omega\mu_k) \sum_{n=1}^{N} a_n H_{nk} \exp(-i\omega\tau_n) \right|^2$$

$$= \left| \sum_{k=1}^{N} c_k \exp(i\omega\mu_k) \right|^2, \tag{17}$$

where

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} 1 & H_{21} & \cdots & H_{N1} \\ H_{12} & 1 & \cdots & H_{N2} \\ \vdots & \vdots & & \vdots \\ H_{1N} & H_{2N} & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \exp(-i\omega\tau_1) \\ a_2 \exp(-i\omega\tau_2) \\ \vdots \\ a_N \exp(-i\omega\tau_N) \end{bmatrix}. \tag{18}$$

In general, then, crosstalk in a steered array is equivalent to a perturbation of the original weights $\{a_n\}$ after they have been phase shifted to steer a beam. Note that the matrix in (18) is the transpose of H.

The perturbation (18) involves both magnitude and

phase effects, so that pointing error can occur when the time delays τ_n are not all zero. Thus considering the special cases of line and planar arrays, pointing error can arise in any of the nonbroadside steered beams. In general nonplanar arrays, however, pointing error probably occurs in every steered beam. Examples indicate that pointing error is not significant when the crosstalk bound XB is satisfied; nonetheless, the only way to be certain in any particular array is to calculate the pointing error directly using (17) and (18).

Note that (18) reduces to (12) if all the τ_n's are zero. Note also that (18) reduces to the usual steered beam pattern if crosstalk is negligible.

In the previous section it was pointed out that crosstalk can be eliminated before the signals enter the beamformer, provided the coefficients $\{H_{nk}\}$ are precisely known. However, if the corrupted signals have already entered the beam former, then crosstalk can still be eliminated by using (18) to properly adjust the beamformer weights. The use of beamformer weights $w = D^{-1}H^{-1}Da$ will result in the desired weighting Da, where D is the diagonal matrix implicit in (18). Obviously, this is much less efficient than correcting for crosstalk before beamforming because w depends on the steered direction $\mathbf{u}$.

III. EXPECTED BEAM PATTERNS

The crosstalk coefficients in (1) are all real; however, as was pointed out previously, they are complex in the frequency domain if time delays exist in the crosstalk mechanism. Because of this possibility, and because greater generality causes no extra difficulty, in this section we model each crosstalk coefficient H_{nk} as a complex random variable with mean $\bar{H}_{nk}$. Clearly, $\bar{H}_{kk} = 1$ for all k. The time delays $\{\tau_n\}$ and the designed channel weights $\{a_n\}$ are known and fixed; so taking the mean in (18) gives

$$\begin{bmatrix} \bar{c}_1 \\ \bar{c}_2 \\ \vdots \\ \bar{c}_N \end{bmatrix} = \begin{bmatrix} 1 & \bar{H}_{21} & \cdots & \bar{H}_{N1} \\ \bar{H}_{12} & 1 & \cdots & \bar{H}_{N2} \\ \vdots & \vdots & & \vdots \\ \bar{H}_{1N} & \bar{H}_{2N} & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \exp(-i\omega\tau_1) \\ a_2 \exp(-i\omega\tau_2) \\ \vdots \\ a_N \exp(-i\omega\tau_N) \end{bmatrix}, \tag{19}$$

for the mean values $\{\bar{c}_n\}$ of the perturbed weights. It is immediately evident from (19) that if the crosstalk is zero mean, that is, $\bar{H}_{nk} = 0$ for $k \neq n$, then the perturbed weights have mean values equal to the original weights phase shifted to steer a beam in the direction $\mathbf{u}$.

Equation (18) can be used to compute the mean value of the beam pattern given a probabilistic model of the crosstalk. The simplest model is to suppose that the random variables $\{H_{nk}\}_{k \neq n}$ are statistically independent, that is,

$$E[H_{nk}\, H^*_{mj}] = 0, \tag{20}$$

whenever $(n,k) \neq (m,j)$ and $k \neq n$ and $j \neq m$. In (20), * denotes complex conjugates. It is not assumed that the crosstalk coefficients are identically distributed; so we define the notation

$$E[H_{nk}\, H^*_{nk}] = \sigma^2_{nk}, \quad \text{for all } k,n. \tag{21}$$

In particular, $\sigma_{kk} = 1$ because $H_{kk} = 1$ identically. For convenience, let $w = (w_1,\ldots,w_N)$, where

$$w_k = a_k \exp(-i\omega\tau_k), \quad \text{all } k. \tag{22}$$

Then, (18) can be rewritten

$$c_k = w_k + \sum_{\substack{n=1 \\ n \neq k}}^{N} H_{nk} w_n. \tag{23}$$

Using (23), (20), and (21), we can write

$$\begin{aligned} E[c_k c_j^*] &= E\left[\left(w_k + \sum_{\substack{n=1 \\ n \neq k}}^{N} H_{nk} w_n\right)\left(w_j^* + \sum_{\substack{m=1 \\ m \neq k}}^{N} H^*_{mj} w^*_m\right)\right] \\ &= w_k w_j^* + w_k \sum_{\substack{m=1 \\ m \neq j}}^{N} \bar{H}^*_{mj} w^*_m + w_j^* \sum_{\substack{n=1 \\ n \neq k}}^{N} \bar{H}_{nk} w_n \\ &\quad + \sum_{\substack{n=1 \\ n \neq k}}^{N} \sum_{\substack{m=1 \\ m \neq j}}^{N} E[H_{nk} H^*_{mj}] w_n w^*_m \\ &= w_k w_j^* + w_k (\bar{c}_j - w_j)^* + w_j^* (\bar{c}_k - w_k) \\ &\quad + \sum_{\substack{n=1 \\ n \neq \{k,j\}}}^{N} E[H_{nk} H^*_{nj}] w_n w^*_n \\ &= w_k \bar{c}_j^* + w_j^* \bar{c}_k - w_j^* w_k + \delta_{kj} \sum_{\substack{n=1 \\ n \neq k}}^{N} \sigma^2_{nk} w_n w^*_n, \end{aligned} \tag{24}$$

where in the last equation δ_{kj} is Kronecker's delta. For convenience we define for any weighting vector $s = (s_1,\ldots,s_N)$,

$$G_s(\mathbf{v}) = \sum_{k=1}^{N} s_k \exp(i\omega\mu_k). \tag{25}$$

The directional beam pattern with crosstalk is therefore, from (23), $|G_c(\mathbf{v})|^2$, while the directional beam pattern without crosstalk is, from (22), $|G_w(\mathbf{v})|^2$. The expected beam pattern is, from (17),

$$E[F(\mathbf{v})] = \sum_{k=1}^{N} \sum_{j=1}^{N} E[c_k c_j^*] \exp[i\omega(\mu_k - \mu_j)].$$

Substituting (24) and using the notation (25) gives

$$\begin{aligned} E[F(v)] &= \sum_{k=1}^{N} \sum_{j=1}^{N} (w_k \bar{c}_j^* + w_j^* \bar{c}_k - w_j^* w_k) \\ &\qquad \times \exp[i\omega(\mu_k - \mu_j)] + \sum_{k=1}^{N} \sum_{\substack{n=1 \\ n \neq k}}^{N} \sigma^2_{nk} w_n w^*_n \\ &= G_w(\mathbf{v}) G^*_{\bar{c}}(\mathbf{v}) + G_{\bar{c}}(\mathbf{v}) G^*_w(\mathbf{v}) - |G_w(\mathbf{v})|^2 \\ &\quad + \sum_{n=1}^{N} \left(\sum_{\substack{k=1 \\ k \neq n}}^{N} \sigma^2_{nk}\right) |w_n|^2. \end{aligned} \tag{26}$$

Using the definition (25), note that

$$\begin{aligned} G_{\bar{c}}(\mathbf{v}) &\approx \sum_{k=1}^{N} \bar{c}_k \exp(i\omega\mu_k) \\ &\approx \sum_{k=1}^{N} w_k \exp(i\omega\mu_k) + \sum_{k=1}^{N} (\bar{c}_k - w_k) \exp(i\omega\mu_k) \\ &\approx G_w(\mathbf{v}) + G_{\bar{c}-w}(\mathbf{v}). \end{aligned}$$

Substituting this identity into (26) gives the result

$$E[F(\mathbf{v})] = |G_{u}(\mathbf{v})|^2 + 2\,\mathrm{Re}\left[G_{u}(\mathbf{v})G^{*}_{u}(\mathbf{v})\right] + \sum_{n=1}^{\hat{N}}\left(\sum_{k=1}^{\hat{N}}\sigma^{2}_{nk}\right)|w_n|^2 . \quad (27)$$

It is important to notice that the only way the middle term on the right-hand side of (27) can be identically zero is if $\bar{c} = w$. As was pointed out above, $\bar{c} = w$ if and only if the crosstalk is zero mean, that is, $\bar{H}_{nk} = 0$ for $k \neq n$. Since crosstalk is not zero mean in general, (27) cannot be further simplified.

The other two terms in (27) also have important interpretations. The first term on the right-hand side of (27) is the directional beam pattern free of crosstalk. The third term in (27) is a positive constant independent of $\mathbf{u}$ and the steered direction $\mathbf{v}$, as can be seen from the definition (22) of w. Thus, with zero mean crosstalk, the middle term of (27) is zero; so the expected beam pattern cannot have nulls. With nonzero mean crosstalk, it is possible, though unlikely, that the middle term may be sufficiently negative so that $E[F(\mathbf{v})]$ has nulls. In no case, however, can $E[F(\mathbf{v})]$ be negative.

Equation (27) can be used to derive an upper bound on the maximum crosstalk variance to guarantee that the sidelobe level of the expected beam pattern does not increase more than a specified amount. See Appendix B for the case of zero mean identically distributed H_{nk}.

It is also clear from (27) that the expected beam pattern cannot have pointing error when the crosstalk is zero mean. In other words, the expected maximum response occurs when $\mathbf{v}$ equals the steered direction $\mathbf{u}$, or

$$\max_{\mathbf{v}} E[F(\mathbf{v})] = E[F(\mathbf{u})] , \quad (28)$$

with zero mean crosstalk. Pointing errors can only arise in the expected beam pattern when the crosstalk is not zero mean.

It is desirable to correct for nonzero mean crosstalk levels before the signals enter the beamformer. Taking the mean in (4) gives

$$|\bar{H}_{nk}| < \epsilon .$$

Requiring the crosstalk upper bound (6) to hold implies that $\bar{H}^{-1}$ exists. Thus we can write

$$Q(t) = \bar{H}^{-1}U(t) . \quad (29)$$

Using the vector $Q(t)$ as the beamformer input vector results in beamformer input channels with zero mean crosstalk. The reason is that the correction (29) modifies the original model (2), which becomes instead

$$Q(t) = \bar{H}^{-1}HV(t) = H_{\mathrm{eff}}V(t) . \quad (30)$$

Clearly, the effective crosstalk matrix H_{eff} is such that $\bar{H}_{\mathrm{eff}} = I$. In other words, H_{eff} has entries that are zero mean on the off-diagonal and unit mean on the main diagonal. The remarks immediately following (3) concerning the use of the inverse of H are directly applicable here for the use of the inverse of $\bar{H}$.

When the crosstalk coefficients are modeled as a joint Gaussian distribution, so that they are not statistically independent, expected beam patterns can still be derived. The interested reader is referred to Refs. 2 and 3 for a general discussion which is relevant here, even though neither discuss crosstalk.

IV. EXAMPLES

As an example of the effect of crosstalk on beam patterns, consider an equispaced line array with $N = 50$ omnidirectional sensors spaced notionally 1 m apart. The design weights $\{a_n\}$ are Taylor weights for a -30-dB sidelobe level (with the number $\bar{n}$ of controlled nulls set equal to 5). Crosstalk between any pair of channels is assumed to be such that the coefficients $\{H_{nk}\}$ are all real and positive. These constants are selected from a uniform pseudorandom distribution on the interval $[0,\epsilon]$, where ϵ is chosen to correspond to a specified maximum crosstalk level. Thus $\bar{H}_{nk} = \epsilon/2$ and is not zero mean. The crosstalk bound, XB, is -34 dB in this case. Figure 1(a)–(d) shows the broadside beam patterns for a maximum crosstalk level of -74, -54, -34, and -24 dB, respectively. By inspection, crosstalk of -74 dB has virtually no impact on the beam pattern [see Fig. 1(a)], while a crosstalk level of -54 dB, a full 20 dB below the crosstalk bound XB $\doteq -34$ dB, raises the peak sidelobes by about 1 dB [see Fig. 1(b)]. When the crosstalk is equal to the upper bound of -34 dB, the beam pattern is significantly perturbed and has a peak sidelobe about 8 dB above the design sidelobe level of -30 dB [see Fig. 1(c)]. When crosstalk reaches -24 dB, the peak sidelobe is 13 dB above the design level [see Fig. 1(d)].

We point out that the largest individual perturbation of the original Taylor weights is 0.9%, 8.6%, 60%, and 111% in the cases corresponding to Fig. 1(a)–(d), respectively. The percentage perturbation was calculated after normalizing the Taylor weights and the perturbed weights to sum to one.

Other realizations of the crosstalk matrix H have been computed, but are not presented here. They reinforce the fundamental point that, for this 50-element array, crosstalk levels should be kept below -54 dB, or 20 dB below XB $\doteq -34$ dB. The question of whether or not this observation remains true for larger values of N naturally arises. Consider, then, a 100-sensor equispaced array with 1-m spacing between sensors and Taylor shaded for -30-dB sidelobes (with $\bar{n} = 10$ controlled nulls). For crosstalk levels of -70, -60, -50, and -40 dB, the beam patterns are shown in Fig. 2(a)–(d), respectively. The largest individual weight perturbation was 2.1%, 6.4%, 18%, and 45%, respectively. The crosstalk bound XB is -40 dB. Examination of these figures shows that crosstalk levels should be kept below -60 dB, or again 20 dB below the upper bound XB. Evidently, therefore, the crosstalk should always be kept 20 dB below the bound XB to prevent significant beam pattern degradation.

Expected beam patterns for these examples are not presented for two reasons. First, as was pointed out previously, it is desirable to correct nonzero mean crosstalk to zero mean [using (30)] before the signals enter the beamformer to prevent pointing error. The examples here are not zero mean. Second, assuming zero mean crosstalk, then (27) clearly shows that the expected beam pattern is the crosstalk-free beam pattern plus a constant term. The expected beam pat-

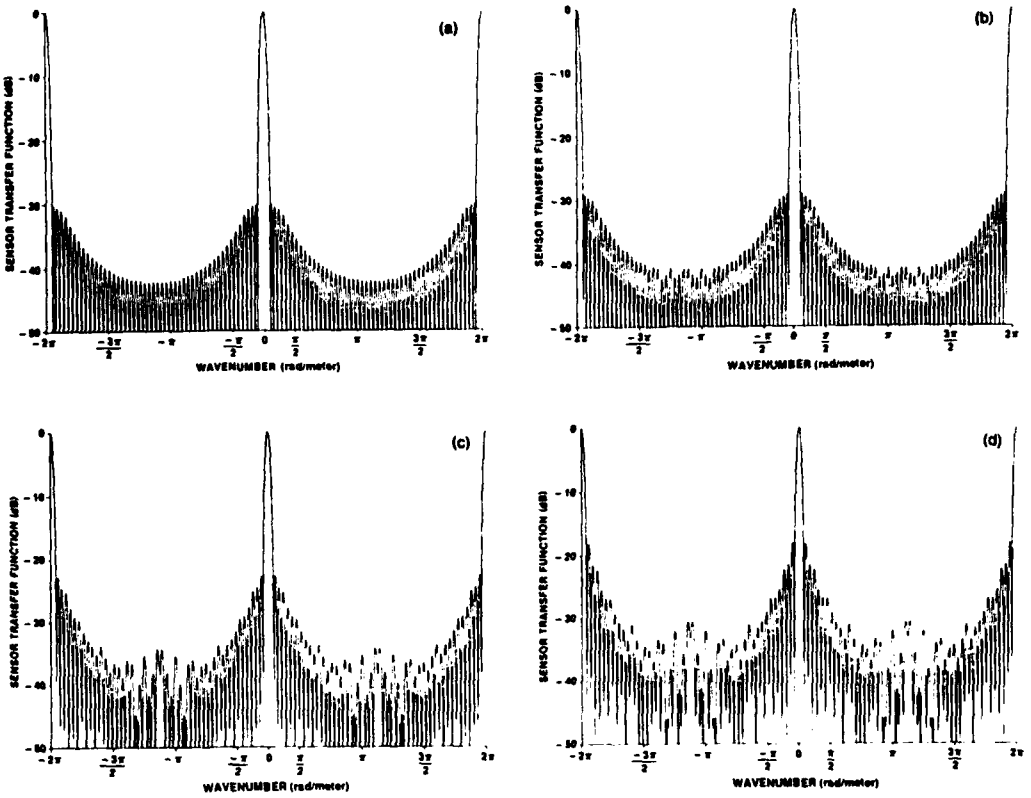


FIG. 1. Broadside beam patterns for 50-sensor array crosstalk levels: (a) = − 74 dB, (b) = − 54 dB, (c) = − 34 dB, and (d) = − 24 dB.

terns for these examples, after correcting the crosstalk to zero mean, are merely the appropriate Taylor beam patterns with an appropriate constant term (independent of angle) added; hence, no nulls appear in the expected beam patterns. Examples of this kind are given in Ref. 2.

The pointing error was determined for the above 100 sensor line array when the maximum crosstalk level was set equal to the bound XB = − 40 dB. Beams were steered from broadside to endfire in one degree increments, and the pointing error determined by direct numerical calculation using (17) and (18). The pointing error in all but one beam was found to be within ± 0.005 deg of the steered direction. The exceptional beam had pointing error of − 0.14 deg. If the crosstalk coefficients are fixed once and for all, then pointing error is a smoothly varying function of steering angle. In these calculations, however, different crosstalk coefficients were calculated for different beams; this explains the one exceptional beam. One tentative conclusion in this example is that, as long as the crosstalk bound XB is satisfied, pointing error is not significant. The one exceptional beam suggests, however, that certain realizations of the crosstalk coefficients might possibly result in surprisingly large pointing error even when the bound XB is satisifed.

V. CONCLUSION

A maximum permissible level of crosstalk in an array of arbitrary geometric configuration has been presented. Also, the beam pattern of an arbitrary array with arbitrary crosstalk, steered in any direction, has been derived.

Crosstalk can result in pointing error when steering the array, an effect of considerable importance. When a random variable model of crosstalk is appropriate, it is worthwhile in practice to correct for nonzero mean crosstalk before the signals enter the beamformer. If this is done, particular realizations of the crosstalk coefficients may still result in array pointing error, but the expected pointing error is zero because the expected beam patterns do not have pointing error.

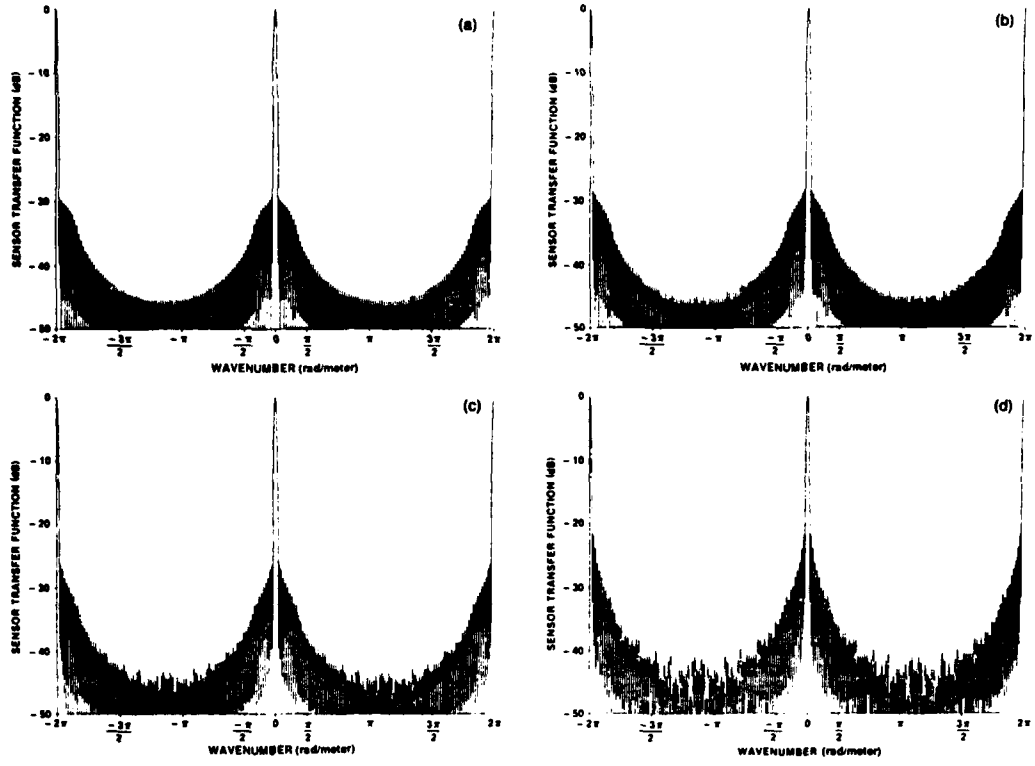


FIG. 2. Broadside beam patterns for 100-sensor array crosstalk levels: (a) = − 70 dB, (b) = − 60 dB, (c) = − 50 dB, and (d) = − 40 dB.

ACKNOWLEDGMENTS

The author wishes to thank Dr. Wayne Strawderman and Dr. Albert Nuttall for their useful comments and suggestions.

APPENDIX A: GERSHGORIN'S THEOREM

Gershgorin's theorem defines a closed set in the complex z plane within which all the eigenvalues of a general complex valued matrix must lie. Let $A = [a_{ij}]$ denote a given $n \times n$ matrix. Define

$$r_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} |a_{ij}|, \quad i = 1,\ldots,n.$$

Then, the eigenvalues of A lie in the region of the complex z plane consisting of the union of all the closed disks:

$$|z - a_{ii}| \leq r_i, \quad i = 1,\ldots,n. \tag{A1}$$

A proof can be found in many places, for example, Ref. 1, p. 146. If the main diagonal of A is constant, that is, $a_{ii} = a$ for all i, then the disks are concentric. It follows that the eigenvalues of A lie in the closed disk:

$$|z - a| \leq \max_{1 \leq i \leq n} r_i. \tag{A2}$$

The result (A2) together with (4) proves (5) in the main text.

APPENDIX B: MAXIMUM CROSSTALK VARIANCE FOR SPECIFIED SIDELOBE LEVEL

Attention in this Appendix is restricted to the case where the crosstalk coefficients H_{nk}, $n \neq k$, are identically distributed with zero mean and common variance σ^2. It is required to find the largest possible value of σ for a specified maximum increase in the sidelobe level of the expected beam pattern (27). Let the sidelobe level of the crosstalk-free beam pattern $G(\mathbf{v})$ be denoted

$$L_G = B^2/A^2,$$

where B is the level of the response of $|G(\mathbf{v})|$ when $\mathbf{v}$ lies in the sidelobe regime, and A is the response of $|G(\mathbf{v})|$ when $\mathbf{v}$ equals the steered direction $\mathbf{u}$. Thus

$$A = \left| \sum_{n=1}^{N} w_n \right|. \tag{B1}$$

From (27), we have in this case

$$E[F(\mathbf{v})] = |G(\mathbf{v})|^2 + (N-1)\alpha_w\sigma^2, \tag{B2}$$

where

$$\alpha_w = \sum_{n=1}^{N} |w_n|^2. \tag{B3}$$

Therefore, the sidelobe level of the expected beam pattern may be written

$$L_X = \frac{B^2 + (N-1)\alpha_w\sigma^2}{A^2 + (N-1)\alpha_w\sigma^2}. \tag{B4}$$

We seek the largest value of σ for which

$$10\log L_X - 10\log L_G \leqslant 10\log(\Delta^2), \tag{B5}$$

where $10\log(\Delta^2)$ is the specified level (in decibels) that the expected beam pattern sidelobes are allowed to increase. Equivalently,

$$\frac{L_X}{L_G} = \frac{1 + (N-1)\alpha_w\sigma^2/B^2}{1 + (N-1)\alpha_w\sigma^2/A^2} \leqslant \Delta^2. \tag{B6}$$

Solving for σ^2 gives

$$\sigma^2 \leqslant \frac{\Delta^2 - 1}{N-1} \frac{L_G}{1 - L_G\Delta^2} \frac{|\Sigma_{n=1}^{N} w_n|^2}{\Sigma_{n=1}^{N} |w_n|^2}, \tag{B7}$$

which is the desired relationship. It holds only for zero mean and identical variance crosstalk coefficients.

If $10\log\Delta^2 = 1$ dB, $10\log L_G = -25$ dB, and identical weights $w_n = \text{constant} \neq 0$, then (B7) gives approximately

$$\sigma^2 \leqslant 0.000\,822,$$

or, taking 10 log of both sides,

$$20\log(\sigma) \leqslant -31 \text{ dB}. \tag{B8}$$

On the other hand, if $10\log\Delta^2 = 2$ dB, then (B7) gives instead

$$20\log(\sigma) \leqslant -27.3 \text{ dB}. \tag{B9}$$

Interestingly, (B8) and (B9) are independent of N in this case because of the assumption of constant weights.

The bound (B7) on the maximum crosstalk variance is more strict when the array is shaded (that is, $w_n \neq \text{const}$) than when it is unshaded ($w_n = \text{constant}$). The proof of this fact follows from the conditions for equality in the Cauchy–Schwartz inequality.

[1] M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities* (Prindle, Weber, and Schmidt, Boston, 1964).

[2] A. H. Nuttall, *Effects of Random Shadings, Phasing Errors, and Element Failures on Beam Patterns of Linear and Planar Arrays*, NUSC Technical Report 6191, Naval Underwater Systems Center, New London, CT, 14 March 1980.

[3] D. J. Ramsdale and R. A. Howerton, "Effect of Element Failure and Random Errors in Amplitude and Phase on the Sidelobe Levels Attainable with a Linear Array," J. Acoust. Soc. Am. **68**, 901–906 (1980).

A Two-Parameter Family Of Weights For Nonrecursive Digital Filters And Antennas

R. L. Streit

A Two-Parameter Family of Weights for Nonrecursive Digital Filters and Antennas

ROY L. STREIT

Abstract—We derive analytically a two-parameter family of weights for use in finite duration nonrecursive digital filters and in finite aperture antennas. This family of weights is based on the Gegenbauer orthogonal polynomials, which are a generalization of both Legendre and Chebyshev polynomials. It is shown that one parameter controls the main lobewidth and the other parameter controls the sidelobe taper. For a fixed main lobewidth, it is observed that the Gegenbauer weights can achieve a dramatic decrease in sidelobes "far removed" from the main lobe in exchange for a "small" increase in the first sidelobe adjacent to the main lobe.

The Gegenbauer weights are derived first for discretely sampled apertures and filters. An appropriate limit is then taken to produce the Gegenbauer weighting function for continuously sampled apertures and filters. The continuous Gegenbauer weighting function contains the Kaiser-Bessel function as a special case. It is thus established that the Kaiser-Bessel function is implicitly based on Chebyshev polynomials of the second kind. Furthermore, the Dolph-Chebyshev/van der Maas weights are a limiting case of the discrete/continuous Gegenbauer weights.

I. Introduction

THE choice of weights in the design of nonrecursive digital filters and antenna apertures is an important problem for which there is a large literature. In this paper we present the Gegenbauer weighting function, so named because it is based on the Gegenbauer orthogonal polynomials [1]. The Gegenbauer weights may be applied equally well to nonrecursive digital filters and both discrete and continuous antenna apertures. The resulting FIR filter coefficients can be used as a shading function for the spectrum analysis of sampled data to reduce sidelobe leakage. Our discussion in this paper will be restricted to the antenna form of the problem merely to avoid unnecessary complication in the presentation.

The Gegenbauer design is a two-parameter family of weighting functions. One parameter, z_0, is used to control the beamwidth. The other parameter, μ, is used to achieve sidelobe taper. Both z_0 and μ may be varied continuously and independently of each other. The Gegenbauer design is especially useful in achieving dramatic decreases in distant sidelobes in exchange for "small" increases in the first sidelobe adjacent to the main lobe. Conversely, dramatic increases in distant sidelobes can be exchanged for "small" decreases in the first sidelobe. This will be clarified by the examples.

Manuscript received August 11, 1982; revised November 30, 1982. This work was supported in part by the Office of Naval Research under Project RR014-07-01 and by The Independent Research Program of the Naval Underwater Systems Center.

The author was on leave at the Department of Operations Research, Stanford University, Stanford, CA 94305. He is with the New London Laboratory, Naval Underwater Systems Center, New London, CT 06320, and the Department of Mathematics, University of Rhode Island, Kingston, RI 06320.

The Gegenbauer weights are derived first for a finite discrete aperture. An appropriate limit then gives the Gegenbauer weighting function for a bounded continuous aperture. Many similarities between the Gegenbauer weights and the Dolph-Chebyshev/van der Maas weights [2], [3] will be evident from the derivation. In fact, these latter weights are limiting forms, as $\mu \to 0$, of Gegenbauer weights. Also, the Kaiser-Bessel weighting function [4, pp. 232-233] for the continuous aperture is the special case $\mu = 1$ of the Gegenbauer design. This shows that the Kaiser-Bessel function is implicitly based upon Chebyshev polynomials of the second kind, a fact which seems to have escaped notice until now. This is interesting since, as is well known, the Dolph-Chebyshev/van der Maas weights are based on Chebyshev polynomials of the first kind.

One drawback to the van der Maas weighting function for the continuous aperture is that it has δ-function spikes at the aperture endpoints. The Gegenbauer function does not have this feature; that is, the Gegenbauer weighting function for the continuous aperture is a bounded continuous real-valued function across the whole aperture. However, since the van der Maas function is a limiting case of the Gegenbauer function as $\mu \to 0$, the Gegenbauer function must approximate this behavior in the neighborhood of $\mu = 0$. The Taylor design [5] is an alternative way to overcome this δ-function behavior of the van der Maas function, but it is unrelated to any of the Gegenbauer designs. The proof of this statement is self-evident from the examples presented later.

The Gegenbauer polynomials $C_n^{\mu}(x)$ are defined here precisely as in Szegö [1] which is used as our standard both in function definition and notation, with only two exceptions. Szegö uses the notation $P_n^{(\mu)}(x)$ instead of $C_n^{\mu}(x)$ and refers to them as the ultraspherical polynomials. This paper will not attempt to recapitulate any of the known facts about the polynomials that can be referenced in Szegö. It suffices to say here only that $C_n^{\mu}(x)$ is a real valued polynomial of degree precisely n, and that the system $\{C_0^{\mu}(x), C_1^{\mu}(x), C_2^{\mu}(x), \cdots\}$ is orthogonal on the real interval $[-1, +1]$ with respect to the weight function $(1 - x^2)^{\mu - 1/2}$ provided $\mu > -\frac{1}{2}$, $\mu \neq 0$. Moreover, by taking appropriate limits and using their hypergeometric functional form, $C_n^{\mu}(x)$ can be defined for all real μ. See [1, eq. (4.7.7)]. In particular, if $T_n(x)$ and $U_n(x)$ denote the Chebyshev polynomials of the first and second kinds, respectively, then [1, eq. (4.7.8), (4.7.17)]

$$C_0^0(x) = T_0(x), \quad \lim_{\mu \to 0} \frac{C_n^{\mu}(x)}{\mu} = \frac{2}{n} T_n(x), \quad n \geq 1 \tag{1}$$

and [1, eq. (4.7.2)]

$$C_n^1(x) = U_n(x), \quad n \geq 0. \tag{2}$$

The derivation of formulas more general than are perhaps necessary in the antenna application is relegated to the Appendix. Special cases of these formulas will be extracted as needed and used without comment in the main body of this paper; however, every effort will be made to motivate the discussion.

II. Gegenbauer Weights for a Discrete Aperture

The Gegenbauer design for a finite discrete aperture is derived for a single frequency half-wavelength equispaced linear array of omnidirectional elements. Other than the steering factor, we will always assume the aperture (discrete or continuous) is symmetrically weighted about the geometric center of the array. The array axis is taken to be the x-axis and all angles are measured from a line normal to the array axis.

Let N be the number of elements in the array (hence $N \geq 2$), and let the positions of these elements be $x_k = k\lambda/2$, $k = 1, 2, \cdots, N$, where λ is the wavelength of the design frequency. (In the Appendix, λ denotes an arbitrary real variable, not frequency.) If the array is steered to look in the direction θ_l, $-\pi/2 \leq \theta_l \leq \pi/2$, and if the array receives a plane wave of wavelength λ from the arrival direction θ_a, $-\pi/2 \leq \theta_a \leq \pi/2$, then the complex transfer function of a linear beamformer is given by

$$F(u) \triangleq \sum_{k=1}^{N} w_k \exp(-i\pi k u) \tag{3}$$

where

$$u \triangleq \sin\theta_a - \sin\theta_l \tag{4}$$

and $\{w_k\}_1^N$ are the individual element weights. Symmetrical weighting is assumed, so $w_{N-k+1} = w_k$ for all k. Positive weighting is desirable, but not necessary.

The Dolph-Chebyshev design proceeds as follows for a design specification of $-S$ dB peak sidelobe level. Let

$$z_0 \triangleq \tfrac{1}{2}\{[r + \sqrt{r^2 - 1}]^{1/n} + [r - \sqrt{r^2 - 1}]^{1/n}\}, \quad r \triangleq 10^{S/20} \tag{5}$$

and $n \triangleq N - 1$. Notice that $z_0 > 1$ if and only if the peak sidelobe level is lower than the level of the maximum response axis, or MRA. From (A20) of the Appendix, the expansion

$$T_n(z_0 \cos u) = \sum_{k=0}^{\lfloor n/2 \rfloor}{}' c_{k,n}(z_0) \cos[(n - 2k)u] \tag{6}$$

clearly exists, where the prime on the summation means that $\frac{1}{2}$ the last term in the sum is taken if n is even, and all of it is taken if n is odd. From (A21) we have explicitly

$$c_{k,n}(z_0) = n(n-k-1)! \sum_{m=0}^{k} \frac{(m)_{k-m}(z_0^2 - 1)^m z_0^{n-2m}}{m!\,(k-m)!\,(n-k-m)!}. \tag{7}$$

The coefficients $c_{k,n}(z_0)$ were first given in this form by van der Maas [3], who derived them using a method different from that in the Appendix. By inspection, notice that $c_{k,n}(z_0) > 0$ for all k whenever $z_0 > 1$. The coefficients $c_{k,n}(z_0)$ yield the element weights $\{w_k\}_1^N$ when we define

for N even:

$$\left.\begin{aligned} w_{N-k+1} = w_k &\triangleq \tfrac{1}{2} c_{t(k),N-1}(z_0) \\ t(k) &\triangleq \tfrac{N}{2} - k \end{aligned}\right\} \quad k = 1, 2, \cdots, \tfrac{N}{2} \tag{8}$$

for N odd:

$$\left.\begin{aligned} w_{n-k+1} = w_k &\triangleq \frac{1}{2} c_{t(k),N-1}(z_0) \\ t(k) &\triangleq \frac{N+1}{2} - k \end{aligned}\right\} \quad k = 1, 2, \cdots, \frac{N+1}{2}. \tag{9}$$

Thus, the complex transfer function (3) is given explicitly for these weights by

$$F(u) = e^{i\pi(N+1)u/2} T_{N-1}(z_0 \cos(\tfrac{1}{2}\pi u)); \tag{10}$$

the maximum response occurs for $u = 0$,

$$F(0) = T_{N-1}(z_0); \tag{11}$$

and the smallest positive value of u such that $F(u) = 0$ is given by

$$u_0 = \frac{2}{\pi} \arccos\left(\frac{1}{z_0} \cos\left(\frac{\pi}{2(N-1)}\right)\right). \tag{12}$$

The half beamwidth as measured to the first null from the MRA is precisely u_0.

The Gegenbauer design proceeds in an analogous fashion. We replace the old constant z_0 by a new variable z_μ which will be defined later (30); however, for $\mu = 0$, z_μ is still defined by (5). Now, in the expansion

$$C_n^\mu(z_\mu \cos u) = \sum_{k=0}^{\lfloor n/2 \rfloor}{}' b_{k,n}(z_\mu) \cos[(n - 2k)u] \tag{13}$$

the coefficients $b_{k,n}(z_\mu)$ depend on z_μ and are given explicitly by

$$b_{k,n}(z_\mu) = 2(\mu)_{n-k} \sum_{m=0}^{k} \frac{(\mu + m)_{k-m}(z_\mu^2 - 1)^m z_\mu^{n-2m}}{m!\,(k-m)!\,(n-k-m)!}. \tag{14}$$

Both of these identities are special cases of (A18) and (A19) of the Appendix. Note that $b_{k,n}(z_0) > 0$ for all k, provided that $z_\mu > 1$ and $\mu > 0$. Note also that, by (1), (14) reduces to (7) in the limit as $\mu \to 0$. For numerical computation, the following form is preferred to (14). Let $A = 1 - z_\mu^{-2}$, so that $0 < A < 1$ when $z_\mu > 1$, and then compute the right-hand side of

$$\frac{b_{k,n}(z_\mu)}{2\mu z_\mu^n} = \frac{1}{n-k} \binom{\mu + n - k - 1}{n - k - 1} \cdot \sum_{m=0}^{k} \binom{\mu + k - 1}{k - m} \binom{n-k}{m} A^m. \tag{15}$$

The binomial coefficients are defined here for any real number α and any nonnegative integer p by

$$\binom{\alpha}{0} \triangleq 1, \binom{\alpha}{p} \triangleq \frac{\alpha(\alpha-1)\cdots(\alpha-p+1)}{p!}, \quad p \geq 1 \tag{16}$$

although they are best computed recursively using

$$\binom{\alpha}{p} = \frac{\alpha - p + 1}{p} \binom{\alpha}{p-1}, \quad p \geq 1 \tag{17}$$

to avoid floating point overflow at some intermediate point in the computation.

It should be pointed out that (15) can be evaluated numerically for all μ since, for fixed n and k, (15) is a polynomial in μ. However, (15) is correct only if $\mu \neq -1, -2, -3, \cdots$. If coefficients are required for, say, $\mu = -5$, both sides of (13) must first be divided by $\mu + 5$ and the limit taken as $\mu + 5 \to 0$. Consequently, in (15), the factor $\mu + 5$ must be divided out algebraically before numerical computation begins.

The coefficients $b_{k,n}(z_\mu)$ yield element weights $\{w_k\}_1^N$ when we define

for N even:

$$\left.\begin{aligned} w_{N-k+1} = w_k &\triangleq \frac{1}{2} b_{t(k),N-1}(z_\mu) \\ t(k) &\triangleq \frac{N}{2} - k \end{aligned}\right\} \quad k = 1, 2, \cdots, \frac{N}{2} \tag{18}$$

for N odd:

$$\left.\begin{aligned} w_{N-k+1} = w_k &\triangleq \frac{1}{2} b_{t(k),N-1}(z_\mu) \\ t(k) &\triangleq \frac{N+1}{2} - k \end{aligned}\right\} \quad k = 1, 2, \cdots, \frac{N+1}{2}. \tag{19}$$

With these weights, the complex transfer function (3) is given explicitly by

$$F(u) = e^{in(N+1)u/2} C_{N-1}^{\mu}(z_\mu \cos(\tfrac{1}{2}\pi u)). \tag{20}$$

The maximum response of $F(u)$ should occur for $u = 0$, and is

$$F(0) = C_{N-1}^{\mu}(z_\mu). \tag{21}$$

(For a discussion of unusual situations when the MRA might not occur at $u = 0$, see below in this section.)

The smallest positive value of u satisfying $F(u) = 0$ is given by

$$u_\mu \triangleq \frac{2}{\pi} \arccos\left(\frac{1}{z_\mu} x_{N-1}^{(\mu)}\right) \tag{22}$$

where $x_{N-1}^{(\mu)}$ is the largest zero of the Gegenbauer polynomial $C_{N-1}^{\mu}(x)$. Thus, for $\mu > -1/2$, $x_{N-1}^{(\mu)}$ must lie in the open interval $(-1, +1)$. In fact, it must be very near $+1$ for values of μ of interest in this application. An explicit analytic expression for $x_{N-1}^{(\mu)}$ is not known except in certain special cases (e.g., the Chebyshev polynomials) and so must be solved for numerically. This minor difficulty is readily overcome using Newton-Raphson iteration. Recall $n = N - 1$. Since [1, eq. (4.7.14)]

$$\frac{d}{dx} C_n^{\mu}(x) = 2\mu C_{n-1}^{\mu+1}(x) \tag{23}$$

the Newton-Raphson iteration is

$$y_{k+1} = y_k - \frac{C_n^{\mu}(y_k)}{2\mu C_{n-1}^{\mu+1}(y_k)}, \quad k = 1, 2, \cdots \tag{24}$$

$$y_1 \triangleq x_n^{(0)} = \cos\left(\frac{\pi}{2n}\right).$$

The ratio in (24) is perhaps best evaluated by computing two different sequences

$$\{C_p^{\mu}(y_k)\}_{p=1}^{n} \quad \text{and} \quad \{C_p^{\mu+1}(y_k)\}_{p=1}^{n} \tag{25}$$

numerically from the fundamental recursion [1, eq. (4.7.17)]

$$pC_p^{\alpha}(x) = 2(p + \alpha - 1)xC_{p-1}^{\alpha}(x) - (p + 2\alpha - 2)C_{p-2}^{\alpha}(x),$$
$$p = 2, 3, 4, \cdots, n \tag{26}$$
$$C_0^{\alpha}(x) = 1, \quad C_1^{\alpha}(x) = 2\alpha x.$$

The recursion (26) is valid for $\alpha \neq 0, -1, -2, -3, \cdots$. This method may have weaknesses whenever μ is very close to 0 (say, $|\mu| \leq 10^{-4}$) because of the division by μ in (24); however, μ would normally be taken either equal to 0 (to give the Dolph-Chebyshev design) or else sufficiently different from 0 to affect sidelobe levels appreciably. This latter stipulation seems to require $|\mu| > 10^{-4}$. In the antenna application, then, computation of the Newton-Raphson iteration step from the recursion (26) seems perfectly safe whenever a special precaution is taken for $\mu = 0$. In practice this author has never seen the iteration require more than four steps, and he has never seen it converge to the wrong point. If, however, it should ever happen to converge to the wrong point, the Newton-Raphson iteration can be restarted with the new initial point $y_1 = 1$. Also, the inequality [1, eq. (6.21.3)]

$$\frac{\partial}{\partial \mu} x_n^{(\mu)} < 0 \quad \text{for all } \mu \tag{27}$$

implies that

$$x_n^{(\mu)} < x_n^{(0)} = \cos\left(\frac{\pi}{2n}\right) < x_n^{(-\mu)}, \quad \mu > 0, \tag{28}$$

which can serve as a check. Incidentally, inequality (27) holds for *all* the positive zeros $C_n^{\mu}(x)$, not merely the largest one.

The reason for all this concern over calculation of the half-beam width (22) is simply to be able to make fair comparisons between sidelobe levels of different Gegenbauer designs, that is, different values of μ. It is well known that the sidelobe levels in Dolph-Chebyshev beam patterns are sensitive functions of the beamwidth, and there is every reason to expect similar behavior in the Gegenbauer designs. Therefore, as μ is varied it is helpful to maintain a fixed beamwidth; specifically, we always require $u_\mu = u_0$ for all μ. This in turn, from (22) and (12), gives

$$\frac{1}{z_\mu} x_{N-1}^{(\mu)} = \frac{1}{z_0} \cos\left(\frac{\pi}{2(N-1)}\right) \tag{29}$$

or, converting convenience into a definition,

$$z_\mu \triangleq z_0 \, x_{N-1}^{(\mu)} \sec\left(\frac{\pi}{2(N-1)}\right). \tag{30}$$

From (30) it is now clear that computing the largest zero, x_{N-1}^{μ}, of $C_{N-1}^{\mu}(x)$ is of considerable importance.

With the definition (30), all Gegenbauer designs with different values of μ and fixed z_0 have the same beamwidth as measured to the first null off the MRA. Thus, the beamwidth is varied simply by changing the value of z_0 in exactly the same way as in Dolph-Chebyshev, i.e., (5).

An interesting consequence of (30) is that z_μ might not always

be greater than 1 for all $\mu \geq 0$. This observation follows immediately from the derivative (27). Hence, for some critical positive value of μ, say μ^*, we have $z_{\mu^*} = 1$. In (15) the number A is negative for $\mu > \mu^*$, so the positivity of the weights cannot be guaranteed without direct calculation because (15) is an alternating series for $\mu > \mu^*$. At the critical point μ^*, $A = 0$ and the sum in (15) collapses to a single term. Simplifying gives

$$h_{k,n}(z_{\mu^*}) = 2\binom{n-k+\mu^*-1}{n-k}\binom{k+\mu^*-1}{k} \tag{31}$$

which can be found also in Szego [1, eq. (4.9.19)]. The weights for the critical case $\mu = \mu^*$ can now be varied merely by changing μ^*. In particular, for $\mu^* = 1$, (31) gives the uniformly weighted array; that is, $\omega_k = 1$ for all k. The beamwidth obtained from the weights (31) depends on (and only on) the critical value μ^* because μ^* implicitly depends on z_0.

Since Gegenbauer designs have the two parameters z_0 and μ, with z_0 controlling main lobewidth, the parameter μ must control sidelobe behavior. From (20) and (30) we see that sidelobes occur for u satisfying

$$|z_\mu \cos(\tfrac{1}{2}\pi u)| \leq \cos(\tfrac{1}{2}\pi u_0) < 1. \tag{32}$$

In the sidelobe region, then, we can define

$$\cos\phi = z_\mu \cos(\tfrac{1}{2}\pi u), \qquad 0 < \phi < \pi.$$

For the moment let us suppose $0 < \mu < 1$. Then, from Szego [1, eq. (7.33.5)]

$$(\sin\phi)^\mu \, |C_n^\mu(\cos\phi)| < 2^{1-\mu} n^{\mu-1}/\Gamma(\mu) \tag{33}$$

so the transfer function $F(u)$ must satisfy

$$|F(u)| < (1 - z_\mu^2 \cos^2(\tfrac{1}{2}\pi u))^{-\mu/2} \, 2^{1-\mu} n^{\mu-1}/\Gamma(\mu) \tag{34}$$

throughout the sidelobe region defined by (32). For μ outside the (0, 1) interval, but excluding $\mu = 0, -1, -2, \cdots$, the sharpness of the inequality (34) is lost. A special case of a result given in Szego [1, eq. (8.21.14) with $p = 1$] implies that

$$|F(u)| \leq (1 - z_\mu^2 \cos^2(\tfrac{1}{2}\pi u))^{-\mu/2} \, 2^{1-\mu}\binom{n+\mu-1}{n} + O(n^{\mu-2}) \tag{35}$$

throughout the sidelobe region defined by (32). For μ outside (35) is asymptotic to $n^{\mu-1}/\Gamma(\mu)$ as $n \to \infty$, so the leading term of the right-hand side of (35) is asymptotic to the right-hand side of (34). For fixed μ, the right-hand side of (35) appears to be an excellent envelope for the sidelobes of the Gegenbauer designs.

For $\mu > 0$, it is clear from (35) that the sidelobe envelope must steadily decay as u approaches endfire, i.e., $u = 1$. Since [use (26)]

$$C_n^\mu(0) = \begin{cases} 0, & \text{if } n \text{ odd} \\ (-1)^m\binom{\mu+m-1}{m}, & \text{if } n = 2m \end{cases}$$

we have

$$|F(1)| = |C_n^\mu(0)| \approx 2^{1-\mu} n^{\mu-1}/\Gamma(\mu) \tag{36}$$

approximately, for n even. Contrasting this approximation with the inequality (35) leads to the conclusion that (36) is an excellent approximation to the sidelobe envelope for both even and odd n. Thus, we utilize (36) for all n. Applying results proved below in another context [specifically, set $v = 0$ in (54) and (55)] gives an approximation for the maximum response

$$|F(0)| \approx \frac{2^{1-\mu} n^{2\mu-1}}{\Gamma(\mu)} \sqrt{\frac{\pi}{2}} \frac{I_{\mu-1/2}(\tau)}{\tau^{\mu-1/2}} \tag{37}$$

with r defined by (5) and

$$\tau \triangleq [(\operatorname{arccosh} r)^2 + \pi^2/4 - j_{\mu-1/2}^2]^{1/2} \tag{38}$$

where $j_{\mu-1/2}$ is the smallest positive zero of the Bessel function $J_{\mu-1/2}(x)$ of the first kind and order $\mu - 1/2$, and $I_{\mu-1/2}(x)$ is the modified Bessel function of the first kind and order $\mu - 1/2$. Therefore, we have the relative level

$$\frac{|F(1)|}{|F(0)|} \approx n^{-\mu}\sqrt{\frac{2}{\pi}}\left\{\frac{I_{\mu-1/2}(\tau)}{\tau^{\mu-1/2}}\right\}^{-1}. \tag{39}$$

This result happens to be exact for $\mu = 0$, the Dolph–Chebyshev case, as can be easily verified. Evidently this result also implies that the sidelobe height at endfire is a function of n, even when μ and the beamwidth parameter z_0 are fixed. In other words, the sidelobe tapering effect of a given value of μ depends on n, unless $\mu = 0$. Numerical examples bear out the $n^{-\mu}$ dependence in (39).

An important observation based on (35) and (39) is that for $\mu < 0$ the sidelobes may well steadily increase as μ approaches endfire. That this is in fact the case is borne out by the examples given later.

It should be emphasized that although the Gegenbauer weights must be positive if $0 \leq \mu \leq \mu^*$, they might not necessarily be positive if $\mu < 0$ or if $\mu > \mu^*$. For $\mu < 0$ it can happen that all are positive, or that some are negative. Only numerical computation can show which is the case. If some of the weights are negative, it becomes a possibility that the maximum response might not occur for $u = 0$.

For the Gegenbauer weights it is readily shown that a sufficient condition for the MRA to be at $u = 0$ is that $C_n^\mu(x)$ attain its maximum over the interval $[-1, 1]$ at $x = 1$. By a well known result [1, eq. (7.33.1)] the maximum of $C_n^\mu(x)$ occurs at $x = 1$ if and only if $\mu > 0$. Thus, a *sufficient* condtion for $u = 0$ to be the MRA is that $\mu \geq 0$. For $\mu < 0$ the MRA depends on the size of z_μ and must be verified numerically. From [1, eq. (7.33.1)] the maximum of $C_n^\mu(x)$ occurs at or near $x = 0$ when $\mu < 0$; therefore, if the MRA is not at $u = 0$, then the MRA must be at or near endfire. This observation is rendered quite reasonable when considered in the light of the examples presented later. This author has never experienced a case where the MRA was not $u = 0$ for $\mu > -1/2$ and reasonable values of z_μ.

It would be interesting to know how much energy is contained in the main lobe of a Gegenbauer design. From (20) and (30), this requires a tractable form for the integral

$$\int_0^{u_0} [C_n^\mu(z_\mu \cos(\tfrac{1}{2}\pi u))]^2 \, du \tag{40}$$

which we do not have. On the other hand, the total "weighted" energy contained in all of the sidelobes is the smallest possible

for $\mu > -1/2$. Specifically, if π_{n-1} denotes a polynomial of degree at most $n-1$, then [1, eq. (4.7.15)] for $\mu > -1/2$

$$\min_{\pi_{n-1}} \int_{-1}^{+1} (1-x^2)^{\mu-1/2} [x^n - \pi_{n-1}(x)]^2 \, dx = \frac{2^{1-2\mu}\pi}{[\Gamma(\mu)]^2} \frac{\Gamma(n+2\mu)}{(n+\mu)\Gamma(n+1)}, \quad (\mu \neq 0). \tag{41}$$

Furthermore, if $\tilde{\pi}_{n-1}(x)$ is the minimizing polynomial, then [1, eq. (4.7.9)]

$$C_n^{\mu}(x) = 2^n \binom{n+\mu-1}{n} (x^n - \tilde{\pi}_{n-1}(x)).$$

Substituting $x = z_\mu \cos(\pi t/2)$ thus establishes our claim. However, a problem with this formulation is that part of the main lobe energy is included in the total weighted sidelobe energy. The reason is that the x-interval $[x_n^{(\mu)}, +1]$ is transformed [use (12) and (30)] to the u-interval

$$\left[\frac{2}{\pi} \arccos\left(\frac{1}{z_0 x_n^{(\mu)}}\right) \cos\left(\frac{\pi}{2n}\right), u_0\right] \tag{42}$$

which is a subset of the main lobe region. For the Dolph-Chebyshev case $\mu = 0$, this u-interval goes from the first null up to the point on the main lobe equal to the overall sidelobe level and, so, is not considerable. For larger values of μ, this u-interval grows larger because of (27) and thus contributes progressively more significant portions to the weighted sidelobe energy estimate.

III. Gegenbauer Weights for a Continuous Aperture

The Gegenbauer weights derived for the discrete finite aperture have a limiting form as $n \to \infty$ with total aperture length $2L$ held constant. This is essentially the high-frequency limit of the weights as functions of design frequency. The limiting form is a continuous real-valued function defined on the whole aperture and must be nonnegative if $0 < \mu < \mu^*$. The case $\mu = 0$ develops δ-function spikes at the aperture endpoints; i.e., the case $\mu = 0$ gives the van der Maas function. For $\mu > \mu^*$ the limit is still continuous, but we cannot guarantee by simple inspection that it is nonnegative across the entire aperture. For $\mu < 0$, the integral (60) below diverges.

Let the continuous aperture be taken to be the closed interval $[-L, L]$ on the x-axis. Rewriting (3) gives

$$F(u) = \int_1^N W_0(x) \exp(-i\pi x u) \, dx \tag{43}$$

where

$$W_0(x) \triangleq \sum_{k=1}^{N} w_k \, \delta(x-k). \tag{44}$$

(The integral in (43) includes *all* of the impulses at 1 and N.) Scaling the interval $[1, N]$ to the given aperture $[-L, L]$ and using the fact that the weights $\{w_k\}_1^N$ are symmetric gives

$$G(v) = 2 \int_0^L W(\zeta) \cos(\zeta v) \, d\zeta \tag{45}$$

where

$$n = N - 1 \tag{46}$$

$$x = \frac{n\zeta}{2L} + \frac{n+2}{2} \tag{47}$$

$$W(\zeta) = \frac{n}{2L} \sum_{k=1}^{N} w_k \, \delta\left[\zeta - \left(k - 1 - \frac{n}{2}\right)\frac{2L}{n}\right] \tag{48}$$

$$v = \frac{\pi n u}{2L} \tag{49}$$

$$G(v) = e^{i\pi(n+2)u/2} F(u). \tag{50}$$

From (20), for the Gegenbauer weights,

$$e^{i2L(n+2)v/n} G(v) = C_n^{\mu}\left(z_\mu \cos\left(\frac{Lv}{n}\right)\right). \tag{51}$$

In order to take the limit in (51) as $n \to \infty$, we need to establish the asymptotic behavior

$$z_\mu \cong \sec\left(\frac{L\tau'}{n}\right), \quad n \to \infty \tag{52}$$

$$\tau' \triangleq \frac{\tau}{L} \tag{53}$$

where τ is defined by (38). The proof uses the asymptotic results

$$z_0 = \cosh\left(\frac{1}{n} \operatorname{arccosh} r\right) \tag{54}$$

$$\cong 1 + \frac{(\operatorname{arccosh} r)^2}{2n^2}, \quad n \to \infty$$

$$\cong \sec\left(\frac{1}{n} \operatorname{arccosh} r\right), \quad n \to \infty \tag{55}$$

and

$$x_n^{(\mu)} \cong \cos\left(\frac{j_{\mu-1/2}}{n}\right), \quad n \to \infty. \tag{56}$$

Apparently (54) was first given in [6]; it follows directly from the definition of the Chebyshev polynomials and the fact that $r > 1$. On the other hand, (56) follows from the Mehler-Heine result, (A2) of the Appendix, by specializing it to the Gegenbauer polynomials using [1, eq. (4.7.1)]. Now, from (30),

$$z_\mu \cong \sec\left(\frac{1}{n} \operatorname{arccosh} r\right) \cos\left(\frac{j_{\mu-1/2}}{2}\right) \sec\left(\frac{\pi}{2n}\right), \quad n \to \infty$$

$$\cong \sec\left(\frac{L\tau'}{n}\right), \quad n \to \infty$$

with τ' defined by (53). We point out that if τ' is pure imaginary, then the hyperbolic secant can replace the secant in (52). The possibility of imaginary τ' does not affect the validity of the following argument.

Finally, from (51), normalizing by the factor $n^{1-2\mu}/(2\mu)$ to

keep $G(v)$ bounded gives

$$H(v) \triangleq \lim_{n\to\infty} \frac{n^{1-2\mu}}{2\mu} e^{j2L(n+2)v/n} G(v)$$

$$= \lim_{n\to\infty} \frac{n^{1-2\mu}}{2\mu} C_n^{\mu}\left(\frac{\cos\dfrac{Lv}{n}}{\cos\dfrac{L\tau'}{n}}\right) \tag{57}$$

$$= \frac{\sqrt{\pi/2}}{2^{\mu}\Gamma(\mu+1)} \frac{J_{\mu-1/2}(L\sqrt{v^2-\tau'^2})}{(L\sqrt{v^2-\tau'^2})^{\mu-1/2}} \tag{58}$$

where (58) is merely (A6) of the Appendix. Thus, (58) gives the beam pattern of the continuous Gegenbauer weighting function on the interval $[-L, L]$. The first null of $H(v)$ is

$$v_0 = \frac{1}{L}\left[(\operatorname{arccosh} r)^2 + \pi^2/4\right]^{1/2} \tag{59}$$

which is derived from (58) by using (53). Note that v_0 is independent of μ because of (30).

The beam pattern (58) is easier to derive than the continuous Gegenbauer weighting function. Although one can find the Fourier transform of (58) as a special case of Sonine's second finite integral, (A1), the assertion that this transform is indeed the limit of the Gegenbauer weights for a discrete aperture requires a separate proof. Conceivably the Gegenbauer weights might diverge even though the limit (58) exists. This in fact happens only for $\mu \leq 0$. The proof constitutes about half the attention of the Appendix: see especially (A8), (A22), (A26), (A27), and (A29). The final answer can be found by specializing (A29), using (A25), to yield

$$H(v) = \frac{1}{2^{\mu}\Gamma(\mu+1)(L\tau')^{\mu-1}} \int_0^1 (\sqrt{1-\xi^2})^{\mu-1}$$

$$J_{\mu-1}(L\tau'\sqrt{1-\xi^2}) \cos Lv\xi \, d\xi. \tag{60}$$

The continuous Gegenbauer weighting function on the aperture is obvious on setting $\zeta = L\xi$. The continuous Gegenbauer function depends on the parameter μ, which we must restrict to $\mu \geq 0$ for the integral to converge [see (A23)]. It also depends on the beamwidth parameter z_0 through the variable τ' defined by (53).

The Kaiser-Bessel window is a special case of (60), as is easily seen by setting $\mu = 1$. Since the Gegenbauer polynomials $C_n^{\mu}(x)$ for $\mu = 1$ are, from (2), the Chebyshev polynomials of the second kind, it is clear that Kaiser-Bessel must be their continuous analog. Also, our claim that the van der Maas weighting function is a limiting case of (60) as $\mu \to 0$ can be seen from

$$\lim_{\mu\to 0+} x^{\mu-1} J_{\mu-1}(x) = \frac{J_1(x)}{x} + 2\delta(x). \tag{61}$$

Substituting $L\tau'\sqrt{1-\xi^2}$ for x in (61) and then substituting in (60) yields the van der Maas function. The result (61) was pointed out to the author by A. H. Nuttall in a private communication [7] while the present paper was being drafted.

Note that the beam pattern function (58) is a well-defined function of v for all real and complex values of μ (in fact, it is an entire function of v for all μ) so that it can be computed and inspected in the absence of any corresponding weighting function. In particular, for negative μ the beam pattern function (58) grows with increasing v just as might be expected from the discrete aperture case. However, the beam pattern (58) for $\mu \leq 0$ is not realizable as the cosine transform of a continuous function on the closed interval, or aperture, $[-L, L]$.

IV. Examples

The five examples presented here are for the discrete aperture with 100 elements at a half wavelength spacing and steered broadside. The half beamwidth, measured from the MRA to the first null, is 2.565588° and is the same for all five examples. This is accomplished by defining z_{μ} as in (30) and computing it in the manner described in detail in Section II, (23)–(28). The remaining free parameter, μ, we take equal to 0.4, 0.2, 0.0, -0.2, -0.4, successively. The Gegenbauer weights are computed in the suggested form (15), and the resulting beam patterns for these five values of μ are given in Figs. 1–5, respectively. The independent variable in these patterns is the angle θ_a, not u; the vertical axis is $20 \log_{10} |F(\sin\theta_a)|$.

Perhaps the most prominent feature of these five beam patterns is that the sidelobe structure for a fixed positive value of μ is "reciprocal" to that for $-\mu$. Consider $\mu = \pm 0.4$, for instance. If the reader takes a Xerox of both beam patterns and turns one of them upside down on top of the other (literally) and holds the pair up to the light, then it will be abundantly clear what "reciprocal" means in this context. The cause of this attractive matching of sidelobe envelopes is that the bound (35) is, in fact, very reflective of true sidelobe taper. Thus, for positive μ the sidelobes decay, while for negative μ the sidelobes grow. For $\mu = 0$ the sidelobes neither grow nor decay; they remain constant. The case $\mu = 0$ is, of course, the Dolph-Chebyshev design. The author has not undertaken any further studies to determine the accuracy of the sidelobe envelope factor.

Another important feature is that the first sidelobe *alone* seems to be extremely important in determining the possible size of the remaining sidelobes. Although this is not a rigorous statement, it does seem to be borne out by these examples. For $\mu = 0.2$ the first sidelobe is increased by about 1 dB to -29 dB, the second sidelobe seems unchanged at -30 dB, and all the remaining sidelobes are uniformly (and progressively) lower than the -30 dB Dolph-Chebyshev case ($\mu = 0$) with the last sidelobe depressed about 34 dB. Similar but "reciprocal" remarks hold for the $\mu = -0.2$ case. For $\mu = 0.4$ ($\mu = -0.4$) the second sidelobe is slightly higher (lower) than -30 dB, but the point made here is still substantially true.

The weights for the cases $\mu = 0.4, 0.2$, and 0.0 are all positive. For the cases -0.2 and -0.4, the only negative weights corresponded to the elements adjacent to the end elements.

All five examples have 49 sidelobes on either side of the MRA. This can be attributed to the fact that the Gegenbauer polynomial $C_n^{\mu}(x)$ has all its n zeros in the open interval $(-1, +1)$ when $\mu > -1/2$. Thus, from (20), $F(u)$ must have $N - 1 = 99$ zeros in the open u interval $(0, 2)$. By Rolle's theorem of elementary calculus, $F(u)$ must have 98 points (i.e., sidelobe peaks) interior to $(0, 2)$ where $|F'(u)| = 0$. Since $|F(u)|$ is an even func-

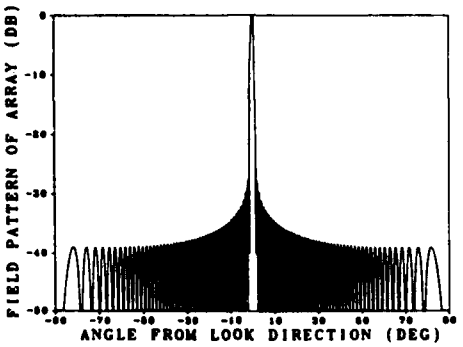


Fig. 1. Gegenbauer 100 element array; $\mu = 0.4$; first null = 2.565588°.

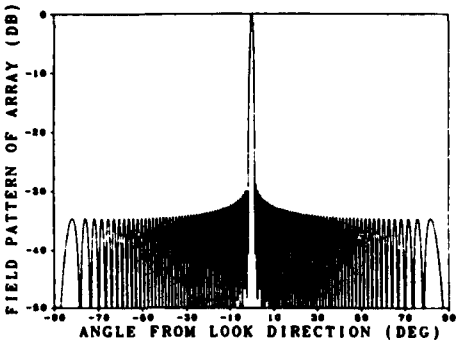


Fig. 2. Gegenbauer 100 element array; $\mu = 0.2$, first null = 2.565588°.

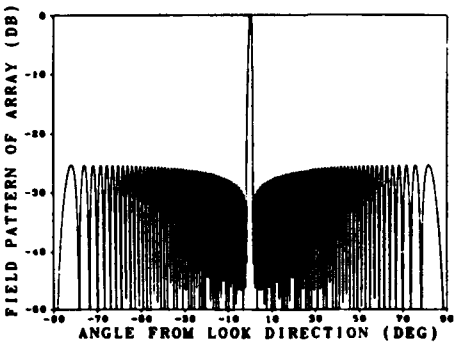


Fig. 4. Gegenbauer 100 element array; $\mu = -0.2$; first null = 2.565588°.

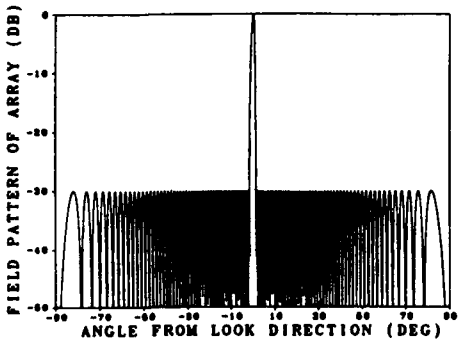


Fig. 3. Gegenbauer 100 element array; $\mu = 0$; first null = 2.565588°. (This is classic Dolph–Chebyshev.)

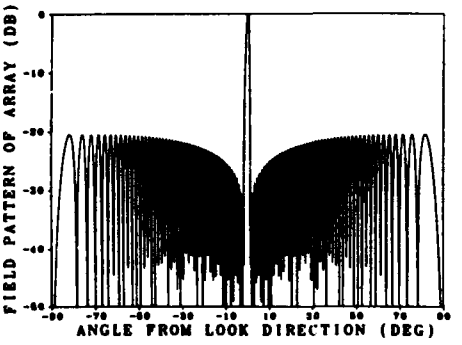


Fig. 5. Gegenbauer 100 element array; $\mu = -0.4$; first null = 2.565588°.

tion of u, half of these sidelobes must be on each side of the MRA.

All five examples exhibit a plateau in the decay, or growth, of sidelobes at sufficiently great distances from the MRA. This feature is also an artifact of the sidelobe envelope factor (35).

Taken together, these examples indicate that the ratio (39) is, on a log plot, roughly linear in μ for fixed n and beamwidth parameter z_0. Whether this linearity is true only for reasonably small values of μ has not been determined. A careful mathematical proof of approximate μ linearity of the logarithm of (39) would be nice to have.

V. Discussion and Summary

The Gegenbauer weighting functions for the discrete and continuous aperture, as well as for nonrecursive digital filters, permits the designer to maintain a fixed specified beamwidth as

defined via (30) while scanning continuously in μ to discriminate against spatially distributed noise sources and/or extraneous signals by tapering the sidelobes. The required weights can be calculated quickly and accurately by the analytic formulas provided here; hence, it might be possible to choose μ adaptively to achieve some objective such as maximizing signal-to-noise ratio. The beam patterns for negative μ are particularly interesting in that it may be possible to discriminate against noise sources that lie nearby (in bearing) the desired signal source, and thereby enhance tracking capability.

One advantage of the Gegenbauer weights is that they are derived for a discrete aperture exactly, and the continuous aperture weighting function is then discovered as their limit. If only a continuous aperture function is defined, then it must be sampled at a finite set of points in any application to a discrete aperture. How this sampling is best done is not commonly discussed, and it leaves a certain ambiguity in the discrete aperture weights. The discrete Gegenbauer weights given by (18) and (19) above do not have this problem.

When steering a Gegenbauer array design, no different problems should arise than what is normally expected in the usual Dolph-Chebyshev design. Gegenbauer designs can be steered nearly to endfire before encountering the first grating lobe.

A difference beam pattern can be constructed from the Gegenbauer weights in the usual way of changing the signs of the weights on one-half of the array. If this is done, the difference beam pattern is proportional to $|C_n^{\mu}(z_{\mu} \sin(\pi u/2))|$. This is easy to show from the constructions (18)-(20). The result is a beam pattern with a null at $u = 0$.

All the nulls of the Gegenbauer beam pattern seem to shift strictly away from the MRA as μ increases. This effect is evident in the examples. It is quite possible to use this effect to deliberately control null placement to cancel localized noise sources. A mathematical proof that the nulls must shift in this manner requires knowledge of the relative size of the derivatives (with respect to μ) of all of the zeros of $C_n^{\mu}(x)$. Although this information is not known to the author, it is not really necessary to have it in order to utilize the null shifting effect in practice.

The Gegenbauer weights for discrete and continuous apertures was derived by the author between March and May 1981. The mathematical results contained in the Appendix first appeared in [11].

Appendix

Mathematical Derivations and Results

Sonine's second finite integral [8, p. 376] may be written

$$\int_0^{\pi/2} J_{\mu}(x \sin\theta) J_{\lambda}(y \cos\theta) \sin^{\mu+1}\theta \cos^{\lambda+1}\theta \, d\theta = \frac{x^{\mu} y^{\lambda} J_{\mu+\lambda+1}(\sqrt{x^2+y^2})}{(\sqrt{x^2+y^2})^{\mu+\lambda+1}} \tag{A1}$$

for all complex x and y, and is valid when both $\mathrm{Re}(\mu) > -1$ and $\mathrm{Re}(\lambda) > -1$. At least three proofs of this result are known. One involves expanding the integral in powers of x and y; another involves integration over subsets of the surface of the unit sphere in R^3. Both are given in [8]. The third proof using the generalized Laguerre polynomials $L_n^{(\alpha)}(x)$ is mentioned in [12].

For the case of real μ and λ, a fourth proof is given here that depends in an essential way on the identity (A7). In this connection, the particular form of the coefficients $a_{k,n}(y)$ is important; that is, the easily derived identity (A10) does not seem to be all useful, but the identity (A8) is exactly what is needed. It facilitates the investigation of the limiting form (A27) of $a_{k,n}(y)$ as n tends to infinity. The identity (A8) is apparently new; however, the special case of $y = 1$ was known to Gegenbauer.

Equation (A8) is interesting in another regard as well. A simple inspection suffices to prove that $a_{k,n}(y) > 0$ for all n and k whenever $y > 1$ and $\mu \geq \lambda > 0$. The coefficients remain positive in the two limiting cases $\mu > 0$, $\lambda = 0$ and $\mu = \lambda = 0$, as can be seen from (A18)-(A21). In fact, it was only this positivity result that the author originally sought.

The result (A3) of the Mehler-Heine type is apparently new. It is needed to prove (A1) by our methods. It has additional interest in that it duplicates the result given by Szegö (A2) simply by setting $y = 0$. Mathematically, however, (A2) and (A3) are equivalent. The special cases (A4a) and (A4b) involving Chebyshev polynomials are particularly striking.

Let α and β be arbitrary real numbers. For any complex number x, the Mehler-Heine theorem states that

$$\lim_{n\to\infty} n^{-\alpha} P_n^{(\alpha,\beta)}\left(\cos\frac{x}{n}\right) = (x/2)^{-\alpha} J_{\alpha}(x) \tag{A2}$$

where $J_{\alpha}(x)$ is the Bessel function of the first kind of order α [1, eq. (1.71.1)], [2, sect. 3.1(8)]. A straightforward proof of (A2) can be found in Szegö [1, Theorem 8.1.1]. Szegö's proof can be readily modified to show that

$$\lim_{n\to\infty} n^{-\alpha} P_n^{(\alpha,\beta)}\left(\frac{\cos\frac{x}{n}}{\cos\frac{y}{n}}\right) = \left(\tfrac{1}{2}\sqrt{x^2-y^2}\right)^{-\alpha} J_{\alpha}\left(\sqrt{x^2-y^2}\right) \tag{A3}$$

for all complex x and y. Like the Mehler-Heine result, this formula holds uniformly for x and y in every bounded region of the complex plane. The special case $\alpha = \beta = -1/2$ gives the interesting result

$$\lim_{n\to\infty} T_n\left(\frac{\cos\frac{x}{n}}{\cos\frac{y}{n}}\right) = \cos\sqrt{x^2-y^2} \tag{A4a}$$

where $T_n(x)$ is the Chebyshev polynomial of the first kind [1, eq. (4.1.7)], while the special case $\alpha = \beta = 1/2$ gives

$$\lim_{n\to\infty} n^{-1} U_n\left(\frac{\cos\frac{x}{n}}{\cos\frac{y}{n}}\right) = \frac{\sin\sqrt{x^2-y^2}}{\sqrt{x^2-y^2}} \tag{A4b}$$

where $U_n(x)$ is the Chebyshev polynomial of the second kind [1, eq. (4.1.7)]. These follow from (A3) by using Stirling's

formula and the well-known results [1, eq. (1.71.2)]

$$J_{-1/2}(z) = \left(\frac{2}{\pi z}\right)^{1/2} \cos z, \quad J_{1/2}(z) = \left(\frac{2}{\pi z}\right)^{1/2} \sin z. \tag{A5}$$

We will need another special case of the general result; specifically, for $\mu > -1$,

$$\lim_{n\to\infty} \frac{n^{1-2\mu}}{2\mu} C_n^{\mu}\left(\frac{\cos\frac{x}{n}}{\cos\frac{y}{n}}\right)$$

$$= \sqrt{\pi/2}\, \frac{J_{\mu-1/2}(\sqrt{x^2-y^2})}{2^{\mu}\Gamma(\mu+1)(\sqrt{x^2-y^2})^{\mu-1/2}} \tag{A6}$$

where $C_n^{\mu}(x)$ are the ultraspherical, or Gegenbauer, polynomials [1, eq. (4.7.1)]. (Szegö uses the notation $P_n^{(\mu)}(x)$ instead of $C_n^{\mu}(x)$.)

We derive Sonine's second finite integral by finding an alternate form for the left-hand side of (A6). This requires the following result. For $\mu \geq \lambda > 0$, the coefficients $a_{k,n}(y)$ in the expansion

$$C_n^{\mu}(xy) = \sum_{k=0}^{\lfloor n/2 \rfloor} a_{k,n}(y)\, C_{n-2k}^{\lambda}(x), \quad n = 0, 1, 2, \cdots \tag{A7}$$

are given explicitly by

$$a_{k,n}(y) = (n - 2k + \lambda)(\mu)_{n-k} \cdot \sum_{m=0}^{k} \frac{(\mu - \lambda + m)_{k-m}\,(y^2-1)^m\, y^{n-2m}}{m!\,(k-m)!\,(\lambda)_{n-k-m+1}} \tag{A8}$$

where we take $0^0 = 1$ and $(0)_0 = 1$ whenever they occur. Setting $y = 1$ in (8) gives

$$a_{k,n}(1) = \frac{(n - 2k + \lambda)(\mu)_{n-k}\,(\mu-\lambda)_k}{k!\,(\lambda)_{n-k+1}} \tag{A9}$$

which is due to Gegenbauer [1, eq. (4.10.27)]. Furthermore, for real $y > 1$ and $\mu \geq \lambda > 0$, the coefficients $a_{k,n}(y)$ are all positive as can be seen by inspection in (A8).

The formula (A.8) is derived as follows. Let $\mu \geq \lambda > 0$. In the expression [1, eq. (4.7.31)]

$$C_n^{\mu}(x) = \sum_{m=0}^{\lfloor n/2 \rfloor} (-1)^m \frac{(\mu)_{n-m}}{m!\,(n-2m)!}\,(2x)^{n-2m}$$

we replace x with xy, substitute

$$\frac{(2x)^{n-2m}}{(n-2m)!} = \sum_{s=0}^{\lfloor (n-2m)/2 \rfloor} \frac{(n-2m+\lambda-2s)}{s!\,(\lambda)_{n-2m-s+1}}\, C_{n-2m-2s}^{\lambda}(x),$$

and collect terms to get

$$a_{k,n}(y) = \sum_{m=0}^{k} (-1)^m \frac{(n-2k+\lambda)(\mu)_{n-m}\, y^{n-2m}}{m!\,(k-m)!\,(\lambda)_{n-m-k+1}} \tag{A10}$$

$$= y^{n-2k}(n-2k+\lambda)\, Q_k(2y^2-1) \tag{A11}$$

where Q_k is a polynomial defined for general complex argument u by

$$Q_k(u) = \sum_{m=0}^{k} \frac{(-1)^m (\mu)_{n-m}}{m!\,(k-m)!\,(\lambda)_{n-m-k+1}} \cdot \left(\frac{u+1}{2}\right)^{k-m}. \tag{A12}$$

For arbitrary α and β, the Jacobi polynomial of degree $k \geq 0$ can be written

$$P_k^{(\alpha,\beta)}(u) = \sum_{m=0}^{k} (-1)^m \frac{(k+\alpha+\beta+1)_{k-m}\,(k-m+\beta+1)_m}{m!\,(k-m)!} \cdot \left(\frac{u+1}{2}\right)^{k-m} \tag{A13}$$

which follows from [1, eq. (4.21.2)] using the identity [1, eq. (4.1.3)]. Setting $\alpha = \mu - \lambda - 1$ and $\beta = \lambda + n - 2k$ in (A13) shows that

$$Q_k(u) = \frac{(\mu)_{n-k}}{(\lambda)_{n-k+1}}\, P_k^{(\mu-\lambda-1,\lambda+n-2k)}(u) \tag{A14}$$

Expanding the Jacobi polynomial in (A14) using [1, eq. (4.3.2)]

$$P_k^{(\alpha,\beta)}(u) = \sum_{m=0}^{k} \frac{(1+\alpha)_k\,(1+\beta)_k}{m!\,(k-m)!\,(1+\alpha)_m\,(1+\beta)_{k-m}} \cdot \left(\frac{u-1}{2}\right)^m \left(\frac{u+1}{2}\right)^{k-m} \tag{A15}$$

and substituting $u = 2y^2 - 1$ gives

$$Q_k(2y^2-1) = (\mu)_{n-k} \sum_{m=0}^{k} \cdot \frac{(\mu-\lambda+m)_{k-m}\,(y^2-1)^m\, y^{2k-2m}}{m!\,(k-m)!\,(\lambda)_{n-k-m+1}}. \tag{A16}$$

Thus, (A16) and (A11) establish (A8).

Two limiting cases of (A7) are easily derived from [1, eq. (4.7.8)]

$$\lim_{\lambda\to 0} \frac{n}{2\lambda}\, C_n^{\lambda}(x) = T_n(x), \quad n \geq 1 \tag{A17}$$

and are worth recording. Thus, for $\mu > 0$,

$$C_n^{\mu}(xy) = \sum_{k=0}^{\lfloor n/2 \rfloor}{}' b_{k,n}(y)\, T_{n-2k}(x), \quad n = 0, 1, 2, \cdots \tag{A18}$$

where

$$b_{k,n}(y) = 2(\mu)_{n-k} \sum_{m=0}^{k} \frac{(\mu+m)_{k-m}\,(y^2-1)^m\, y^{n-2m}}{m!\,(k-m)!\,(n-k-m)!} \tag{A19}$$

and

$$T_n(xy) = \sum_{k=0}^{\lfloor n/2 \rfloor}{}' c_{k,n}(y)\, T_{n-2k}(x), \qquad n = 1, 2, 3, \cdots \tag{A20}$$

where

$$c_{k,n}(y) = n(n-k-1)! \sum_{m=0}^{k} \frac{(m)_{k-m}(y^2-1)^m y^{n-2m}}{m!\,(k-m)!\,(n-k-m)!}. \tag{A21}$$

The notation Σ' means that 1/2 of the last term in the sum is taken if n is even, and all of it is taken if n is odd. Note that inspection shows that $y > 1$ implies that $b_{k,n}(y)$ and $c_{k,n}(y)$ are positive.

Sonine's second finite integral is now derived from (A6). Fix x and y. Let $N = [n/2]$. From (A7)

$$\frac{n^{1-2\mu}}{2\mu} C_n^{\mu}\left(\frac{\cos\frac{x}{n}}{\cos\frac{y}{n}}\right) = \frac{1}{1+N}\sum_{k=0}^{N} \left\{\frac{\lambda n^{1-2\mu}(1+N)}{\mu(n-2k)^{1-2\lambda}} a_{k,n}\left(\frac{1}{\cos\frac{y}{n}}\right)\right\}\left\{\frac{(n-2k)^{1-2\lambda}}{2\lambda} C_{n-2k}^{\lambda}\left(\cos\frac{x}{n}\right)\right\} = \int_0^1 f_n(1-\zeta) g_n(1-\zeta)\, d\zeta \tag{A22}$$

where we have defined for $0 \leqslant \zeta \leqslant 1$

$$f_n(1-\zeta) = \sum_{k=0}^{N} \frac{\lambda n^{1-2\mu}(1+N)}{\mu(n-2k)^{1-2\lambda}} a_{k,n}\left(\frac{1}{\cos\frac{y}{n}}\right) \chi_{E_k}(1-\zeta)$$

$$g_n(1-\zeta) = \sum_{k=0}^{N} \frac{(n-2k)^{1-2\lambda}}{2\lambda} C_{n-2k}^{\lambda}\left(\cos\frac{x}{n}\right) \chi_{E_k}(1-\zeta)$$

and χ_{E_k} is the characteristic (indicator) function of the interval

$$E_k = \begin{cases} \left[\frac{k}{N+1}, \frac{k+1}{N+1}\right), & k = 0, 1, \cdots, N-1 \\ \left[\frac{k}{N+1}, \frac{k+1}{N+1}\right], & k = N. \end{cases}$$

It can be verified that $\chi_{E_k}(2k/n) = 1$ for $k = 0, 1, \cdots, N$.

Assume for the moment that both $|f_n(\zeta)|$ and $|g_n(\zeta)|$ are bounded above by integrable functions of ζ. To do this, it will be seen that we must restrict attention to $\lambda > -1/2$, $\mu > -1/2$, $\mu > \lambda$, so that the integral [1, eq. (1.7.4)]

$$\int_0^1 \zeta^{2\lambda}(1-\zeta^2)^{\mu-\lambda-1}\, d\zeta = \frac{\Gamma(\lambda+\frac{1}{2})\Gamma(\mu-\lambda)}{2\Gamma(\mu+\frac{1}{2})} \tag{A23}$$

will be finite. If $f = \lim f_n$ and $g = \lim g_n$, the bounded convergence theorem [9, p. 110] implies

$$\lim_{n\to\infty} \frac{n^{1-2\mu}}{2\mu} C_n^{\mu}\left(\frac{\cos\frac{x}{n}}{\cos\frac{y}{n}}\right) = \int_0^1 f(1-\zeta) g(1-\zeta)\, d\zeta. \tag{A24}$$

Let ζ in (0, 1) be rational. Then $1 - \zeta = 2k/n$ for sufficiently large k and n, so that

$$g(1-\zeta) = \lim_{n\to\infty} g_n(1-\zeta) = \lim_{\substack{n\to\infty \\ 1-\zeta = 2k/n}} \frac{(n-2k)^{1-2\lambda}}{2\lambda} C_{n-2k}^{\lambda}\left(\cos\frac{x}{n}\right) = \sqrt{\pi/2}\, \frac{J_{\lambda-1/2}(\zeta x)}{2^{\lambda}\Gamma(\lambda+1)(\zeta x)^{\lambda-1/2}} \tag{A25}$$

with the last step following immediate from (A6). Thus, (A25) holds for all ζ in [0, 1] by continuity. Similarly, from (A8) and for all ζ rational in (0, 1),

$$f(1-\zeta) = \lim_{n\to\infty} f_n(1-\zeta) = \lim_{\substack{n\to\infty \\ 1-\zeta = 2k/n}} \frac{\lambda n^{1-2\mu}(1+N)}{\mu(n-2k)^{1-2\lambda}} a_{k,n}\left(\frac{1}{\cos\frac{y}{n}}\right)$$

$$= \lim_{\substack{n\to\infty \\ 1-\zeta = 2k/n}} \sum_{m=0}^{k} \frac{\left(\zeta+\frac{\lambda}{n}\right)(\mu+1)_{n-k-1}\, n^{-2(\mu-\lambda-1)}\left(\frac{1}{n}+\frac{N}{n}\right)(\mu-\lambda+m)_{k-m}\sin^{2m}\frac{y}{n}}{\zeta^{1-2\lambda}\cos^{n}\frac{y}{n}\, m!\,(k-m)!\,(\lambda+1)_{n-k-m}} \tag{A26}$$

Interchange the limit and the summation, and evaluate the limit of the mth term (convert Pochhammer symbols to gamma functions, apply Stirling's formula, and use $k(n-k) = (1-\zeta^2)n^2/4$) to obtain

$$f(1-\zeta) = \sum_{m=0}^{\infty} \frac{\zeta^{2\lambda}(1-\zeta^2)^{\mu-\lambda-1}}{2^{2\mu-2\lambda-1}} \frac{\Gamma(\lambda+1)}{\Gamma(\mu+1)} \cdot \frac{(\frac{1}{2}y\sqrt{1-\zeta^2})^{2m}}{m!\,\Gamma(\mu-\lambda+m)} = \frac{\zeta^{2\lambda}(1-\zeta^2)^{\mu-\lambda-1}}{2^{2\mu-2\lambda-1}} \frac{\Gamma(\lambda+1)}{\Gamma(\mu+1)} \cdot \frac{I_{\mu-\lambda-1}(y\sqrt{1-\zeta^2})}{(\frac{1}{2}y\sqrt{1-\zeta^2})^{\mu-\lambda-1}} \tag{A27}$$

where $I_{\nu}(z)$ denotes the modified Bessel function of the first order ν (see [8, sect. 3.7(2)]). We must require $\mu > \lambda$ in (A27) to have convergence. Continuity again assures that (A27) holds for all ζ in (0, 1). Now, interchanging the limit and the sum was valid because an upper bound for the total sum can be found. Since the absolute value of the mth term in (A26) is bounded by

$$B \frac{\Gamma(\lambda+1)}{\Gamma(\mu+1)} \frac{(\frac{1}{2}|y|\sqrt{1-\zeta^2})^{2m}}{m!\,\Gamma(\mu-\lambda+m)}$$

where

$$B = \frac{\zeta + \frac{\lambda}{n}}{\zeta^{1-2\lambda}} \frac{n^{-2(\mu-\lambda-1)}}{n^{2m}\left|\cos^n \frac{y}{n}\right|} \left(\frac{1-\zeta^2}{4}\right)^{-m}$$

$$\cdot \frac{\Gamma(k+\mu-\lambda)}{\Gamma(k-m+1)} \frac{\Gamma(n-k+\mu)}{\Gamma(n-k+\lambda+1-m)}$$

$$\cong \zeta^{2\lambda} \left(\frac{1-\zeta^2}{4}\right)^{\mu-\lambda-1}, \quad n \to \infty,$$

the total sum in (A26) is bounded by

$$F(\zeta) = L\zeta^{2\lambda}(1-\zeta^2)^{\mu-\lambda-1} \frac{\Gamma(\lambda+1)}{\Gamma(\mu+1)}$$

$$\cdot \sum_{m=0}^{\infty} \frac{(\frac{1}{2}|y|\sqrt{1-\zeta^2})^{2m}}{m!\,\Gamma(\mu-\lambda+m)} < \infty \tag{A28}$$

for some constant L independent of ζ. The *series* in (A28) is a continuous function of ζ on $[0, 1]$ if $\mu > \lambda$. Hence, from (A23), $F(\zeta)$ is an integrable function that bounds $|f_n(\zeta)|$ for all n.

From (A24), (A25), and (A27) we have

$$\lim_{n\to\infty} \frac{n^{1-2\mu}}{2\mu} C_n^{\mu}\left(\frac{\cos\frac{x}{n}}{\cos\frac{y}{n}}\right)$$

$$= \frac{\sqrt{\pi/2}}{2^{\mu}\Gamma(\mu+1)x^{\lambda-1/2}y^{\mu-\lambda-1}}$$

$$\cdot \int_0^1 \zeta^{\lambda+1/2}(\sqrt{1-\zeta^2})^{\mu-\lambda-1}$$

$$\cdot J_{\lambda-1/2}(\zeta x) I_{\mu-\lambda-1}(y\sqrt{1-\zeta^2})\, d\zeta$$

$$= \frac{\sqrt{\pi/2}}{2^{\mu}\Gamma(\mu+1)} \frac{J_{\mu-1/2}(\sqrt{x^2-y^2})}{(\sqrt{x^2-y^2})^{\mu-1/2}} \tag{A29}$$

with the last equation from (A6). Substituting $\zeta = \sin\theta$ and $y = iy'$ in the last two formulas, and setting

$$\mu' = \lambda - \tfrac{1}{2} > -1 \quad \text{and} \quad \lambda' = \mu - \lambda - 1 > -1 \tag{A30}$$

yields Sonine's second finite integral (A1). The only thing left to prove is that $|g_n(\zeta)|$ is bounded by an integrable function on $[0, 1]$. Szegö's argument [1, p. 192] in the proof of (A2) can be modified easily to show $|g_n(\zeta)|$ is bounded by a constant.

The proof of (A1) presented here was intentionally restricted to real μ and λ. However, it is not hard to see from (A23) and (A30) that the proof can be carried out for complex μ and λ, provided appropriate remarks are made in appropriate places about the complex case. If such remarks are made, our derivation proves (A1) for $\mathrm{Re}(\mu) > -1$ and $\mathrm{Re}(\lambda) > -1$. Divergence of (A23) is seen to be the *cause* of the restrictions on μ and λ.

The material contained in this Appendix was first documented in [11].

References

[1] G. Szego, *Orthogonal Polynomials*, 4th ed. Amer. Math. Soc. Colloquium Pub., vol. 23, 1978.

[2] C. L. Dolph, "A current distribution for broadside arrays which optimizes the relationship between beam width and side-lobe level," *Proc. IRE and Waves and Electrons*, vol. 34, pp. 335-348, June 1946.

[3] G. J. van der Maas, "A simplified calculation for Dolph-Tchebycheff arrays," *J. Appl. Phys.*, vol. 25, Jan. 1954.

[4] J. F. Kaiser, "Digital filters," in *System Analysis by Digital Computer*, F. F. Kuo and J. F. Kaiser, Eds. New York: Wiley, 1966, pp. 218-285.

[5] T. T. Taylor, "Design of line-source antennas for narrow beamwidth and low side lobes," *IRE Trans. Antennas Propagat.*, vol. AP-3, pp. 16-28, Jan. 1955.

[6] R. J. Stegun, "Excitation coefficients and beamwidths of Tschebyscheff arrays," *Proc. IRE*, vol. 41, pp. 1671-1674, Nov. 1953.

[7] A. H. Nuttall, private communication, Naval Underwater Syst. Cen., New London, CT, Apr. 13, 1982.

[8] G. N. Watson, *Theory of Bessel Functions*, 2nd ed. New York: Cambridge Univ. Press, 1966.

[9] P. R. Halmos, *Measure Theory*. Princeton, NJ: Van Nostrand, 1950.

[10] G. A. Campbell and R. M. Foster, *Fourier Transforms for Practical Applications*. Princeton, NJ: Van Nostrand, 1948.

[11] R. L. Streit, "An expansion of the Gegenbauer polynomial $C_n^{\mu}(xy)$," NUSC Tech. Rep. 6579, Naval Underwater Syst. Cen., New London, CT, Mar. 25, 1982.

[12] R. Askey and J. Fitch, "Integral representations for Jacobi polynomials and some applications," *J. Math. Anal. Appl.*, vol. 26, pp. 411-437, 1969.

Roy L. Streit was born in Guthrie, OK, on October 14, 1947. He received the B.A. degree (with honors) in mathematics and physics from East Texas State University, Commerce, in 1968, the M.A. degree in mathematics from the University of Missouri, Columbia, in 1970, and the Ph.D. degree in mathematics from the University of Rhode Island, Kingston, in 1978.

He is currently an Adjunct Assistant Professor for the Department of Mathematics, University of Rhode Island. He was a Visiting Scholar in the Department of Operations Research, Stanford University, Stanford, CA, during 1981-1982. He joined the staff of the Naval Underwater Systems Center, New London, CT (then the Underwater Sound Laboratory), in 1970. He is an applied mathematician and has published work in several areas, including antenna design, complex function approximation theory and methods, and semi-infinite programming. He is currently conducting research sponsored by the Office of Naval Research in nonlinear optimization methods for acoustic array design.

Orthogonal Polynomial Based Array Design

R. L. Streit

Abstract

Array weighting designs of the Dolph–Chebyshev and Kaiser–Bessel type are based mathematically on orthogonal polynomials. The theoretical properties of these polynomials give rise to the desirable properties of the resulting arrays. This paper presents results for array weights based on a very general set of orthogonal polynomials called the Jacobi polynomials. Many interesting array far–field beampatterns are exhibited. A practical means of computing all the array weights exactly by means of one fast Fourier transform (FFT) is given. This method is quick and accurate and can compute the weights for arrays having large numbers of elements. It can efficiently compute both Dolph–Chebyshev and discrete Kaiser–Bessel weights as special cases.

I. INTRODUCTION

Array weights of the Dolph-Chebyshev and Kaiser-Bessel type are based on orthogonal polynomials. The desirable properties of these arrays are due entirely to the properties of the underlying orthogonal polynomials. In this paper the mathematical design methodology developed in [1], which parallels the techniques of the Dolph-Chebyshev designs, is further explored for Jacobi polynomials.

Analytical expressions for the underlying weights are not sought here, as they are in [1], because such expressions are probably not competitive with the exact FFT based method presented in this paper. The primary purpose of this paper is to present a unified FFT based method for computing array weights based on any of the Jacobi polynomials. This method is quick and accurate and can compute the weights for arrays having large numbers of elements. The Appendix gives a short Fortran program for computing the weights (given a subroutine for the FFT). This program can efficiently compute both Dolph-Chebyshev and discrete Kaiser-Bessel weights as special cases.

This paper represents, in a sense, a completion of certain ideas about array design using orthogonal polynomials. Dolph was apparently the first to use an orthogonal polynomial for designing an array weighting function. The polynomial he used was the Chebyshev polynomial of the first kind, $T_n(x)$, and he was able to prove an optimality condition. Unfortunately his proof of this optimality condition relies on the unique behavior of the graph of $T_n(x)$, and so generalizations of the optimality condition seem unlikely. Nonetheless, the use of different orthogonal polynomials can lead to useful weighting functions. For example, if the Chebyshev polynomial of the second kind, $U_n(x)$, is used in place of $T_n(x)$ a weighting function of Kaiser-Bessel type results (see [1]). It is also possible to use a general family of orthogonal polynomials that contains both $T_n(x)$ and $U_n(x)$ as special cases. The Gegenbauer polynomials, $C_n^{\mu}(x)$, are one such family; that is, $T_n(x)$ and $U_n(x)$ are the special cases $\mu = 0$ and $\mu = 1$, respectively. This family is used in [1] and gives useful and interesting designs. The most general of the so-called classical orthogonal polynominals that contains all these examples as special cases is the family of Jacobi polynomials, $P_n^{(\alpha, \beta)}(x)$. For $\alpha = \beta = \mu - 1/2$ they reduce to $C_n^{\mu}(x)$.

Do Jacobi polynomials turn out to be useful? The examples presented in this paper indicate that, although many interesting new array designs are possible using the Jacobi polynominals, the most useful designs in this general family are probably those that have already been discussed. Consequently the new Jacobi designs might be said to be, at present, "a solution looking for a problem."

II. WEIGHT GENERATION BY FFT

The far-field beampattern of a general linear beamformer for a linear equispaced array having $2N + 1$ elements ($N \geq 1$) with element positions $x_k = k\,\lambda/(\nu D)$, $k = 0, \pm 1, \ldots, \pm N$, is given by:

$$F(u) = \sum_{k=-N}^{N} w_k \exp(-i2\pi\, ku/(\nu D)) \tag{1}$$

where the integer $D \geq 1$ is given and $\nu > 0$ is a fixed real constant (νD is the number of elements per wavelength), λ is the wavelength of the design frequency, and u is defined by

$$u = \sin\theta_a - \sin\theta_s, \tag{2}$$

where θ_s, $-\pi/2 \leq \theta_s \leq \pi/2$, is the steering (look) angle and θ_a, $-\pi/2 \leq \theta_a \leq \pi/2$, is the arrival angle of a plane wave. Both angles are measured from a line normal to the array axis. The weights $\{w_k\}$ can be, in general, any set of complex constants.

Define the functions

$$t_D(z) = \sum_{k=-D}^{D} a_k\, z^k \tag{3}$$

$$P_n(z) = \sum_{k=0}^{n} b_k\, z^k \qquad , n \geq 0, \tag{4}$$

where $\{a_k\}$ and $\{b_k\}$ are specified constants. By simple algebra

$$P_n(t_D(z)) = \sum_{k=-nD}^{nD} c_k z^k \tag{5}$$

where $\{c_k\}$ depend on both $\{a_k\}$ and $\{b_k\}$. Substituting $z = \exp(-i\pi u/(\nu D))$ gives

$$H(u) = P_n(t_D(\exp(-i\pi u/(\nu D)))) \tag{6}$$

$$= \sum_{k=-nD}^{nD} c_k \exp(-i2\pi ku/(2\nu D)). \tag{7}$$

By comparing (7) with (1), it is seen that H(u) is the far-field beampattern of a linear array with 2nD + 1 elements, equispaced $\lambda/(2vD)$ apart, and with the weight c_k applied to the k-th element. It will be shown that the array weights c_k can be computed from function values of t_D and P_n by means of an FFT.

When some of the weights $c_k = 0$, the corresponding elements may be eliminated from the physical array without altering the array's far-field beampattern. Elimination of zero weighted elements (whenever possible) minimizes the total number of elements required. This consideration is especially important when t_D and P_n are chosen so that $P_n(t_D(z))$ is either even or odd in z, for then about half the elements need not be physically present. This is discussed further in section III.

We now show that the weights $\{c_k\}$ in (7) can be computed exactly from (3) and (4) by the fast Fourier transform (FFT). It is stressed that this procedure is theoretically <u>exact</u>, not approximate, for the $\{c_k\}$.

Let f(u) be any complex valued function of a real variable u which can be written exactly in the form

$$f(u) = \sum_{k=-p}^{p} d_k \, e^{-iku} \tag{8}$$

for some complex constants $\{d_k\}$. Let

$$f_k = \frac{1}{2p} \sum_{j=0}^{2p-1} F_j \, e^{i2\pi kj/(2p)}, \quad k = 0, 1, \ldots, 2p-1, \tag{9}$$

where

$$F_j = (-1)^j \, f((p-j)\pi/p) \quad , \; j = 0, 1, \ldots, 2p-1. \tag{10}$$

Thus $\{f_k\}$ is the inverse FFT of order 2p of the sequence $\{F_j\}$. Substitute (8) into (10), and then into (9) to get

$$f_k = \frac{1}{2p} \sum_{j=0}^{2p-1} \left\{ (-1)^j \sum_{q=-p}^{p} d_q \, e^{-i\pi q(p-j)/p} \right\} e^{i\pi kj/p}$$

$$= \frac{1}{2p} \sum_{q=-p}^{p} (-1)^q \, d_q \left\{ \sum_{j=0}^{2p-1} (-1)^j \, e^{i\pi(q+k)j/p} \right\}, \quad k=0,1,\ldots,2p-1.$$

7

The inner sum equals 2p when $q + k = \pm p, \pm 3p, \ldots$ and equals zero otherwise. Since $d_q = 0$ for $|q| > p$,

$$f_k = \begin{cases} (-1)^p (d_{-p} + d_p), & \text{if } k = 0 \qquad (11.a) \\ (-1)^{p-k} d_{p-k}, & k = 1, \ldots, 2p-1. \qquad (11.b) \end{cases}$$

Now let $f(u) = H(\nu Du/\pi)$, where $H(u)$ is given by (7), and $p = nD$. Then

$$F_j = (-1)^j P_n(t_D(e^{-i\pi(nD - j)/nD})), \quad j = 0, 1, \ldots, 2nD-1,$$

and the coefficients c_k in (7) are just

$$c_k = \begin{cases} a_{-D}^n b_n, & \text{if } k = -nD, \qquad (12.a) \\ (-1)^{k-nD} f_{nD-k}, & \text{if } k = -nD+1, \ldots, nD-1 \qquad (12.b) \\ a_D^n b_n, & \text{if } k = nD, \qquad (12.c) \end{cases}$$

where $\{f_k\}_0^{2nD-1}$ is the inverse FFT of $\{F_j\}_0^{2nD-1}$. The coefficients c_{-nD} and c_{nD} cannot be computed directly by FFT because of aliasing, as indicated in (11.a); however, by direct appeal to the defining equations (3) - (6), it follows that c_{-nD} and c_{nD} are as given by (12.a) and (12.c), respectively. In fact, (11.a) can be used as a check on numerical accuracy in the computations.

It should be obvious that some special forms of t_D and P_n can be used advantageously to reduce the size of the FFT required to compute (12). In general, however, no special structure exists and the smallest FFT size that can be used has order 2nD.

In cases where 2nD is not an integer power of two, the FFT is still applicable by zero filling in t_D and P_n. That is, D is replaced by the smallest power of two which exceeds or equals D, say D'. The function t_D is then merely considered to be a special case of $t_{D'}$. Similarly, P_n is a special case of $P_{n'}$ for some smallest power of two, n', which is greater than or equal to n. The required size of the FFT is thus 2n'D'. The coefficients $\{c_k\}$ are still given by (12); however, from (7), it must be the case that $c_k = 0$ for $|k| > nD$.

The Appendix gives a Fortran subroutine for computing the array weights by an FFT, given subroutines for evaluating $P_n(z)$ and $t_D(z)$. The program is specialized to the case D = 1, but it can be easily altered to accommodate larger values of D. The program is also written on the assumption that 2nD is a power of two. As discussed in the preceding paragraph, zero filling allows the most general situation to go through.

III. THE TEN PARAMETER JACOBI WEIGHT FAMILY

The most important special case of H(u) seems to be D = 1. Although there may be interesting possibilities when D > 1 (consider, for example, the identity $T_n(T_D(\cos\theta)) = T_{n+D}(\cos\theta)$, where T_n is the Chebyshev polynomial of the first kind), these cases are not explored in this paper.

Before proceeding, it is instructive to see first how the usual Dolph-Chebyshev case for half wavelength equispaced arrays is derived as a special case of H(u). Let

$\tilde{N}$ = number of array elements
$D = 1$
$\nu = 2$
$\tilde{n} = \tilde{N} - 1$
$t_1(z) = Z_0(z^{-1} + z)/2$
$P_{\tilde{n}}(z) = T_{\tilde{n}}(z)$,

where $T_{\tilde{n}}(z)$ is the Chebyshev polynomial of the first kind and the real constant Z_0 is given by

$$Z_0 = \frac{1}{2}\left\{(Q + (Q^2-1)^{1/2})^{1/\tilde{n}} + (Q - (Q^2-1)^{1/2})^{1/\tilde{n}}\right\} \tag{15}$$

where

$Q = 10^{|S|/20}$
S = specified sidelobe level (in dB).

For these values it follows that $t_1(\exp(-i\pi u/2)) = Z_0\cos(\pi u/2)$ and, from (6) and (7),

$$H(u) = T_{\tilde{n}}(Z_0\cos(\pi u/c)) \tag{16}$$

$$= \sum_{k=-\tilde{n}}^{\tilde{n}} c_k \exp(-i\pi k u/2). \tag{17}$$

By comparing (17) to (1), it would appear at first glance that this array is quarter wavelength equispaced with $2\tilde{n} + 1 = 2\tilde{N} - 1$ elements. However, every other coefficient in (17) is identically zero because $T_{\tilde{n}}(z)$ is always either an even or an odd function in z. Deleting the zero weighted elements reduces (17) to two slightly different cases, depending only on whether $\tilde{N}$ is even or odd. These cases are not given explicitly here. Note, however, that $\tilde{N}$ even implies that $\tilde{n}$ is odd and that $T_{\tilde{n}}(z)$ contains no even powers of z, which means that $T_{\tilde{n}}(z)$ has $(\tilde{n} + 1)/2$ non-zero coefficients and, consequently, that $T_{\tilde{n}}(t_1(z))$ has precisely $\tilde{n} + 1 = \tilde{N}$ non-zero terms in the expansion (17). Similar reasoning holds for $\tilde{N}$ odd.

To summarize: The Dolph-Chebyshev array design is thought of, in the context of this paper, as a quarter wavelength equispaced array in which half the elements (every other one) has been zero weighted. Dolph-Chebyshev arrays of both even and odd numbers of elements are thought of in this way.

From (3), the most general form for $D = 1$ is

$$t_1(z) = a_{-1} z^{-1} + a_0 + a_1 z.$$

For reasons that will become clear, the slightly more restrictive form

$$t_1(z) = z_0(r_0^{-1} z^{-1} + a_0 + r_0 z)/2, \; r \neq 0, \tag{18}$$

is adopted, where z_0, r_0, and a_0 are arbitrary complex constants. The only useful form excluded by (18) is obtained by changing the sign of the highest order term; this latter form corresponds to "difference" patterns and, for ease of exposition, is not discussed further. For the remainder of the paper, (18) is taken as the definition of $t_1(z)$. Note that $t_1(e^{-i\theta}) = z_0 \cos\theta$ if $r_0 = 1$ and $a_0 = 0$.

The most general class of polynomials $P_n(z)$ considered in this paper are the Jacobi polynomials, denoted $P_n^{(\alpha,\beta)}(z)$. They are defined explicitly by

$$P_n^{(\alpha,\beta)}(z) = \frac{1}{n!} \sum_{k=0}^{n} \binom{n}{k} (n+\alpha+\beta+1)_k \, (\alpha+k+1)_{n-k} \left(\frac{z-1}{2}\right)^k \tag{19}$$

for all complex values of α, β, and z. For $\alpha > -1$ and $\beta > -1$, the Jacobi polynomials are orthogonal on the real interval $-1 \leq z \leq +1$, but they are not necessarily orthogonal for other values of α and β. The best available method for computing $P_n^{(\alpha,\beta)}(z)$ relies not on (19) but on the three term recursion which they satisfy. The published algorithm [2], [3] based on this recursion is easily modified to compute the Jacobi polynomials for complex α, β, and z for all values of the degree n that are likely to be of practical interest, say $n \leq 150$. A thorough mathematical treatment of $P_n^{(\alpha,\beta)}(z)$ is available in [4].

The generalized Laguerre and Hermite polynomials, together with the Jacobi polynomials, constitute a complete list of the so-called classical orthogonal polynomials. Array designs can also be based on the Laguerre and Hermite polynomials and will, of course, be different from those based on Jacobi's. Although these designs are probably interesting, in this paper attention is restricted to the Jacobi polynomials.

The use of (18) and (19) gives rise to a five parameter family of weights which includes nearly all the well known analytic families of weights as special cases. (The most prominent exception is Taylor weighting). The five parameters are z_0, r_0, a_0, α, and β. Each parameter can be complex, so there are actually 10 real parameters if the real and imaginary parts of each are counted separately. The Fortran program listed in the Appendix is written for this general case.

The simplest way to explore the properties of these ten parameters is to perturb each parameter separately while holding the others fixed at some nominal value. The nominal parameter values chosen here for the examples are those that give rise to the Dolph-Chebyshev design for an array of 33 elements with a sidelobe level of -30 dB. Specifically,

$$\begin{aligned} \alpha_0 &= -.50 + 0.0\,i \\ \beta_0 &= -.50 + 0.0\,i \\ a_0 &= 0.0 + 0.0\,i \\ r_0 &= 1.0 + 0.0\,i \\ z_0 &= Z_0 = 1.008408 + 0.0\,i, \end{aligned} \tag{20}$$

where Z_0 is computed from (15) with $S = -30$ dB and $\tilde{n} = 32$. Recalling earlier remarks in this section concerning the interpretation of half wavelength equispaced arrays as quarterwavelength equispaced arrays with zero weights, set $n = 32$ in (6) and (7). Thus, in principle, the example is a $2n + 1 = 65$ element quarterwavelength equispaced array. The coefficients $[c_k]$ are computed from (12). In (12.a) and (12.c), note that the identity [4, Eq. (4.21.6)]

$$b_n = 2^{-n} \binom{2n + \alpha + \beta}{n}, \quad n \geq 0, \tag{21}$$

holds and so, from (18),

$$a_D = \frac{1}{2} z_0 r_0, \; a_{-D} = \frac{1}{2} z_0 r_0^{-1} \tag{22}$$

Each of the ten parameters is both increased as well as decreased from its nominal value given in (20). Thus there are 21 cases, including the nominal case itself in (20). Table 1 displays these cases and gives each an identifying case name. To each case in Table 1 there corresponds a graph of the far-field beampattern H(u) on the u-interval [0,4] and two bar charts, one for the real part and one for the imaginary part of the weights corresponding to that case.

Figure Number	Case Names				Value of Perturbed Parameter			
1	NOM				no deviations from (20)			
2	Z.1	Z.2	Z.3	Z.4	z_0+.003	z_0-.003	z_0+.003i	z_0-.003i
3	A.1	A.2	A.3	A.4	a_0+.003	a_0-.003	a_0+.003i	a_0-.003i
4	R.1	R.2	R.3	R.4	r_0+.03	r_0-.03	r_0+.03i	r_0-.03i
5	α.1	α.2	α.3	α.4	α_0+.3	α_0-.3	α_0+.3i	α_0-.3i
6	β.1	β.2	β.3	β.4	β_0+.3	β_0-.3	β_0+.3i	β_0-.3i

Table 1 Perturbed parameter values; deviation from the nominal values (20)

In general, H(u) is periodic with a period of length $2\nu D$. Since $D = 1$ and $\nu = 2$ in these examples, any interval of length 4 suffices to exhibit all the structure of H(u).

In all the bar charts presented, upward lines indicate positive weights and length is proportional to magnitude. Similarly, downward lines indicate negative weights. These upward and downward lines are ordered from left to right and correspond to elements numbered from -32 to +32. Any element receiving a zero weight is indicated by a simple "x" marking its position. In particular, notice that the nominal case, NOM, of Figure 1 has only 33 non-zero weights. The nominal case, as has been said, is a half wavelength equispaced array being treated as a quarter wavelength equispaced array. The weights in each case are normalized by the largest magnitude of any real or imaginary part; thus, the normalization between cases is not exactly the same.

Figure 1 is the reference case (20) and needs no further comment.

Figure 2 perturbs only z_0. The cases Z.1 and Z.2 are expected since z_0 is merely increased or decreased in its real part alone. When z_0 is perturbed by adding an imaginary component, the array still has 33 non-zero weights and so is, in effect, half wavelength equispaced. It is surprising how much can be added to the imaginary parts of the weights without seriously degrading the beampattern. The beampatterns in Z.3 and Z.4 are identical.

Figure 3 perturbs a_0 from its nominal zero value. Any perturbation produces a quarter wavelength equispaced array which is symmetrically weighted. One way to discuss the results is to visualize the 65 element array as being composed of two half wavelength equispaced arrays---one having 33 elements and the other 32 elements with the elements of the two arrays interlaced. Thus, perturbing the real part of a_0 is equivalent to adding or subtracting the outputs of these two arrays. Perturbing the imaginary part of a_0 is equivalent to adding or subtracting the outputs after first putting them in phase quadrature with respect to each other. The beampatterns A.3 and A.4 are identical to each other, but they are NOT the same as Z.3 and Z.4.

Figure 4 perturbs r_0 from its nominal value of +1. Any small perturbation produces asymmetrically weighted half wavelength equispaced arrays. Real perturbations of r_0 produce only real weights and have beampatterns without any true nulls. Pure imaginary perturbations do not alter the beampattern from its nominal case, even though the weights develop an interesting sinusoidal character in their imaginary parts.

Figure 5 perturbs α from its nominal value of $\alpha_0 = -1/2$. Any perturbation produces a quarter wavelength equispaced array which is symmetrically weighted. Real perturbations yield real weights while pure imaginary perturbations yield complex weights. The first grating lobe in case α.1 is at about -8 dB instead of 0 dB; the same is true of the MRA in case α.2.

Figure 6 perturbs β from its nominal value of $\beta_0 = -1/2$. Any perturbation produces quarter wavelength equispaced arrays which are symmetrically weighted. Real perturbations yield real weights, while pure imaginary perturbations yield complex weights. The first grating lobe in case β.2 is suppressed to about -8 dB, while in case β.1 the MRA is depressed to -8 dB. Figure 6 and Figure 5 should be compared.

The following observations seem to hold:

1. The real part of α controls the "upper" envelope of H(u) near the center peak.

2. The absolute value of the imaginary part of α controls the "lower" envelope of H(u) near the center peak.

3. The real part of β controls the "upper" envelope of H(u) near the first grating lobe.

4. The absolute value of the imaginary part of β controls the "lower" envelope of H(u) near the first grating lobe.

Other parameters (the imaginary parts of a and z_0) also affect the "lower" envelope of H(u), but the dominant effects seem to be due to the imaginary parts of α and β. The imaginary part of r does not affect the lower envelope at all.

By changing the parameters simultaneously in different ways, the different effects may be combined, at least for small perturbations. Examples of this are not included here.

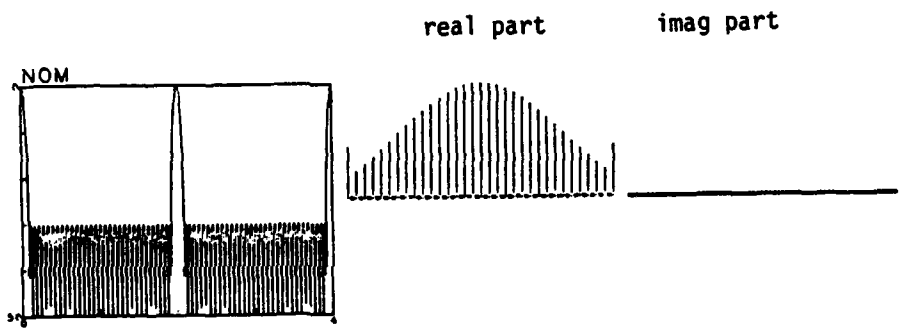


Figure 1. No perturbations; the nominal case (20)

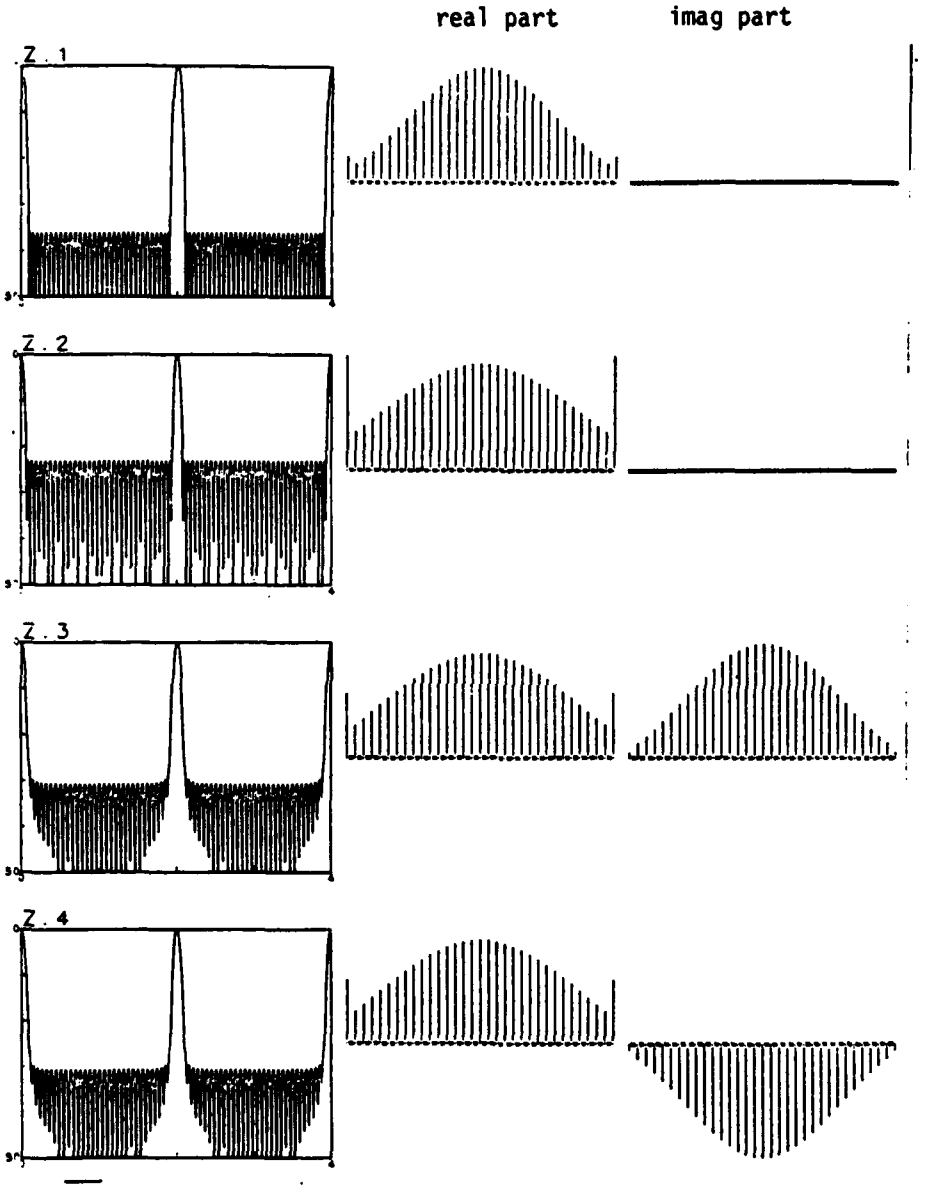


Figure 2. Perturbations of z_0

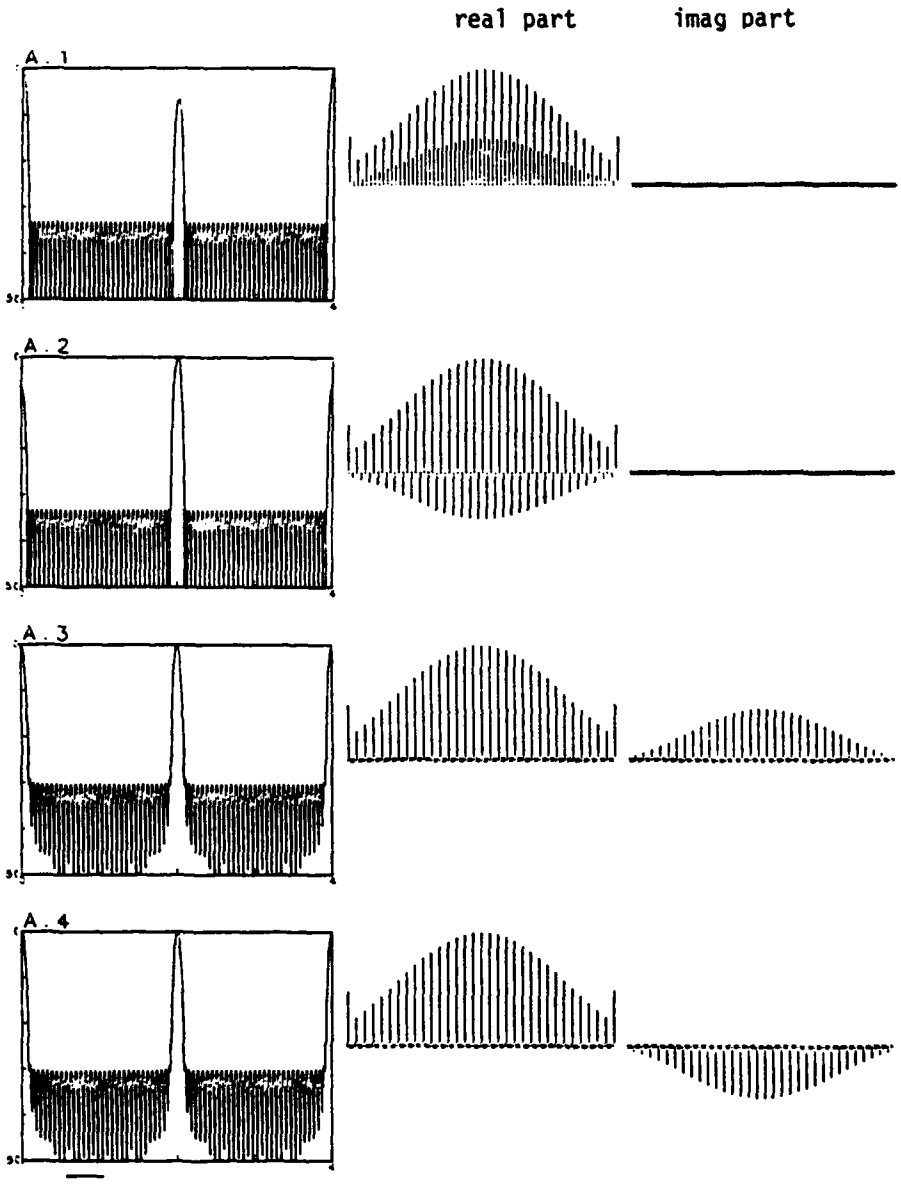


Figure 3. Perturbations of a_0

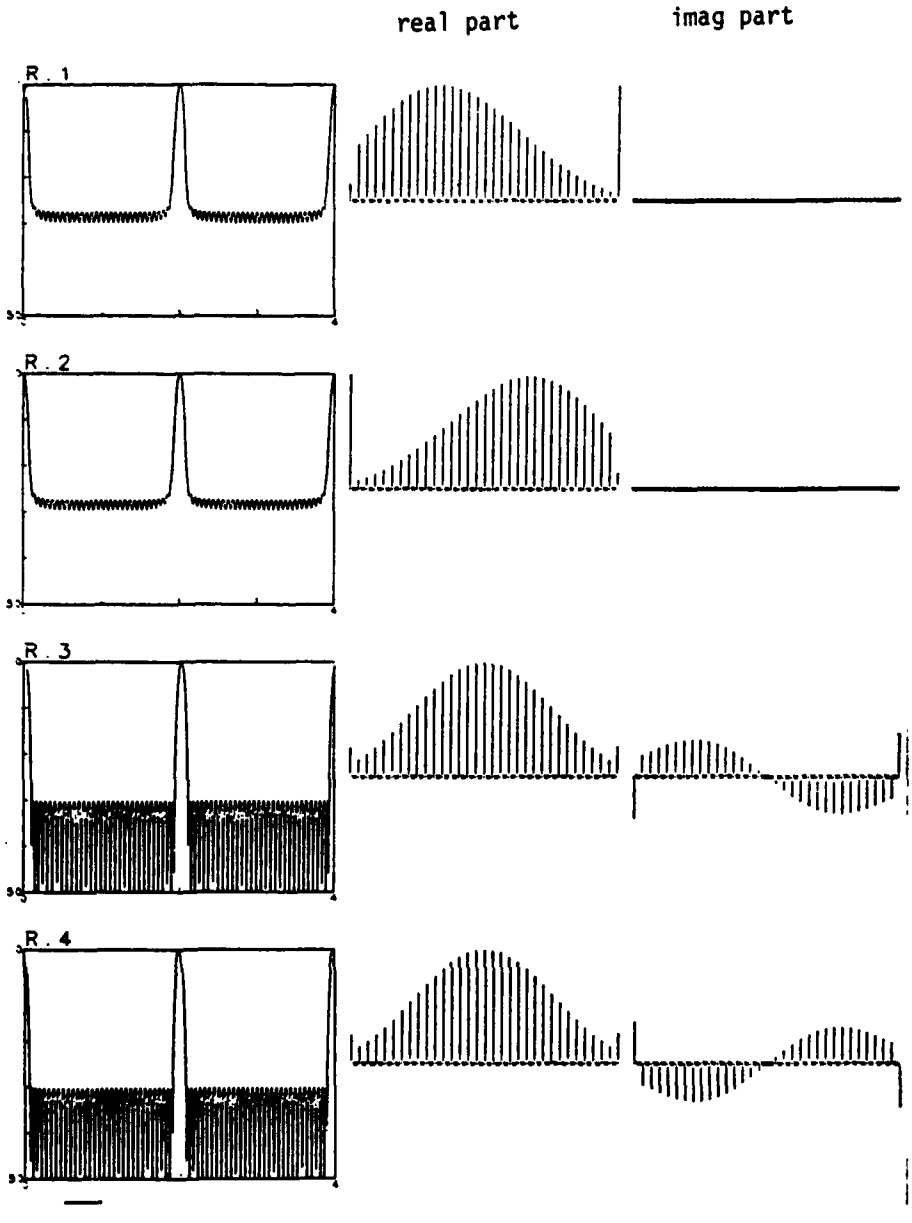


Figure 4. Perturbations of r_o

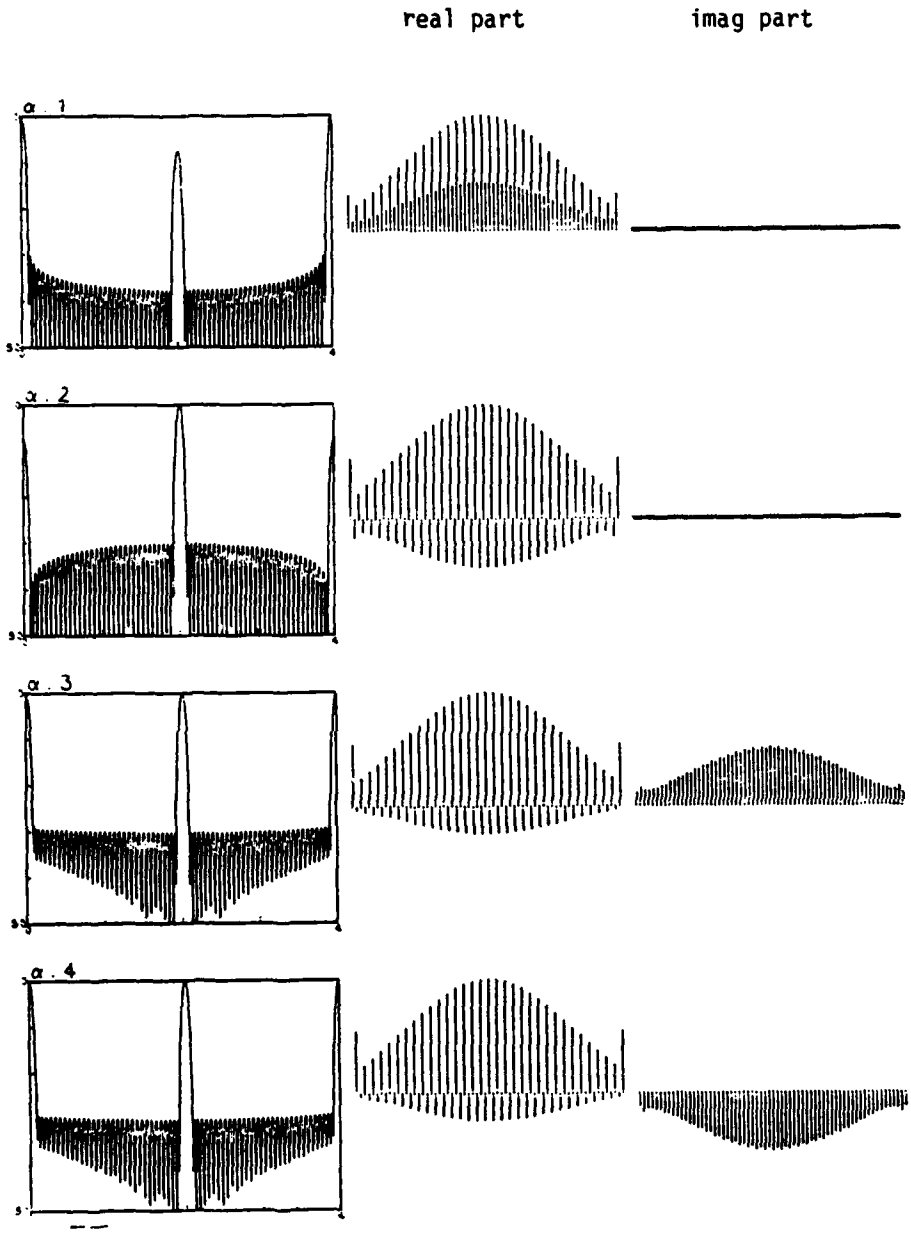


Figure 5. Perturbations of α_0

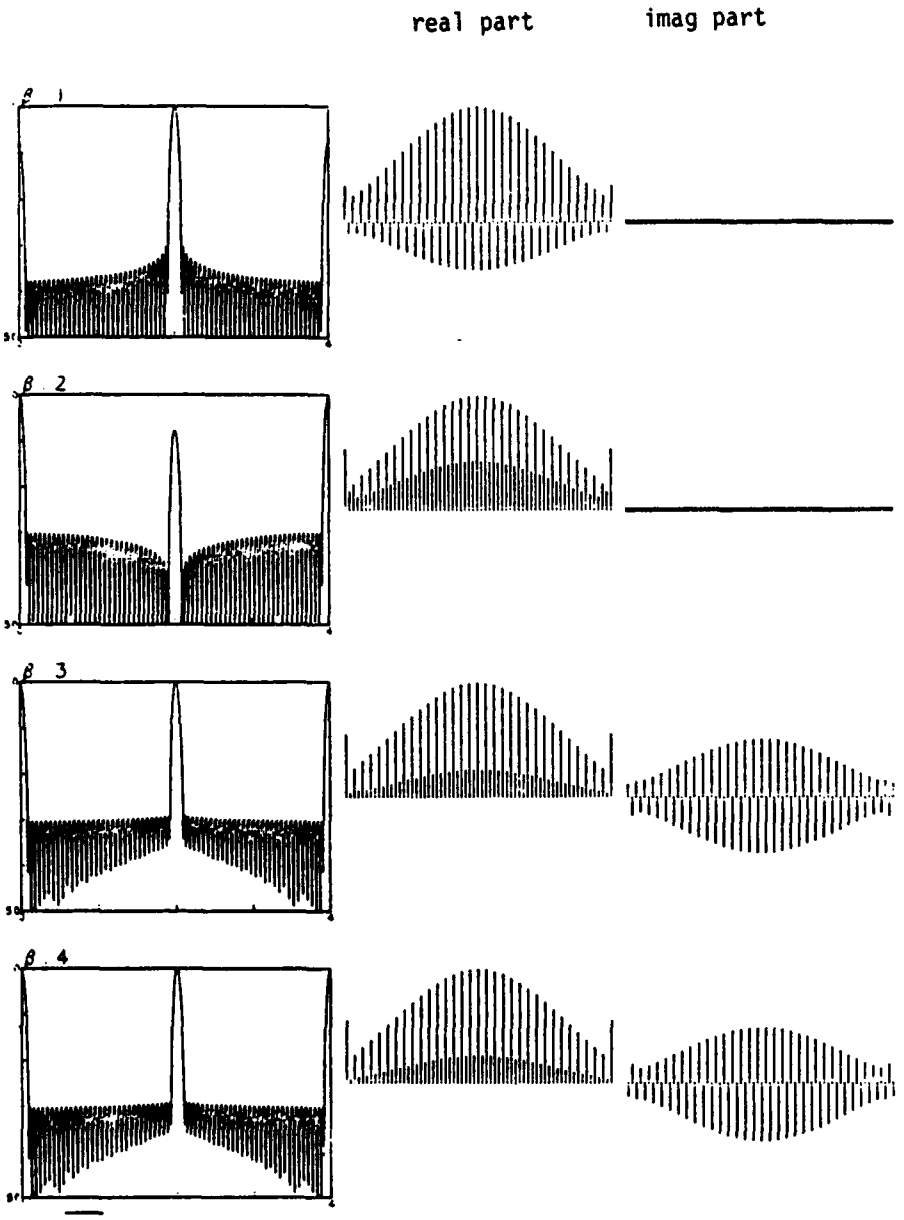


Figure 6. Perturbations of β_0

IV. SUMMARY AND CONCLUDING REMARKS

It has been shown that array weights based on the Jacobi orthogonal polynomials can be computed exactly by means of FFT. As a special case, weights based on the Gegenbauer polynomials can also be computed exactly by FFT, instead of analytically as in [1]. Examples have been presented to show the effects of varying the ten parameters in the Jacobi family.

Further work in this area is possible. In addition to the Jacobi polynomials, one may also use the generalized Laguerre and the Hermite polynomials. In fact, any orthogonal polynomial family that has interesting structural features can be the basis of a weighting family which inherits this structure. In a different direction, certain cases for $D > 1$ may yield interesting designs and have not been explored. The weights corresponding to all these cases can computed exactly by the FFT method presented in this paper.

APPENDIX

The Fortran program JACWTS listed below assumes that D = 1 and that 2nD is a power of 2. The function $t_0(z)$ is defined exactly as in (18), and the polynomial $P_n(z)$ is taken to be the Jacobi polynomial $P_n^{(\alpha,\beta)}(z)$.

This program is an implementation of the (exact) FFT method described by Eq. (12), where $a_{\pm D}$ and b_n are given by (22) and (21), respectively. The user of JACWTS need only specify values for α, β, a_0, r_0, and z_0. In JACWTS these variables are referred to by the labels ALPHA, BETA, A0, R0, Z0, respectively. The arrays X and Y contain, on output, the real and imaginary parts of the array weights $\{c_k\}$. These two arrays must be dimensioned at least 2nD+1 in the routine which calls JACWTS. The integers n and D are referred to by the labels N and D, respectively, in the subroutine argument list. Also, LOGN and LOGD are defined so that N = 2**LOGN and D = 2**LOGD.

This program assumes that a subroutine named JACOBI evaluates the Jacobi polynomial (19) for arbitrary complex values of α, β, and z. This subroutine can be based on the published codes in [2] and [3]. This program also assumes that subroutines are available for computing a complex FFT of size 2nD; the particular ones used here are based on Markel's method and are not listed. Their names are DPMCOS and DPMFFT. These routines require a work array, C, dimensioned at least 2nD in the routine which calls JACWTS.

JACWTS is written in double precision complex mode to forestall any numerical round-off error problems that might arise. The test suggested in Section II (that follows from the resolution of the aliasing effects as in (12)) is incorporated. It is the only test used to ascertain whether numerical round-off of significant proportions occurred. No numerical difficulties have been detected by this test to date, which indicates that the computation is usually numerically reliable.

```
      SUBROUTINE JACOTS(X,Y,C,N,LOGN,D,LOGD,ALPHA,BETA,A0,R0,Z0)
      COMPLEX*16 Z,T,H,S,R,TSUBD,ALPHA,BETA,Z0,A0,R0,JACOBI
      INTEGER N,LOGN,D,LOGD,TWOND
      DOUBLE PRECISION EPSI,ARG,PI,X(1),Y(1),C(1)
      DATA PI,EPSI/3.14159265358979323D0,0.5D-7/
      M=LOGN+LOGD+1
      ND=N*D
      TWOND=2*ND
C     ... ALL WEIGHTS EXCEPT THE FIRST AND THE LAST
      I=+1
      DO 10 J=1,TWOND
      ARG=-PI*(ND-J+1)/ND
      Z=DCMPLX(COS(ARG),SIN(ARG))
      T=TSUBD(D,A0,R0,Z0,Z)
      H=JACOBI(N,ALPHA,BETA,T)
      X(J)=I*DREAL(H)
      Y(J)=I*DIMAG(H)
      I=-I
10    CONTINUE
      CALL DPMCOS(C,TWOND)
      CALL DPMFFT(X,Y,C,M,+1)
      I=+1
      DO 15 J=1,TWOND
      X(J)=I*X(J)/TWOND
      Y(J)=I*Y(J)/TWOND
      I=-I
15    CONTINUE
      ... THE FIRST AND LAST WEIGHTS
      H=.25D0*Z0*R0
      S=.25D0*Z0/R0
      R=1.0D0
      T=1.0D0
      DO 18 I=1,N
      Z=((N+N-I+1)+ALPHA+BETA)/I
      T=T*H*Z
      R=R*S*Z
18    CONTINUE
      X(1)=DREAL(R)
      Y(1)=DIMAG(R)
      X(TWOND+1)=DREAL(T)
      Y(TWOND+1)=DIMAG(T)
      ... NUMERICAL ACCURACY TEST
      ARG=(ABS(X(1)-DREAL(R+T))+ABS(Y(1)-DIMAG(R+T)))
     +     /(1.0D0+ABS((1))+ABS(Y(1)))
      IF(ARG.GT.EPSI)PRINT 50
50    FORMAT(' NUMERICAL ROUND-OFF ERROR IS SIGNIFICANT.')
      RETURN
      END
      FUNCTION TSUBD(D,A0,R0,Z0,Z)
      COMPLEX*16 Z,Z0,A0,R0,TSUBD
      INTEGER D
C     TREAT THE CASE D=1; IGNORE OTHER VALUES.
      TSUBD=.5D0*Z0*( (1.0D0/(R0*Z)) + A0 + (R0*Z) )
      RETURN
      END
```

22

REFERENCES

1. R. L. Streit, "A Two-Parameter Family of Weights for Nonrecursive Digital Filters and Antennas," IEEE Trans. on ASSP, Vol. ASSP-32, February 1984, pp. 108-118.

2. B. F. W. Witte, "Algorithm 332, Jacobi Polynomials," Comm. ACM, Vol. 11, June 1968, p. 436.

3. O. Skovgaard, "Remark on Algorithm 332," Comm. ACM, Vol. 18, February 1975, pp. 116-117.

4. G. Szego, Orthogonal Polynomials, Fourth Edition, AMS Colloquium Publications, Vol. XXIII, American Mathematical Society, Providence, RI, 1975.

A Discussion Of Taylor Weighting For Continuous Apertures

R. L. Streit

Abstract

It is shown that Taylor's beampattern for a continuous aperture can be computed analytically without Fourier transforming the weighting function itself, thereby achieving economies in computational effort in some modeling situations. A short Fortran program is given. An approximate formula for the half-power beamwidth is derived. It is pointed out that the Taylor weighting function can be negative for large n, a fact that does not seem to be well known. In addition, modification of Taylor's design to force the weighting function to go to zero as a power a of distance from the aperture endpoints is discussed. For $a = 1$ and $a = 2$, this results in an increase of 5 and 10 percent, respectively, in the beamwidth.

I. INTRODUCTION

This Memorandum is a review of Taylor's original weighting function for continuous apertures. It is presented in some detail in Sections II and III. It is shown that Taylor's beampattern and weighting function can be computed easily by analytically exact formulas. Taylor's beampattern turns out to be the product of a rational function and the beampattern of a uniformly weighted aperture.

Also reviewed is a modification due to Rhodes of Taylor's pattern for the purpose of forcing the weighting function to go zero as a power α of distance from the aperture endpoints. This results in a 5% increase in beamwidth over the beamwidth of Taylor's original pattern if $\alpha = 1$, and a 10% increase if $\alpha = 2$ (for $\bar{n} = 10$; see below). These modifications are discussed in Section IV.

Taylor's original paper [1] derives a symmetric weighting function for a continuous aperture. He does not discuss or even mention its use for arrays of point sensors. His method is essentially an ad hoc, but intuitively sensible, procedure which blends together the desirable characteristics of uniform weighting and the van der Maas weighting into one weighting design. The blending is accomplished by careful specification of the beampattern nulls. The various sidelobe levels do not enter the method's derivation. In other words, the sidelobes are whatever they turn out to be after specification of the nulls.

It is often said that Taylor weighting makes the first few sidelobes near the mainlobe nearly flat; that is, all "near-in" sidelobes have essentially the same amplitude. This statement is erroneous. See Figure 1, for example, where the 9 sidelobes ($\bar{n} = 10$) nearest the mainlobe would all be at -20 dB if the statement were true. Instead, the first sidelobe is at -20 dB and the ninth sidelobe is at (roughly) -25 dB.

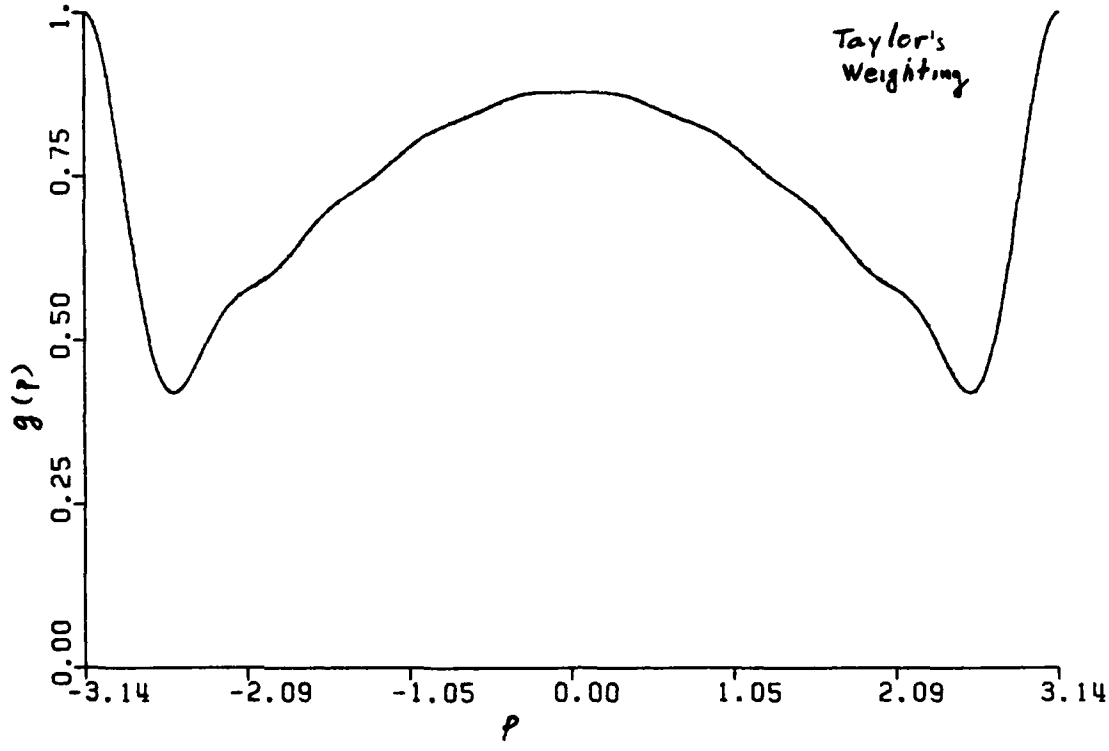
It is a useful fact that the beampattern corresponding to Taylor weighting can be computed analytically, without Fourier transforming the weighting function. This can be seen from Taylor's original discussion [1], which is reviewed in this Memorandum. Taylor's original notation is retained here. Appendix A gives a FORTRAN program which computes the beampattern and/or the weighting function using the analytical formulas developed below. In addition, it computes the exact half-power beamwidth.

The aperture is assumed to lie on the p-interval from $-\pi$ to $+\pi$. The weighting function g(p) is related to the far-field beampattern F(z) by

$$F(z) = \int_{-\pi}^{\pi} g(p)\, e^{izp}\, dp. \qquad (1)$$

Taylor assumes throughout that g(p) is a real even function. Consequently, F(z) is also an even function of z.

It is well known that



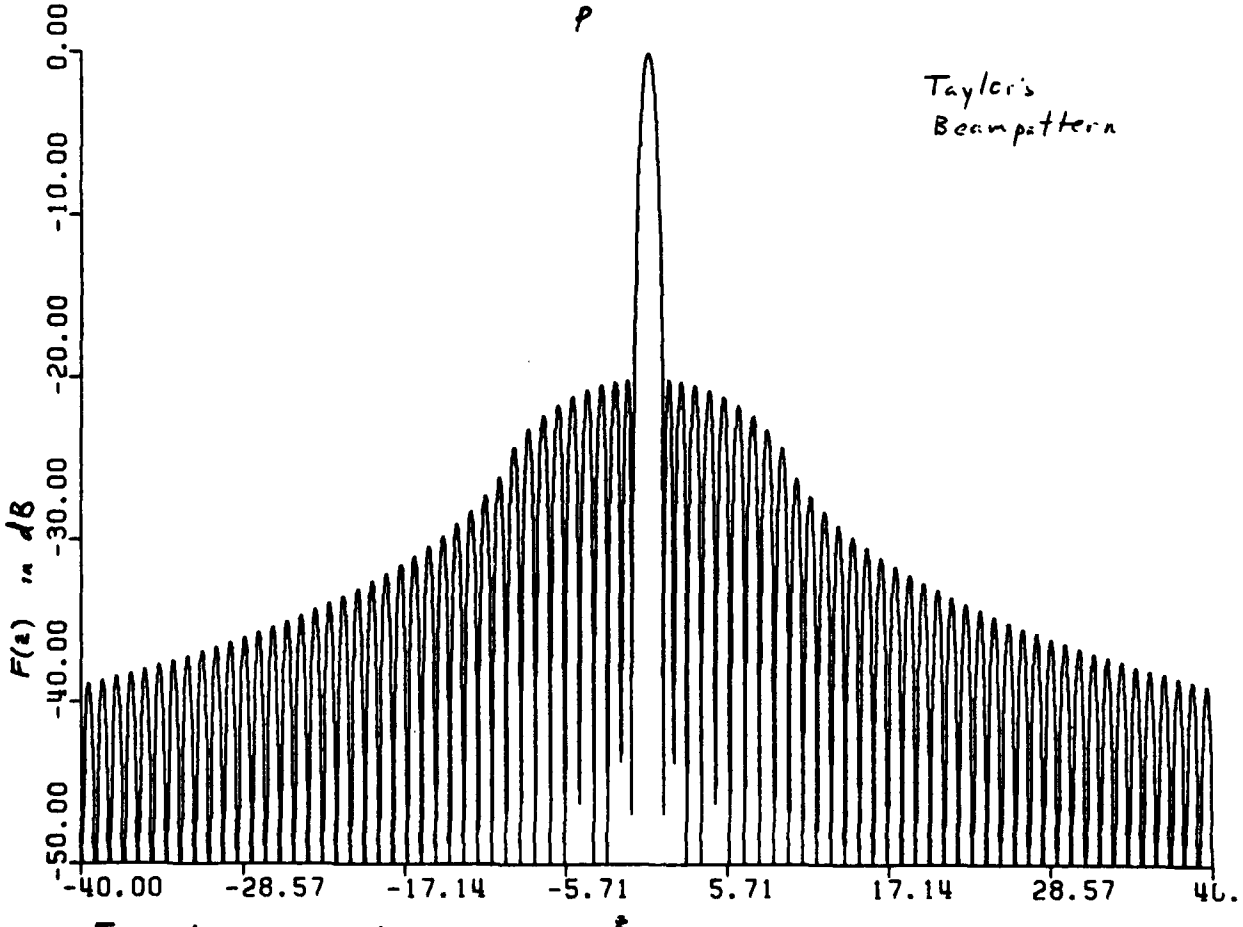


Figure 1. Taylor design for -20dB sidelobe level and $\bar{n} = 10$

$$F(z) = 2\pi \sum_{m=0}^{\infty} \epsilon_m F(m) \left(\frac{\sin \pi(z-m)}{\pi(z-m)} + \frac{\sin \pi(z+m)}{\pi(z+m)} \right) \tag{2}$$

where $\epsilon_0 = 1$ and $\epsilon_m = 2$ for $m \geq 1$. In other words, knowledge of the integer samples of $F(z)$ implies knowledge of $F(z)$ everywhere. A very different representation of $F(z)$ is the infinite product

$$F(z) = \prod_{n=1}^{\infty} \left(1 - \frac{z^2}{z_n^2} \right) \tag{3}$$

where $\{z_1, z_2, \ldots\}$ is a complete list of all the positive zeros of $F(z)$. It is an interesting mathematical fact that these zeros must all lie on the real z-axis. For example, uniform weighting $g(p) = 1/(2\pi)$ gives

$$F(z) = \frac{\sin \pi z}{\pi z},$$

whose positive nulls are $\{1,2,3, \ldots.\}$ From (3), then,

$$\frac{\sin \pi z}{\pi z} = \prod_{n=1}^{\infty} (1 - \frac{z}{n^2}), \tag{4}$$

a well known identity dating back at least to Euler's time (circa 1750).

By means of his choice of nulls $\{z_n\}$ in the representation (3) of $F(z)$, Taylor sought a beampattern which had a flat envelope near the mainbeam and, for large z, an asymptotic 6 dB/octave decay rate. He also sought by this same means a physically realizable aperture to approximate the physically unrealizable ideal van der Maas function. (It is unrealizable because of the presence of delta function spikes at the aperture end-points, $p = \pm \pi$.) Taylor found a set of nulls which came close to attaining his first objective and which did attain his second objective. The next section is a description of Taylor's nulls.

II. TAYLOR'S NULL SPECIFICATION

Taylor specifies the nulls z_n of his beampattern, starting with $n = \bar{n}$, to be exactly the same as those of the uniform weighting function; that is,

$$z_n = n \qquad \text{for } n = \bar{n}, \bar{n} + 1, \ldots \tag{5}$$

The positive integer $\bar{n}$ is a free parameter which can be chosen as desired. Note that $\bar{n} = 1$ gives exactly uniform shading. Note also that the null list (5) guarantees a 6 dB/octave asymptotic decay rate as $z \to \infty$. (This follows from (8) below.)

To complete the list of positive nulls for his beampattern, Taylor selects (when $\bar{n} > 1$) the "near-in" nulls to be

$$z_n = \sigma \sqrt{A^2 + (n - \tfrac{1}{2})^2} \quad \text{for } n = 1,2, \ldots, \bar{n} - 1, \tag{6}$$

where

$$\sigma = \bar{n} \,/\, \sqrt{A^2 + (\bar{n} - \tfrac{1}{2})^2}$$

$$A = \frac{1}{\pi} \ln (R + \sqrt{R^2 - 1})$$

$$R = 10^{|S|/20}$$

S = maximum sidelobe level (in dB).

This choice for the first $\bar{n}$-1 nulls may seem mysterious at first glance, but it is a choice based on the ideal van der Maas function [2], defined by

$$F_0(z, A) = \cos \pi \sqrt{z^2 - A^2}, \quad A>0.$$

It is an interesting mathematical fact that among all beampattern functions F(z) such that

(a) F(z) has a Fourier transform vanishing outside the aperture $-\pi$ to $+\pi$

(b) $|F(z)| \leq 1$ for $|z| \geq A$,

the one with the maximum possible value at z=0 is the van der Maas function $F(z) = F_0(z,A)$. The positive nulls of $F_0(z,A)$ are

$$z_n = \sqrt{A^2 + (n - \tfrac{1}{2})^2}, \; n = 1, 2, 3, \ldots$$

Comparison of these nulls with Taylor's ad hoc null specification (6) shows that Taylor's nulls are related to the van der Maas nulls by a dilation factor σ. The factor σ is chosen to be slightly larger than unity to compensate for the 6 dB/octave decay of the beampattern for $z \geq \bar{n}$. Note that $\bar{n} = \infty$ gives exactly the van der Maas beampattern.

III. TAYLOR'S BEAMPATTERN AND WEIGHTING FUNCTION.

Taylor's beampattern can now be expressed, using (3), as

$$F(z) = \prod_{n=1}^{\bar{n}-1} \left(1 - \frac{z^2}{\sigma^2 (A^2 + (n-\frac{1}{2})^2)}\right) \prod_{n=\bar{n}}^{\infty} \left(1 - \frac{z^2}{n^2}\right). \tag{7}$$

The last expression in (7) can be rewritten, using (4), to give

$$F(z) = \prod_{n=1}^{\bar{n}-1} \left(\frac{1 - \frac{z^2}{\sigma^2 \left(A^2 + \left(n - \frac{1}{2}\right)^2\right)}}{1 - \frac{z^2}{n^2}} \right) \frac{\sin \pi z}{\pi z} \tag{8}$$

In this expression, limits must be taken whenever $z = 0, 1, 2, 3, \ldots, \bar{n}-1$ to avoid the indeterminate form 0/0. See Appendix B. Note that Taylor's beampattern is identically the product of a rational function (of degree $\bar{n}-1$ in z^2) and the beampattern of the uniformly weighted aperture, $\sin(\pi z)/\pi z$.

It is clear that Taylor's beampattern can be computed analytically from (8) without computing the weighting function at all. The representation (2) of F(z) is

$$F(z) = 2\pi \sum_{m=0}^{\bar{n}-1} \epsilon_m \, F(m) \left(\frac{\sin \pi (z-m)}{\pi(z-m)} + \frac{\sin \pi(z+m)}{\pi(z+m)} \right) \tag{9}$$

since $F(n) = 0$ for $n \geq \bar{n}$. This is not as efficient as using (8). However, it does yield an efficient way to compute the weighting function g(p). By Fourier transforming it term by term and using the fact that $F(m) = F(-m)$, we get

$$g(p) = \frac{1}{2\pi} \left\{ F(0) + 2 \sum_{n=1}^{\bar{n}-1} F(m) \cos mp \right\}, \quad |p| \leq \pi \tag{10}$$

This is the (spatial) Fourier series of Taylor's weighting function. By computing once and for all the constants $F(0), F(1), \ldots, F(\bar{n}-1)$ using (8), the series (9) can be an efficient formula for computation.

The beamwidth measured between the first nulls is (from (6) with n=1)

$$BW_{NULL} = 2\sigma \sqrt{A^2 + \frac{1}{4}}$$

where σ and A are given as above. An exact formula for the half-power beamwidth is not available. Table 2 gives half-power beamwidths that were computed numerically (using a general purpose subroutine in [3, Chapter 7]). More useful perhaps is the following approximate formula for the half-power beamwidth

$$BW_{3dB} \doteq \left\{ \frac{\pi^2}{12} + \frac{1}{2} \sum_{n=1}^{\bar{n}-1} \left(\frac{1}{\sigma^2 \left(A^2 + \left(n - \frac{1}{2}\right)^2\right)} - \frac{1}{n^2} \right) \right\}^{-1/2} \tag{11}$$

To prove (11), note that the asymptotic expansion

$$F(z) = 1 - \left\{ \sum_{n=1}^{\bar{n}-1} \frac{1}{\sigma^2 \left(A^2 + \left(n - \frac{1}{2}\right)^2\right)} + \sum_{n=\bar{n}}^{\infty} \frac{1}{n^2} \right\} z^2 \ , \ z \to 0$$

follows immediately from (7). Since

$$\sum_{n=\bar{n}}^{\infty} \frac{1}{n^2} = \sum_{n=1}^{\infty} \frac{1}{n^2} - \sum_{n=1}^{\bar{n}-1} \frac{1}{n^2} = \frac{\pi^2}{6} - \sum_{n=1}^{\bar{n}-1} \frac{1}{n^2} \ ,$$

we have

$$F(z) = 1 - \frac{\pi^2}{6} + \sum_{n=1}^{\bar{n}-1} \left(\frac{1}{\sigma^2 \left(A^2 + \left(n - \frac{1}{2}\right)^2\right)} - \frac{1}{n^2} \right) z^2 \ , \ z \to 0 \ .$$

Setting $F(z) = 1/2$ and solving for z gives (11).

The accuracy of (11) is good in two limiting cases. As $\bar{n} \to \infty$, $\sigma \to 1$ and (11) becomes

$$BW_{3dB} \doteq \left\{ \frac{1}{2} \sum_{n=1}^{\infty} \frac{1}{A^2 + \left(n - \frac{1}{2}\right)^2} \right\}^{-1/2} \ . \tag{12}$$

The exact answer for the van der Maas function is

$$BW_{3dB} = 2 \left\{ A^2 - \left(\frac{1}{\pi} \operatorname{arc\,cosh} \left(\frac{1}{2} \cosh \pi A \right) \right)^2 \right\}^{1/2}$$

and a comparison with (12) is given in the last row in Table 3. Similarly, for $\bar{n} = 1$, the sum in (11) vanishes and

$$BW_{3dB} \doteq \frac{2\sqrt{3}}{\pi} = 1.103 \text{ radians}$$

10

which is within 10 percent of the correct answer of $BW_{3dB} = 1.207$ radians for the uniformly weighted aperture.

Table 3 gives the relative error between the approximation (11) and the exact half-power beamwidth for the same entries as in Table 2. It may be concluded from Table 3 that

(a) the approximation (11) is always on the low side of the exact half-power beamwidth, and

(b) the correction required to make (11) exact is a constant factor which depends strongly on the specified sidelobe level and very weakly on $\bar{n}$.

Consequently, a suitable correction factor depending only on specified sidelobe level would make (11) very accurate.

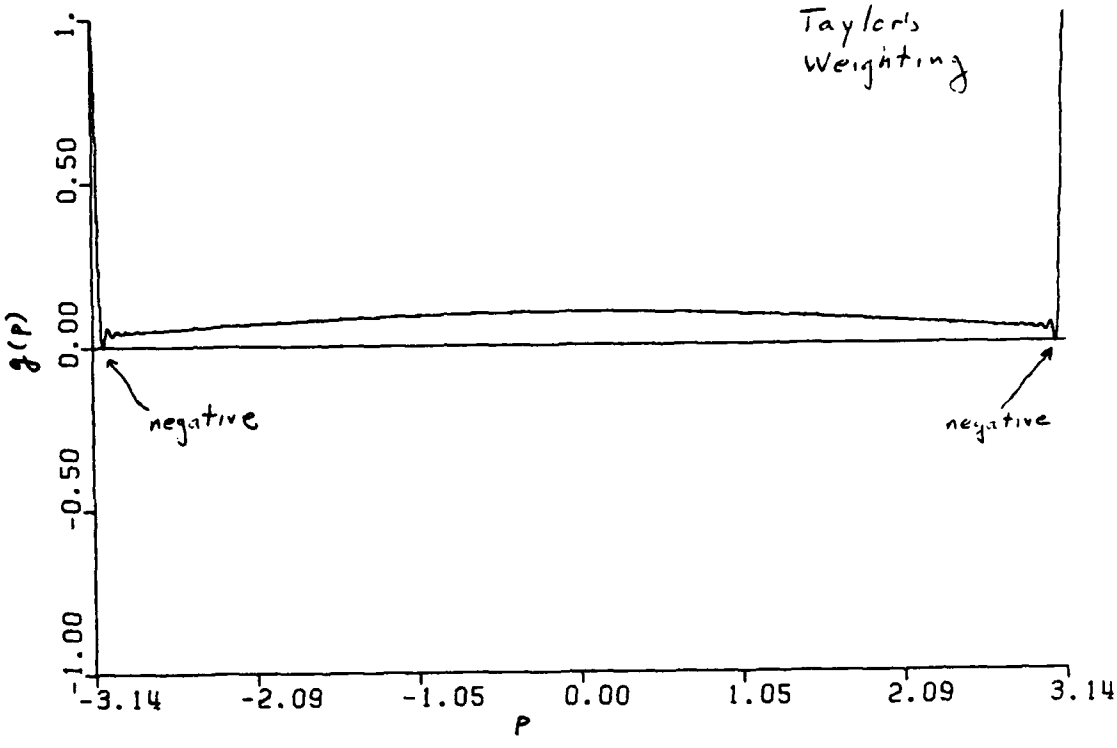
The Taylor weighting function need not always be a positive function. The best way to show this is by example. Consider the case $\bar{n} = 100$ and a sidelobe level of $S = -20$ dB. The weighting function is slightly negative just inside the aperture endpoints (for $p = \pm 3.078761$, for example, Taylor's weight is -.005519929). See Figure 2. The Taylor function in practice is nearly always positive for smaller values of $\bar{n}$.

$\bar{n}$	-10dB (A = .578)	-20dB (A= .953)	-30dB (A= 1.32)	-40dB (A=1.69)
5	1.0475	1.3264	1.5526	1.7323
10	1.0009	1.2818	1.5220	1.7262
15	.9851	1.2641	1.5051	1.7126
20	.9771	1.2548	1.4954	1.7036
25	.9724	1.2491	1.4892	1.6975
30	.9692	1.2452	1.4849	1.6932
100	.9581	1.2313	1.4691	1.6761
∞	.9533	1.2252	1.4619	1.6680

Table 2. Exact Taylor half-power beamwidths

$\bar{n}$	-10dB (A=.578	-20dB (A=953)	-30dB (A=1.32)	-40dB (A=1.69)
5	7.67 %	9.98 %	11.4 %	12.2%
10	7.57 %	9.91 %	11.3 %	12.2%
15	7.55 %	9.90 %	11.3 %	12.2%
20	7.54 %	9.89 %	11.3 %	12.2%
25	7.55 %	9.89 %	11.3 %	12.2%
30	7.54 %	9.89 %	11.3 %	12.2%
100	7.55 %	9.88 %	11.3 %	12.1%
∞	7.54 %	9.88 %	11.3 %	12.1%

Table 3. Relative error of approximation (11) to Taylor half-power beamwidths.



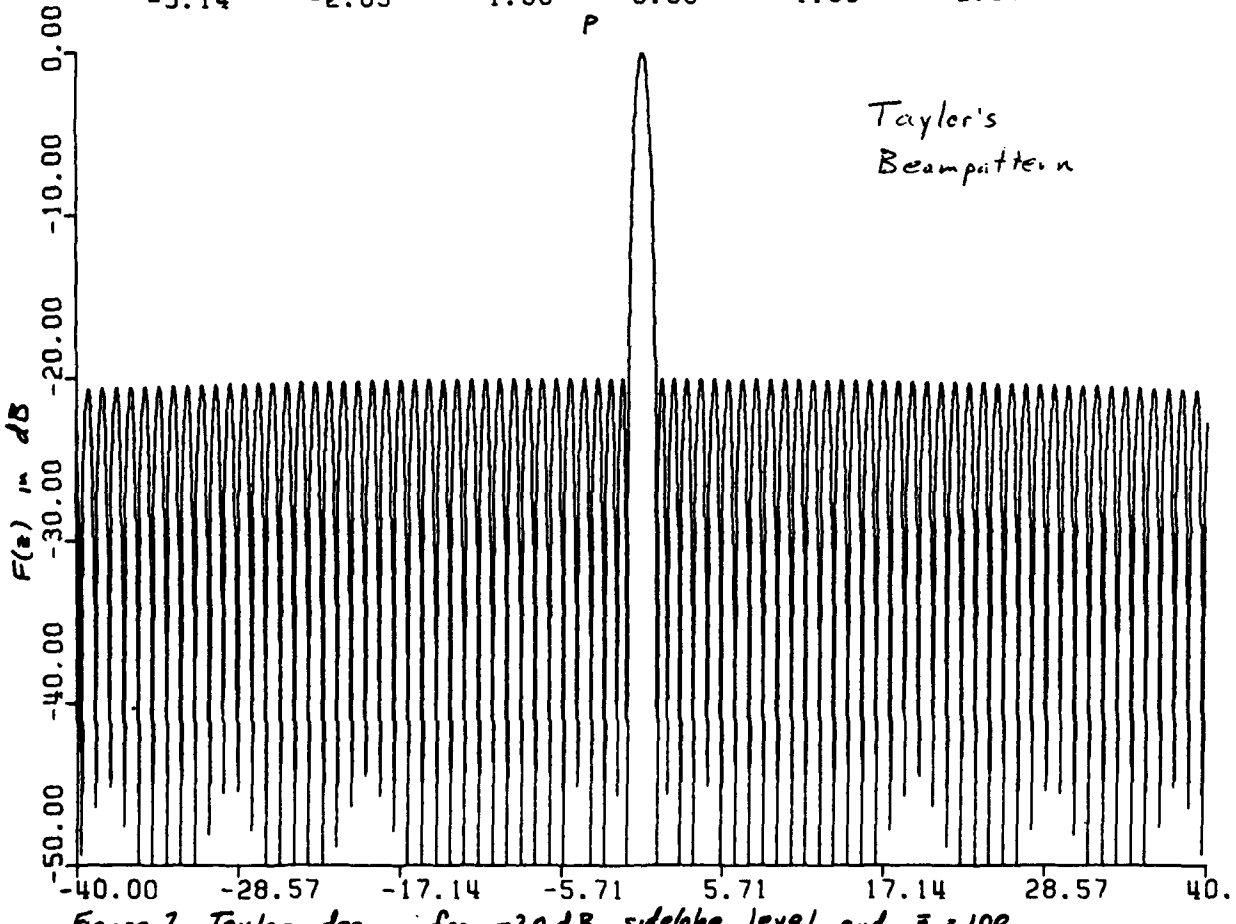


Figure 2. Taylor design for -20 dB sidelobe level and $\bar{n} = 100$.

12

IV. MODIFICATIONS OF TAYLOR WEIGHTING

Rhodes [4,5] shows that the Taylor weighting function g(p) can be made to go to zero as any power $\alpha > -1$ of distance from the aperture endpoints by altering the position of the nulls in Taylor's function F(z). The general design technique can be viewed as an extension of certain ideas in Taylor's original paper [1], using mathematical methods developed by Rhodes. The most important cases are

1. $\alpha = 0$, which is exactly Taylor's original case; F(z) decays asymptotically at 6 dB per octave.

2. $\alpha = 1$, for which the weighting function goes to zero linearly at the aperture endpoints; F(z) decays asymptotically at 12 dB per octave.

3. $\alpha = 2$, for which the weighting function goes to zero quadratically at the aperture endpoints; F(z) decays asymptotically at 18 dB per octave.

The cases $\alpha = 1$ and $\alpha = 2$ are given explicitly below, after giving the method for any $\alpha > -1$.

A theoretically significant criticism of Rhodes' work is that he does not prove that his technique is mathematically correct. The available theory (due to Paley and Wiener, and to Levinson) provides a proof only for $-1/2 < \alpha < 1/2$ and $\alpha=1$. As Rhodes states [5], "it is not unreasonable to expect that the general theory" is valid for all $\alpha > -1$. In any event, we can proceed to develop the method for all $\alpha > -1$ in a purely formal way, ignoring a theoretical question which may in the end not be of any practical importance. Taylor's original method is, after all, an ad hoc technique and so is Rhodes' generalization of it.

Rhodes' development retains the integer $\bar{n}$ as the breakpoint between the near-in nulls, which are dilated versions of van der Maas' nulls, and the outer nulls, which force the asymptotic decay rate for F(z) to be $6(1+\alpha)$ dB per octave. Consequently, in the limit as $\bar{n}\to\infty$, the van der Maas function is again obtained for all $\alpha > -1$, just as in Taylor's original design $\alpha = 0$. This means that the desired behavior of the weighting function at the aperture endpoints is confined to small neighborhoods of the aperture endpoints for larger $\bar{n}$. In other words, the weighting function changes rapidly just inside the aperture endpoints for large $\bar{n}$.

The development in [4] is brief and only the case $\alpha = 1$ is given in any detail. His later paper [5] gives enough detail to carry out the general development for $\alpha > -1$. This requires the identity, valid for $\alpha > -1$,

$$T_\alpha(z) = \prod_{n=1}^{\infty} \left(1 - \frac{z^2}{(n + \frac{\alpha}{2})^2}\right) = \Gamma^2 (1 + \alpha/2) \frac{\Gamma(z + 1 - \alpha/2)}{\Gamma(z + 1 + \alpha/2)} \frac{\sin \pi(z - \alpha/2)}{\pi(z - \alpha/2)} \quad (13)$$

It is proved as follows. A special case ($z_1 = z_2 = \alpha/2$ and $z_3 = z$) of a result in [6, Equ. 1.3(4)] gives

$$\frac{\Gamma^2(\alpha/2)}{\Gamma(z + \alpha/2)\,\Gamma(-z + \alpha/2)} = \prod_{n=0}^{\infty} \left(1 - \frac{z^2}{(n + \frac{\alpha}{2})^2}\right).$$

Dividing by the first term in the infinite product, and then using the recurrence formula [6, Equ. 6.1.15] and the reflection formula [6, Equ. 6.1.17] whenever necessary, gives

$$T_\alpha(z) = \frac{(\alpha/2)^2\,\Gamma^2(\alpha/2)}{[(\alpha/2)^2 - z^2]\,\Gamma(\alpha/2 + z)\,\Gamma(\alpha/2 - z)}$$

$$= \frac{\Gamma^2(1 + \alpha/2)}{\Gamma(1 + z + \alpha/2)\,\Gamma(1 - z + \alpha/2)}$$

$$= \frac{\Gamma^2(\frac{\alpha}{2} + 1)}{\Gamma(1 + \frac{\alpha}{2} + z)} \frac{\Gamma(z - \frac{\alpha}{2})}{\Gamma(z - \frac{\alpha}{2})} \frac{1}{\Gamma(1 - (z - \frac{\alpha}{2}))}$$

$$= \frac{\Gamma^2(\frac{\alpha}{2} + 1)\,\Gamma(z - \frac{\alpha}{2})}{\Gamma(z + \frac{\alpha}{2} + 1)} \frac{\sin \pi(z - \frac{\alpha}{2})}{\pi} \qquad (14)$$

Multiplying and dividing by $z - (\alpha/2)$ on the right hand side of the last equation yields (13). (We note that above is given without proof by Taylor [1, Equ. (29)].)

Rhodes defines the general Taylor pattern, $F_\alpha(z)$, for $\alpha > -1$ to be

$$F_\alpha(z) = \prod_{n=1}^{\bar{n}-1} \left(1 - \frac{z^2}{\sigma_\alpha^2(A^2 + (n - \frac{1}{2})^2)}\right) \prod_{n=\bar{n}}^{\infty} \left(1 - \frac{z^2}{(n + \alpha/2)^2}\right) \qquad (15)$$

where

$$\sigma_\alpha = (\bar{n} + \tfrac{\alpha}{2}) \Big/ \sqrt{A^2 + (\bar{n} - \tfrac{1}{2})^2} \qquad (16)$$

and A is the same as given above (just after (6)). Note that for $\alpha = 0$ the function $F_0(z)$ is exactly Taylor's original function $F(z)$. The analog of (8) for general α is

$$F_\alpha(z) = \prod_{n=1}^{\bar{n}-1} \left(\frac{1 - \dfrac{z^2}{\sigma_\alpha^2 (A^2 + (n - \frac{1}{2})^2)}}{1 - \dfrac{z^2}{(n + \frac{\alpha}{2})^2}} \right) T_\alpha(z) \tag{17}$$

as is clear from (15) and (13). As $z \to \infty$, the rational function of degree $n-1$ in z^2 in (17) approaches a constant and $T_\alpha(z)$ is asymptotic to a constant (depending only on α) times $1/|z|^{1+\alpha}$. The asymptotic decay rate of $F_\alpha(z)$ is therefore $6(1 + \alpha)$ dB per octave. In addition, the asymptotic decay rate means that $F_\alpha(z)$ has a Fourier transform vanishing outside the aperture $[-\pi, \pi]$ for every $\alpha > -1$ and $\bar{n} \geq 1$.

For $\alpha > -1$ define the "sampling functions"

$$G_n^{(\alpha)}(z) = c_n(\alpha) \frac{\Gamma(z - \frac{\alpha}{2} + 1)}{\Gamma(z + \frac{\alpha}{2})} \frac{\sin \pi(z - \frac{\alpha}{2})}{\pi(z^2 - (n + \frac{\alpha}{2})^2)} \tag{18}$$

where

$$c_n(\alpha) = \begin{cases} (-1)^n (2n + \alpha)(n + \alpha)/n! & , \text{ if } \alpha \neq 0 \\ 1 & , \text{ if } \alpha = 0,\ n = 0 \\ (-1)^n 2 & , \text{ if } \alpha = 0,\ n = 1,2,3,\ldots \end{cases}$$

Each $G_n^{(\alpha)}(z)$ is an even function of z. These functions are essentially the Lagrange interpolating functions for the points $\{\pm(n + \alpha/2);\ n = 0,1,2,\ldots\}$. More precisely, the only nulls of $G_n^{(\alpha)}(z)$ are of the form $\pm(n + \alpha/2)$ and, furthermore,

$$G_n^{(\alpha)}(\pm(m + \tfrac{\alpha}{2})) = \begin{cases} 1, & \text{if } m = n \\ 0, & \text{if } m \neq n. \end{cases} \tag{19}$$

The functions $G_n^{(\alpha)}(z)$ are derived using methods due originally to Paley and Wiener.

The open theoretical question mentioned earlier in this section concerns the completeness of the sampling functions (18) with respect to all even aperture-limited functions. As stated already, it is known that $G_n^{(\alpha)}$ are complete for $-1/2 < \alpha < 1/2$ and for $\alpha = 1$. For other values of $\alpha > -1$, nothing is known. Proceeding on the assumption that $G_n^{(\alpha)}$ are complete for all $\alpha > -1$, it follows $F_\alpha(z)$ can be expanded in the form

$$F_\alpha(z) = \sum_{n=0}^{\infty} a_n G_n^{(\alpha)}(z)$$

for some constants a_n. From (19) it follows that $a_n = F_\alpha(n + (\alpha/2))$ for all n. From (15) it follows that $a_n = 0$ for $n \geq \bar{n}$. Therefore,

$$F_\alpha(z) = \sum_{n=0}^{\bar{n}-1} F_\alpha(n + \tfrac{\alpha}{2})\, G_n^{(\alpha)}(z) , \tag{20}$$

which generalizes (9).

Denote by $g_\alpha(p)$ the weighting function corresponding to $F_\alpha(z)$. As just stated, $g_\alpha(p)$ vanishes outside the aperture $[-\pi, \pi]$. Question: Does $g_\alpha(p)$ go to zero as the power $\alpha > -1$ of distance from the aperture endpoints? Taylor [1] proves that any even function with this endpoint behavior has a Fourier transform whose nulls are asymptotic to $\pm(n + (\alpha/2))$, but he does NOT prove the converse. Consequently, although $g_\alpha(p)$ is even and has a Fourier transform with the proper null locations, this is not necessarily sufficient to answer the question in the affirmative. However, taking the term by term Fourier transform of (20) gives the expansion

$$g_{(\alpha)}(p) = \sum_{n=0}^{\bar{n}-1} F_\alpha(n + \tfrac{\alpha}{2})\, H_n^{(\alpha)}(p) \tag{21}$$

where for $n = 0, 1, 2, \ldots$

$$H_n^{(\alpha)}(p) = \begin{cases} (2\cos\tfrac{p}{2})^\alpha \sum_{r=0}^{n} (-1)^{n-r} \dfrac{\epsilon_r}{2} \dfrac{(\alpha)_{n-r}}{(n-r)!} \cos(rp), & |p| < \pi \\ 0, & |p| \geq \pi \end{cases} \tag{22}$$

where $\epsilon_0 = 1$ and $\epsilon_r = 2$ for $r > 0$, and

$$(\alpha)_k = \begin{cases} 1 \quad \text{for all } \alpha, \text{ if } k = 0 \\ (\alpha + 1) \ldots (\alpha + k - 1) \text{ for all } \alpha, \text{ if } k > 0. \end{cases}$$

Since each of the functions $H_n^{(\alpha)}(p)$ has the correct endpoint behavior, $g_\alpha(p)$ must also have this same behavior.

Just as in Taylor's original case, both the aperture function $g_\alpha(p)$ and the beampattern function $F_\alpha(z)$ can be computed independently of each other using the analytically exact formulas (21) and (17), respectively, for any $\alpha > -1$. Appropriate approximations near the points $\pm(n + \alpha/2)$ analogous to those developed in Appendix B for $\alpha = 0$, are necessary for computing $F_\alpha(z)$ using (17). Developing these approximations should not present any mathematical difficulties.

The three cases $\alpha = 0, 1, 2$ are now given explicitly. Fortran programs implementing these three cases should be easy to write. The sampling functions are

$$G_n^{(0)}(z) = (-1)^n \frac{\epsilon_n}{2} \frac{2z \sin \pi z}{\pi (z^2 - n^2)} \tag{23}$$

$$G_n^{(1)}(z) = (-1)^{n+1} \frac{(2n+1) \cos \pi z}{\pi (z^2 - (n+\frac{1}{2})^2)} \tag{24}$$

$$G_n^{(2)}(z) = (-1)^{n+1} \frac{2(n+1)^2 \sin \pi z}{\pi z (z^2 - (n+1)^2)} . \tag{25}$$

and the corresponding aperture basis functions are

$$H_n^{(0)}(p) = \frac{\epsilon_n}{2} \cos np \tag{26}$$

$$H_n^{(1)}(p) = \cos (n+\tfrac{1}{2})p \tag{27}$$

$$H_n^{(2)}(p) = (-1)^n + \cos (n+1)p \quad . \tag{28}$$

It should be noted that (23) and (26) are, within a scale factor, identical to (9) and (10), respectively. In all cases, the aperture function $g_\alpha(p)$ is computed from (21). Consequently, the only potential difficulty is computing the constants $F_\alpha(n + \alpha/2)$ for is $n = 0, 1, \ldots, \bar{n} - 1$. Fortunately, for $\alpha = 0$, 1, and 2, these constants are easy to compute using (17) since the following identities hold:

$$T_0(z) = \frac{\sin \pi z}{\pi z} \tag{29}$$

$$T_1(z) = \frac{\cos \pi z}{1 - 4z^2} \tag{30}$$

$$T_2(z) = \frac{\sin \pi z}{\pi z (1-z^2)} \quad . \tag{31}$$

The price paid for the desired end effects is an increase in the beamwidth over the beamwidth of Taylor's original weighting function. The beamwidth measured to the first null is, for all $\alpha > -1$,

$$BW_{NULL} = \sigma_\alpha \sqrt{A^2 + \frac{1}{4}}$$

where σ_α is given by (16) above. This gives exactly, for fixed $\bar{n}$,

$$\frac{BW_{NULL} \text{ (for any } \alpha)}{BW_{NULL} \text{ (for } \alpha = 0)} = \frac{\sigma_\alpha}{\sigma_0} = 1 + \frac{\alpha}{2\bar{n}} .$$

This means that for $\bar{n} = 10$, the beamwidth measured between nulls is 5% larger for $\alpha = 1$ and 10% larger for $\alpha = 2$ than for Taylor's original $\alpha = 0$ beampattern. It is anticipated that approximately the same percentage increases occur in the half-power beamwidths.

A different modification to Taylor's nulls can be utilized to produce asymmetric beampatterns using complex valued aperture functions g(p). This is described in [8] for an application in radar to minimize ground clutter. The magnitude of the aperture function turns out to be an even function, while the phase of the aperture function turns out to be odd.

V. CONCLUSIONS

Taylor weighting can be modified to force the weighting function to go zero as any power $\alpha > -1$ of distance from the aperture endpoints. Taylor's original weighting ($\alpha = 0$) results in a pedestal, while for $\alpha = 1$ the weighting function goes to zero linearly as in a cosine window, and for $\alpha = 2$ the weighting function behaves like a cosine-squared window at the aperture endpoints. The endpoint effect is achieved for a modest increase in the mainlobe beamwidth.

```
c       COMPUTE TAYLOR'S CONTINUOUS SHADING FUNCTION AND TRANSFER FUNCTION
c       INPUT REQUIREMENT: 1 .LE. NBAR
c                        : THE SPATIAL APERTURE LIES FROM -PI TO +PI
c       ARGUMENT DEFINITIONS:
c               x  = abscissas for sampling the shading function
c               s  = shading function values
c               ns = number of s samples; none computed if ns = 0
c               k  = abscissas for sampling the transfer function
c               f  = transfer function values
c               nk = number of f samples; none computed if nk = 0
c               nbar= the first nbar-1 zeros of f are those of van der Maas
c               db = sidelobe level in dB of limiting Dolph-Chebyshev array
c               fm = coefficients of the shading function cosine series
c               bw3db = -3 dB beamwidth; not computed if bw3db is set to -1.
c       DIMENSION LIMITS:
c           x and s must be dimensioned at least max(ns,+1)
c           k and f must be dimensioned at least max(nk,+1)
c           fm must be dimensioned at least nbar
c       TECHNICAL NOTES:
c             NBAR=1 GIVES THE UNIFORM SHADING FUNCTION
c             NBAR=INFINITY GIVES DOLPH-CHEBYSHEV SHADING
c             FIRST BEAMPATTERN NULL = SIGMA * SQRT(A**2+.25)
c             THE COSINE SERIES FOR S HAS DEGREE EXACTLY NBAR-1
c       PROGRAMMER: R. B. STREIT, NUSC, DECEMBER 21, 1984.
c                LAST REVISION: JANUARY 11, 1985
c
      subroutine taylor(x,s,ns,k,f,nk,nbar,db,fm,bw3db)
      double precision x(1),s(1),k(1),f(1),db,fm(1),xpt,a,sigma,q,pi,q,
     +          zeroin,ap,bp,tol,bw3db
      data pi/3.14159265358979d0/
      a=10.0d0**abs(db/20.0d0)
      a=dlog(a+sqrt(a*a-1.0d0))/pi
      sigma=nbar/sqrt(a*a+(nbar-.5d0)*(nbar-.5d0))
      nbar1=nbar-1
      fm(1)=1.0d0
      if(nbar.eq.1)go to 15
      do 10 i=2,nbar
      xpt=i-1
      fm(i)=g(xpt,a,nbar1,sigma)
   10 continue
   15 if(ns.le.0)go to 25
      do 20 i=1,ns
      s(i)=0.0d0
      if(abs(x(i)).gt.(pi+1.d-7))go to 20
      s(i)=g(x(i),fm,nbar)
   20 continue
   25 if(nk.le.0)go to 35
      do 30 i=1,nk
      f(i)=g(k(i),a,nbar1,sigma)
   30 continue
   35 if(bw3db.lt.0.0d0)go to 40
      ap=0.0d0
      bp=sigma*sqrt(a*a+.25d0)
      tol=1.0d-10
      bw3db=2.0d0*zeroin(ap,bp,g,tol,a,nbar1,sigma)
   40 continue
      return
      end
      double precision function g(z,a,nbar1,sigma)
      double precision pi,piz,zn,z,a,sigma
      data pi/3.14159265358979d0/
```

19

```
      do 10 k=0,nbar1
      if(abs(z-k).lt.1.0d-4)go to 30
10    continue
      a=1.0d0
      if(nbar1.eq.0)go to 21
      do 20 n=1,nbar1
      zn=(z/sigma)**2/(a*a+(n-.5d0)**2)
      a=a*(1.0d0-zn)/(1.0d0-(z/n)**2)
20    continue
21    piz=pi*z
      a=(sin(piz)/piz)*a
      return
30    if(k.gt.0)go to 50
      a=1.0d0
      if(nbar1.eq.0)go to 41
      do 40 n=1,nbar1
      zn=(z/sigma)**2/(a*a+(n-.5d0)**2)
      a=a*(1.0d0-zn)/(1.0d0-(z/n)**2)
40    continue
41    piz=pi*z
      a=(1.0d0-piz*piz*(1.0d0-piz*piz/20.d0)/6.0d0)*a
      return
50    a=1.0d0
      do 60 n=1,nbar1
      if(n.eq.k)go to 60
      zn=(z/sigma)**2/(a*a+(n-.5d0)**2)
      a=a*(1.0d0-zn)/(1.0d0-(z/n)**2)
60    continue
      zn=(z/sigma)**2/(a*a+(k-.5d0)**2)
      a=a*(1.0d0-zn)
      piz=pi*(z-k)
      a=(1.0d0-piz*piz*(1.0d0-piz*piz/20.d0)/6.0d0)*a
      a=a*(-1)**(k+1)*(k/(z+z*z/k))
      return
      end
      double precision function g(p,fm,nbar)
      double precision p,twopi,fm(1)
      data twopi/6.2831853071795864769d0/
      g=0.0d0
      if(nbar.eq.1)go to 20
      do 10 i=2,nbar
      g=g+fm(i)*cos((i-1)*p)
10    continue
20    g=(fm(1)+g+g)/twopi
      return
      end
c
c     Compute a zero of a real function f in the interval [ax, bx].
c     Double precision version of program on pp. 164-166 of "Computer
c     Methods for Mathematical Computations," by G.E. Forsythe,
c     M.A. Malcolm, and C.B. Moler, Prentice-Hall, 1977, but slightly
c     altered for use in computing Taylor's half-power beamwidth.
      double precision function zeroin(ax,bx,f,tol,adum,nbar1,sigma)
      double precision ax,bx,f,tol,adum,sigma
      double precision a,b,c,d,e,eps,fa,fb,fc,tol1,xm,p,q,r,s
      eps=1.0d0
10    eps=eps/2.0d0
      tol1=1.0d0+eps
      if(tol1.gt.1.0d0)go to 10
      a=ax
      b=bx
20
```

```
      fa=f(a,adum,nbar1,sigma)-.5d0
      fb=f(b,adum,nbar1,sigma)-.5d0
20    c=a
      fc=fa
      d=b-a
      e=d
30    if(abs(fc).ge.abs(fb))go to 40
      a=b
      b=c
      c=a
      fa=fb
      fb=fc
      fc=fa
40    tol1=2.0d0*eps*abs(b)+.5d0*tol
      xm=.5d0*(c-b)
      if(abs(xm).le.tol1)go to 90
      if(fb.eq.0.0d0)go to 90
      if(abs(e).lt.tol1)go to 70
      if(abs(fa).le.abs(fb))go to 70
      if(a.ne.c)go to 50
      s=fb/fa
      p=2.0d0*xm*s
      q=1.0d0-s
      go to 60
50    q=fa/fc
      r=fb/fc
      s=fb/fa
      p=s*(2.0d0*xm*q*(q-r)-(b-a)*(r-1.0d0))
      q=(q-1.0d0)*(r-1.0d0)*(s-1.0d0)
60    if(p.gt.0.0d0)q=-q
      p=abs(p)
      if((2.0d0*p).ge.(3.0d0*xm*q-abs(tol1*q)))go to 70
      if(p.ge.abs(0.5d0*e*q))go to 70
      e=d
      d=p/q
      go to 80
70    d=xm
      e=d
80    a=b
      fa=fb
      if(abs(d).gt.tol1)b=b+d
      if(abs(d).le.tol1)b=b+sign(tol1,xm)
      fb=f(b,adum,nbar1,sigma)-.5d0
      if((fb*(fc/abs(fc))).gt.0.0d0)go to 20
      go to 30
90    zeroin=b
      return
      end
```

21

Appendix B. Calculation of F(z)

For some small number $\epsilon > 0$, say $\epsilon = 10^{-4}$, define

$$s(z, k) = \begin{cases} \dfrac{\sin \pi z}{\pi(z-k)}, & |z-k| > \epsilon > 0 \\ (-1)^k \left[1 - \dfrac{\pi^2}{6}(z-k)^2 + \dfrac{\pi^4}{120}(z-k)^4 - \ldots\right], & |z-k| < \epsilon. \end{cases}$$

Now, if $|z-k| > \epsilon$ for $k = 0,1,2,\ldots, \bar{n}-1$, compute F(z) exactly as in (8). If $|z-k| \leq \epsilon$ for $k=0$, then compute

$$F(z) = s(z,0) \prod_{n=1}^{\bar{n}-1} \left(\frac{1 - \dfrac{z^2}{\sigma^2 (A^2 + (n-\frac{1}{2})^2)}}{1 - \dfrac{z^2}{n^2}} \right).$$

If $|z-k| \leq \epsilon$ for $k=1, \ldots, \bar{n}-1$, then compute

$$F(z) = s(z,k) \frac{(-k)\left(1 - \dfrac{1}{\sigma^2 (A^2 + (k-\frac{1}{2})^2)}\right)}{z(1 + \frac{z}{k})} \prod_{\substack{n=1 \\ n \neq k}}^{\bar{n}-1} \left\{ \frac{1 - \dfrac{z^2}{\sigma^2 (A^2 + (n-\frac{1}{2})^2)}}{1 - \dfrac{z^2}{n^2}} \right\}.$$

Use of these formulae eliminates all indeterminate 0/0 forms that arise during actual computation using (8).

REFERENCES

1. T. T. Taylor, "Design of Line-Source Antennas for Narrow Beamwidth and Low Side Lobes", IRE Trans. on Ant. and Prop., vol. AP-3, pp. 16-28, Jan 1955.

2. V. Barcilon and G. Temes, "Optimum Impulse Response and the van der Maas Function," IEEE Trans. on Circuit Theory, vol. CT-19, July 1972, pp. 336-342.

3. G. E. Forsythe, M.A. Malcolm, and C. B. Moler, Computer Methods for Mathematical Computations, Prentice-Hall, 1977.

4. D. R. Rhodes, "On the Taylor Distribution," IEEE Trans. on Ant. and Prop., Vol AP-20, March 1972, pp. 143-145.

5. D. R. Rhodes, "A General Theory of Sampling Synthesis," IEEE Trans. on Ant. and Prop., Vol AP-21, March 1973, pp. 176-181.

6. A. Erdelyi, Editor, Higher Transcental Functions, Vol. I, McGraw-Hill, 1953.

7. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, NBS Applied Mathematics Series, Vol 55, US Dept of Commerce, 1972.

8. R. S. Elliott, "Design of Line Source Antennas for Narrow Beamwidth and Asymmetric Low Sidelobes," IEEE Trans. on Ant. and Prop., vol. AP-23, Jan. 1975, pp. 100-107.

Sufficient Conditions For The Existence Of Optimum Beam Patterns For Unequally Spaced Linear Arrays With An Example

R. L. Streit

Sufficient Conditions for the Existence of Optimum Beam Patterns for Unequally Spaced Linear Arrays with an Example

ROY STREIT

Abstract—Dolph's method for determining the optimum element currents for half-wavelength equispaced discrete linear arrays is generalized to symmetric discrete linear arrays. The theorem proved gives sufficient conditions for the existence of optimum beam patterns for arrays with elements symmetrically positioned about the array center, but with fixed unequal spacings between the elements. The conditions are such that the Remes exchange algorithm for minimax approximation of functions can be employed to compute the optimum element currents corresponding to an optimum beam pattern directly from the given spacings of the elements. Half-wavelength spaced linear arrays satisfy the conditions of the theorem; therefore, it provides a new method of calculating the well-known Dolph–Chebyshev element currents. An example with unequal spacings is included to show the utility of the method even when the hypotheses of the theorem may not be met.

I. INTRODUCTION

Optimum beam patterns and element currents for single frequency linear arrays with a finite number of omnidirectional half-wavelength spaced elements were determined by Dolph [1] through a technique involving the Chebyshev polynomials. All these beam patterns have equal amplitude sidelobes. Sufficient conditions are given here for symmetric linear arrays to possess optimum beam patterns with equal amplitude sidelobes. This feature is precisely the fact needed in the calculation of the element currents.

The definition of an optimum beam pattern used in Dolph's paper will be used: a beam pattern is optimum if, for a given main lobe beamwidth, the overall sidelobe amplitude is minimized. Beamwidth is measured from the maximum response axis to the first null. The linear arrays considered in this paper are those whose elements are symmetrically spaced and have symmetrically tapered element currents about the center of the array.

Manuscript received January 10, 1974; revised August 17, 1974.
The author is with the Naval Underwater Systems Center, New London Laboratory, New London, Conn. 06320.

II. PRELIMINARIES

Let $f(x)$ be a real valued continuous function defined on the closed interval $[a,b]$. The norm of $f(x)$, denoted $\|f\|_{[a,b]}$, is defined to be

$$\|f\|_{[a,b]} = \max_{a\le x\le b} |f(x)|.$$

Now let $h_1(x),\cdots,h_N(x)$ be a given finite collection of real valued continuous functions defined on the closed interval $[a,b]$. The linear span of these basis functions is a proper closed subspace of the space of all continuous functions on the interval $[a,b]$ equipped with this max norm. It is known that there exist real constants $\alpha_1,\cdots,\alpha_N$ such that

$$\|f(x) - \sum_{i=1}^{N} \alpha_i h_i(x)\|_{[a,b]}$$

is a minimum. The function $h(x) = \sum_{i=1}^{N} \alpha_i h_i(x)$ is defined to be a minimiax approximation to the function $f(x)$ from the basis $h_1(x),\cdots,h_N(x)$. The crucial property that these basis functions must satisfy to guarantee the uniqueness of a minimax approximation is embodied in the definition: the functions $h_1(x),\cdots,h_N(x)$ form a Chebyshev basis of degree N on the closed interval $[a,b]$, if and only if every nontrivial linear combination of these functions possesses at most $N-1$ real roots in the interval $[a,b]$. A particularly well-known Chebyshev basis is the collection $1,x,\cdots,x^{N-1}$ on any finite or infinite interval. It is possible that a given collection of functions may be a Chebyshev basis on one interval but not on another. It can be shown that the functions $h_1(x),\cdots,h_N(x)$ form a Chebyshev basis on the interval $[a,b]$, if and only if the determinants

$$U(x_1,\cdots,x_N) = \begin{vmatrix} h_1(x_1) & h_1(x_2) & \cdots & h_1(x_N) \\ h_2(x_1) & h_2(x_2) & \cdots & h_2(x_N) \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ h_N(x_1) & h_N(x_2) & \cdots & h_N(x_N) \end{vmatrix} \neq 0$$

for all points x_i, such that

$$a \le x_1 < x_2 < \cdots < x_{N-1} < x_N \le b.$$

The reader is referred to Karlin and Studden [2] for a proof of this and other equivalent formulations of a Chebyshev basis, as well as for a proof of the following fundamental theorem.

Theorem

Let $h_1(x),\cdots,h_N(x)$ be a Chebyshev basis on the interval $[a,b]$. Then $h(x) = \sum_{i=1}^{N} \alpha_i h_i(x)$, for some real constants α_i, is a minimax approximation to $f(x)$ on $[a,b]$, if and only if there exist at least $N+1$ points $a \le x_0 < x_1 < \cdots < x_M \le b$, $M > N$, such that

$$f(x_i) - h(x_i) = \pm\|f - h\|_{[a,b]}, \qquad i = 0,\cdots,M$$

and the sign of the error alternates from point to point. Furthermore, the approximating function $h(x)$ is unique.

In other words there exists exactly one linear combination of the N Chebyshev basis functions which has at least $N+1$ points of equiripple (but alternately signed) error for a given function $f(x)$, and it is this linear combination that forms the unique minimax approximation to $f(x)$. If the basis functions do not form a Chebyshev basis, however, the minimax error curve need not be equiripple.

The Remes exchange algorithm [3] employs the equal oscillation error of the minimax approximation to compute the constants $\alpha_1,\cdots,\alpha_N$. The algorithm is iterative and has been shown to converge under very general conditions.

III. THE SUFFICIENCY THEOREM

As stated in the introduction, every linear array considered is assumed to be symmetric and to have symmetrically tapered element currents about the array's center. If the center of an M element linear array is chosen as the origin of the coordinate system, then the field pattern, as a function of the angle measured from a normal to the array, is proportional to the absolute value of

$$\sum_{i=1}^{N} \alpha_i \cos\left(\frac{2\pi x_i}{\lambda} \sin\theta\right), \qquad 0 \le \theta \le 2\pi$$

where

$$N = \left[\frac{M+1}{2}\right]$$

λ wavelength of design frequency
x_i distance of ith element (counted from the center of the array)
α_i current of the elements at x_i (if M is odd, α_1 is half the current).

Putting $u = \pi \sin\theta$ and restricting θ to $0 \le \theta \le \pi/2$ to utilize symmetry, the field pattern is proportional to the absolute value of

$$P(u) = \sum_{i=1}^{N} \alpha_i \cos(\xi_i u), \qquad 0 \le u \le \pi \tag{1}$$

where $\xi_i = x_i/(\lambda/2)$, $i = 1,\cdots,N$. We always have $0 \le \xi_1 < \xi_2 < \cdots \le \xi_N$. For the field pattern $P(u)$ defined in (1), we define the sidelobe level on the interval $[u_1,\pi)$ to be the norm $\| P(u) \|_{[u_1,\pi)}$, where $u_1 > 0$ is the first null of $P(u)$. We define the sidelobe ratio on the same interval to be the ratio $| P(0) | \div \| P(u) \|_{[u_1,\pi)}$. Note that both these terms are in linear units. Also, the symbol for the half-open interval $[a,b)$ is interpreted to mean the closed interval $[a,b-\epsilon]$, where ϵ is some preselected small positive number. We now state and prove the main result.

Sufficiency Theorem

Suppose that the functions

$$\cos(\xi_1 u),\cdots,\cos(\xi_N u) \tag{2}$$

form a Chebyshev basis on the interval $[0,\pi)$, and that the functions

$$\cos(\xi_1 u),\cdots,\cos(\xi_{N-1} u) \tag{3}$$

form a Chebyshev basis on the interval $[u_0,\pi)$ for some real number u_0, $0 \le u_0 < \pi$. Then there is an angle θ_0, $0 \le \theta_0 < \pi/2$, such that for any specified beamwidth θ, $\theta_0 \le \theta < \pi/2$, there exists a unique optimum field pattern. This optimum field pattern will have equal amplitude sidelobes.

Proof: Since the functions (3) form a Chebyshev basis on the interval $[u_0,\pi)$, there must exist a unique minimax approximation to the function $-\cos(\xi_N u)$ from this basis; that is, there exist constants $\alpha_1,\cdots,\alpha_{N-1}$ such that

$$-\cos\xi_N u \doteq \alpha_1 \cos\xi_1 u + \cdots + \alpha_{N-1}\cos\xi_{N-1}u.$$

Thus if e_0 is the magnitude of the maximum error committed by this uniform approximation, then the function

$$f(u) = \alpha_1 \cos\xi_1 u + \cdots + \alpha_{N-1}\cos\xi_{N-1}u + \cos\xi_N u$$

must oscillate about the zero function in the interval $[u_0,\pi)$ with the magnitude of the oscillation no greater than e_0 and with at least $(N-1)+1 = N$ points where the oscillation is exactly e_0. Let u_1 be the first zero of $f(u)$ greater than u_0 and let $\theta_0 = \sin^{-1}(u_1/\pi)$. We claim that $f(u)$ constitutes the optimum field pattern for a beamwidth to the first null of θ_0. For if this pattern is not optimum, there exists another field pattern function

$$g(u) = \beta_1 \cos\xi_1 u + \cdots + \beta_{N-1}\cos\xi_{N-1}u + \beta_N \cos\xi_N u$$

such that $g(0) = f(0)$, u_1 is the smallest positive root of $g(u)$, and g has a strictly smaller sidelobe level on the interval $[u_1,\pi)$. Since $f(u)$ has at least N points of maximum error on $[u_0,\pi)$, $f(u)$ has at least $N-1$ points of maximum error on $[u_1,\pi)$. It is clear that any function which is constrained to agree with f at $u = u_1$, which is everywhere strictly less than f on $[u_1,\pi)$, must intersect f in at least $N-1$ points in the interval $[u_1,\pi)$. Additionally, $f(0) = g(0)$, so that f and g must agree with at least N points in $[0,\pi)$. However, $f(u) - g(u)$ is then a linear combination of the functions (2), which has at least N zeros on $[0,\pi)$, contradicting the definition of a Chebyshev basis unless $f = g$. Thus f is the unique optimum field pattern for beamwidth of θ_0.

To complete the proof of the theorem, we need to demonstrate that for each angle $\theta > \theta_0$, there exists a number $\hat{u}_0$, $u_0 < \hat{u}_0 < \pi$, such that the function

$$f(u) = \gamma_1 \cos\xi_1 u + \cdots + \gamma_{N-1}\cos\xi_{N-1}u + \cos\xi_N u$$

has $\hat{u}_1 = \pi \sin\theta$ as its first real root greater than or equal to $\hat{u}_0$, where $\sum_{i=1}^{N-1}\gamma_i \cos\xi_i u$ is the uniform approximation to $-\cos\xi_N u$ on $[\hat{u}_0,\pi)$. Since $[\hat{u}_0,\pi) \subset [u_0,\pi)$, the collection of functions (3) must form a Chebyshev basis on all intervals $[\hat{u}_0,\pi)$, so that the function $f(u)$ is well-defined for each $\hat{u}_0$. Also $\hat{u}_0 > u_0$ implies that $\hat{u}_1 > u_1$ (u_1 as defined earlier) since otherwise the beam pattern for a beamwidth of θ_0 is not optimum. Finally, as $\hat{u}_0$ is varied continuously, the constants $\gamma_1,\cdots,\gamma_{N-1}$ vary continuously, so that the first real zero greater than $\hat{u}_0$ varies continuously. Since $\hat{u}_0$ may be taken as close to π as desired, it must be the case that for some $\hat{u}_0$ the first real zero of $f(u)$ greater than $\hat{u}_0$ is equal to $\pi \sin\theta$.

IV. DOLPH–CHEBYSHEV SHADINGS AS A SPECIAL CASE

As mentioned in the introduction, the Dolph–Chebyshev shadings are designed specifically for a half-wavelength equispaced line array. We consider here only the case of $2N$, $N \ge 1$, elements in the array. For an odd number of elements, the arguments are essentially unchanged. Counting from the center of the array, the position x_i of the ith element is

$$x_i = \left(\frac{2i-1}{2}\right)\frac{\lambda}{2}, \qquad i = 1,\cdots,N$$

where λ is the wavelength of the design frequency. Then

$$\xi_i = \frac{2x_i}{\lambda} = \frac{(2i-1)}{2}$$

so that from (1), the field pattern is proportional to be absolute value of

$$P(u) = \sum_{i=1}^{N} \alpha_i \cos\left(\frac{2i-1}{2}u\right)$$

where α_i is the element current of the ith element from the center of the array. The Dolph–Chebyshev coefficients are determined for each specified beamwidth greater than zero. To apply the theorem of Section III, it is necessary to show that the N functions

$$\cos\left(\frac{u}{2}\right), \cos\left(\frac{3}{2}u\right),\cdots, \cos\left(\frac{2N-1}{2}u\right) \tag{4}$$

and the $N-1$ functions

$$\cos\left(\frac{u}{2}\right), \cos\left(\frac{3}{2}u\right),\cdots, \cos\left(\frac{2N-3}{2}u\right) \tag{5}$$

form Chebyshev bases on the intervals $[0,\pi)$ and $[u_0,\pi)$, where $u_0 = 0$ here. Consider, then, any linear combination of the functions (4)

$$f(u) = \sum_{i=1}^{N} \beta_i \cos\left(\frac{2i-1}{2}u\right), \qquad 0 \le u < \pi$$

$$= \sum_{i=1}^{N} \beta_i T_{2i-1}\left(\cos\frac{u}{2}\right)$$

TABLE I
OPTIMUM ELEMENT CURRENTS FOR FIELD PATTERNS GIVEN IN FIG. 1 WITH A COMPARISON TO OPTIMIZED EQUISPACED ARRAYS

u_0	.40138	.50138	.66138	.78138	.88138
Element Currents α_1	.66524	1.40005	2.60994	4.23857	5.99725
α_2	.56748	1.14597	2.03999	3.16892	4.32041
α_3	.41913	.79211	1.30817	1.88695	2.41420
α_4	.23500	.39087	.54719	.66452	.73742
α_5	1.00000	1.00000	1.00000	1.00000	1.00000
Side-Lobe Level (dB)	-9.92	-14.91	-20.11	-25.12	-29.50
Beamwidth (deg)	9.72	11.68	13.73	15.74	17.50
Beamwidth (deg) for 10-element, half-wavelength, equispaced array with Dolph-Chebyshev for the same side-lobe levels.	9.69	11.57	13.62	15.65	17.49

where T_{2i-1} is the Chebyshev polynomial of degree $2i - 1$. Put $u = 2\cos^{-1}(x)$. Then $0 < x \leq 1$ and

$$f(u) \equiv g(x) = \sum_{i=1}^{N} \beta_i T_{2i-1}(x)$$

so that $g(x)$ is a polynomial of degree at most $2N - 1$. Thus $g(x)$ can have at most $2N - 1$ zeros in any interval. Furthermore, $g(x)$ is an odd function and so can have at most $N - 1$ zeros in the interval $(0,1]$, so that $f(u)$ can have at most $N - 1$ zeros in $[0,\pi)$. Hence the functions (4) form a Chebyshev basis on the interval $[0,\pi)$. Replacing N by $N - 1$ in this argument shows that the functions (5) also form the required basis.

By the theorem of Section III, for each specified beamwidth, $\theta \geq 0$, there exists a unique optimum field pattern and a unique set of element currents. These currents are the Dolph–Chebyshev coefficients for the beamwidth θ. It should be pointed out that, if $0 \leq \theta < \pi/(4N - 2)$, the sidelobes have larger amplitude than the main lobe. The next section shows how the Remes exchange algorithm may be used as an alternative means of calculating the Dolph–Chebyshev coefficients, although the usual methods of calculation of these coefficients are preferable to this method.

V. EXAMPLE

The example chosen is a ten-element linear array with the elements located at positions proportional to the abscissas of a ten-point Gaussian quadrature formula:

$$\xi_1 = 0.68788$$
$$\xi_2 = 2.00253$$
$$\xi_3 = 3.13926$$
$$\xi_4 = 3.99708$$
$$\xi_5 = 4.50000.$$

The length of this array is the same as that of a ten-element half-wavelength equispaced array, but the element positions are substantially displaced from equal spacing. An effort to verify that the functions (2) and (3) form Chebyshev bases in this case was unsuccessful, and direct numerical verification was not attempted. Instead, the Remes exchange algorithm was employed immediately to find the element currents, and the observed behavior of the algorithm itself was used to make inferences about the functions (2) and (3). In this case, it will be seen that the functions (2) are not, in fact, a Chebyshev basis, and that it is likely that the functions (3) are not either. The example presented here shows the utility of the approach even when the hypotheses of the theorem of Section III do not apply.

Ma [4] describes what is essentially the Remes exchange algorithm and applies it to the synthesis of nonuniformly spaced arrays. However, Ma seeks approximations to the function $f(u) = \exp(-Au^2)$, where A is a positive real number, so that the element currents obtained are only approximately optimum. To find the optimum element currents, proceed as in the proof of the theorem in Section III to find a minimax approximation to $-\cos(\xi_1 u)$ in the form

$$-\cos(\xi_1 u) \doteq \sum_{k=1}^{4} \alpha_k \cos(\xi_k u) \tag{6}$$

on some interval $[u_0,\pi)$, $u_0 > 0$. The error curve of this approximation over the full interval $[0,\pi)$ is identically the optimum beam pattern for the beamwidth determined by the first null. The parameter u_0 alone controls the tradeoff between the sidelobe level and the beamwidth. Therefore, u_0 is varied systematically here.

To begin, the Dolph–Chebyshev coefficients corresponding to a sidelobe level of $R = -10$ dB were used as the initial guess for α_1, α_2, α_3, and α_4, so that the choice

$$u_0 = 2\cos^{-1}\left(\frac{1}{z_0}\right) \doteq 0.40138$$

was made, where

$$z_0 = \frac{1}{2}\left\{\left[r + (r^2 - 1)^{1/2}\right]^{1/L} + \left[r - (r^2 - 1)^{1/2}\right]^{1/L}\right\}$$

with $L = M - 1 = 9$ and $r = 10^{R/20}$. With this initial guess on this interval, the Remes exchange algorithm computed the minimax approximation (6) in two iterations and produced a result shown in Table I. To continue the procedure, u_0 was incremented by 0.01, and the Remes algorithm employed again using these newly computed coefficients as the initial guess on this smaller interval. Convergence occurred in two iterations, the beamwidth increased slightly, and the sidelobe level reduced to —10.3 dB. Continuing in this fashion, u_0 was systematically increased from 0.40138 to 1.02138. Representative beam patterns appear in Fig 1 and the corresponding element currents in Table I. Notice that the beamwidths attainable by the Dolph–Chebyshev current amplitudes for an equispaced array are remarkably close to those obtained in this example.

By inspecting the beam patterns with the three lowest sidelobe levels, it is seen that each of these beam patterns possesses 5 zeros.

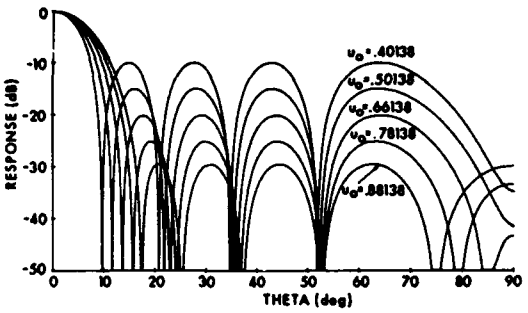


Fig. 1. Optimum field patterns for ten-element symmetrically positioned and unequally spaced linear array.

Since each of these beam patterns is also identically the error curve of a minimax approximation (6) on an interval $[u_0,\pi)$, it must be concluded that the functions (2) do not form a Chebyshev basis on the interval $[0,\pi)$. Consequently, the element currents may not be unique.

For each iteration of the Remes exchange algorithm, the solution of a system of linear equations is required. If there are $N-1$ Chebyshev basis elements, then one equation in N unknowns is established for each point where equiripple error should occur. By the theorem in Section II, there must be at least $(N-1)+1=N$ points of equiripple error. In the five results given for the present example, there are exactly $N=5$ points of equiripple error, counting one point on the main lobe down at the sidelobe level (at $u=u_0$), so that unexpected numerical difficulties do not occur. To proceed further than these results requires the solution of six equations in five unknowns, because of the growth of the extra lobe at $\theta=90°$. The straightforward procedure of solving any five of these six equations proved unsatisfactory because erratic behavior developed in the sidelobe corresponding to the equation deleted. An attempt to solve all six equations in both the least squares sense and the least maximum error sense by employing the generalized inverse of the coefficient matrix also proved unsatisfactory. It would seem, then, that either numerical difficulties are the cause of the problem or that the functions (3) do not form a Chebyshev basis. The author favors the latter possibility.

It should be noted that the Dolph–Chebyshev element currents for both a 10-element and a 50-element half-wavelength equispaced array have been computed in the aforementioned manner without difficulty from -10 dB to over -70 dB. (In these cases, an extra side lobe at 90° never develops.) Unequally spaced arrays with as many as 50 elements have also been successfully treated by this method.

All calculations were performed in double precision on the Univac 1108. Total CPU time, including plot generation for the example given was 67 s, although a more carefully written program could have reduced this time by at least a factor of two. A total of 63 sets of current amplitudes were computed.

VI. SUMMARY

Sufficient conditions for the existence of optimum field patterns for symmetrically spaced and amplitude tapered linear arrays have been proved. The theorem proved is a generalization of the work of Dolph on half-wavelength spaced linear arrays. A well-known algorithm from approximation theory has been employed in an example to compute the element currents corresponding to the optimum beam patterns using only the given element spacings themselves.

REFERENCES

[1] C. L. Dolph, "A current distribution of broadside arrays which optimizes the relationship between beam width and side-lobe level," *Proc. IRE Waves Electrons*, vol. 34, pp. 335–348, June 1946.

[2] S. Karlin and W. J. Studden, *Tchebysheff Systems: With Applications in Analysis and Statistics*. New York: Wiley, 1966.

[3] T. J. Rivlin, *An Introduction to the Approximation of Functions*. Waltham, Mass.: Blaisdell, 1969.

[4] M. T. Ma, "Another method of synthesizing nonuniformly spaced arrays," *IEEE Trans. Antennas Propagat.* (Commun.), vol. AP-13, pp. 833–834, Sept. 1965.

Optimized Symmetric Discrete Line Arrays

R. L. Streit

Optimized Symmetric Discrete Line Arrays

ROY L. STREIT

Abstract—**A generalization of Dolph's method for the synthesis of discrete antenna arrays is applied to six different symmetric line arrays. Based on these examples, it is concluded that 1) the field patterns of optimized symmetric line arrays with the same number of elements and with the same aperture are virtually indistinguishable and 2) optimized arrays with an odd number of elements are substantially better, in general, than arrays with an even number of elements.**

I. Introduction

If all the elements in a linear array are equally spaced at a half wavelength, then Dolph's method [1] may be used to compute the element currents of optimum field patterns for any specified beamwidth and for any number of array elements. Optimized equispaced arrays have two striking characteristics: 1) as the specified beamwidth is increased, the corresponding sidelobe level diminishes and 2) all the sidelobes are of equal amplitude. In the generalization of Dolph's method, both 1) the tradeoff between the beamwidth and the sidelobe level and 2) the equal amplitude sidelobe structure are extended to a larger class of symmetric arrays.

By application of the generalized Dolph method to six specific arrays, it was noticed that the method was more successful for arrays with an odd number of elements than arrays with an even number of elements. All symmetric arrays with the same odd number of elements and the same aperture as a half-wave equispaced array seem to possess optimum field patterns with equal amplitude sidelobes for any specified beamwidth. Only very special even numbered symmetric arrays appear to share this feature; that is, for nearly all even arrays, element currents which suppress uniformly all sidelobes do not appear to exist for every desired beamwidth. Equispaced even numbered arrays seem to be the primary exception to this statement.

A second observation based on these examples is this: for fixed aperture, odd number of elements, and desired beamwidth, any symmetric set of positions is nearly as good as any other symmetric set provided the element currents are optimized for the given positions. To find unequally spaced arrays with field patterns substantially better than optimized equispaced arrays thus requires either a larger aperture or more elements. If an even number of elements is specified, these observations are false: equal spacing is definitely better, in general, because of the remarks in the preceding paragraph.

The directional response of a symmetric line array of M elements is directly proportional to the absolute value of a linear combination of cosines:

$$P(u) = \sum_{k=1}^{N} a_k \cos(\xi_k u), \qquad 0 \le u < \pi$$

where

$$N = \left[\frac{M+1}{2}\right]$$

$$\xi_k = \frac{x_k}{(\lambda/2)}$$

$$u = \pi \sin\theta$$

Manuscript received March 2, 1975; revised June 12, 1975.
The author is with the Naval Underwater Systems Center, New London Laboratory, New London, Conn. 06320.

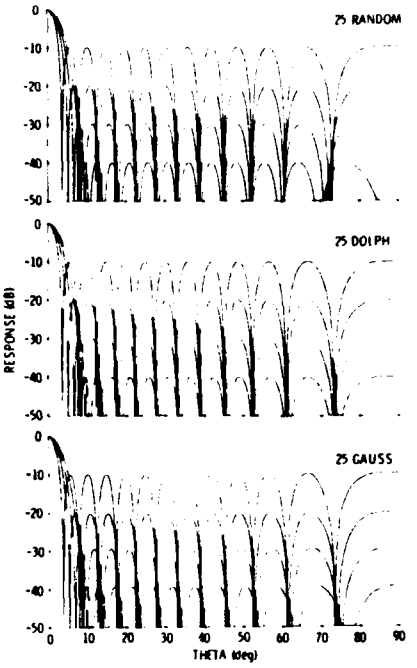


Fig. 1. Field patterns for element currents in Tables I and II

and θ is measured from a normal to the line of the array, λ is the wavelength of the design frequency, $\{x_k\}$ are element positions measured from the array center, and $\{a_k\}$ are element currents. Thus, if M is odd, the center element must lie at the origin and the center element current is half of a_1. In Fig. 1 the field patterns shown are $20 \log_{10} |P(u)|$, normalized by its maximum absolute value on $[0,\pi)$, but plotted versus the angle θ and not u.

The theorem in [2] is valid for both even and odd numbers of elements. For the odd case, however, the result can be strengthened. In the following theorem, the beamwidth is measured from $u = 0$ down to the sidelobe level. Also, by definition, a collection of continuous functions form a Chebyshev basis on a finite interval if the zeros of any nontrivial linear combination of these functions number at most one less than the number of functions in the basis.

Theorem (Odd Case): If M is odd, then the function

$$P(u) \equiv \cos(\xi_N u) + \sum_{k=1}^{N-1} a_k \cos(\xi_k u), \qquad 0 \le u < \pi$$

is the unique optimum field pattern for a beamwidth of $\arcsin(u_0/\pi)$ provided

i)

$$\max_{u_0 \le u < \pi} |P(u)| = \min_{\{a_k\}} \max_{u_0 \le u < \pi} \left|\cos(\xi_N u) + \sum_{k=1}^{N-1} a_k \cos(\xi_k u)\right|$$

ii)

$\{\cos(\xi_1 u), \cdots, \cos(\xi_{N-1} u)\}$ is a Chebyshev basis on $[u_0, \pi)$, $u_0 > 0$

iii)

$\{\cos(\xi_1 u), \cdots, \cos(\xi_N u)\}$ is a Chebyshev basis on $[0,\pi)$.

This result is stronger than that given in [2] because the beamwidth is given as an explicit function of u_0. This is possible because Meinardus has shown ([3, theorem 30]) that assumptions ii) and iii) together imply that u_0 must be an extreme point of the approximation i). Therefore, u_0 is precisely that point on the main lobe which is down at the sidelobe level. (It can also be shown that the first zero of the minimax approximation i) is a strictly increasing function of u_0, so that the theorem is essentially unchanged if beamwidth is measured to the first null as in [1] instead of down to the sidelobe level. The only difference is that the beamwidth cannot be given as an explicit function of u_0. See [5].)

Optimum field patterns (in the odd case) do not necessarily possess exactly $N-1$ equal amplitude sidelobes nor are optimum field patterns necessarily unique unless the assumptions ii) and iii) are made. The assumption ii) is well known ([3, theorems 19 and 20]) to be both a necessary and sufficient condition for the existence and uniqueness of the minimax approximation i), so that without this assumption an optimum field pattern will not be unique if it satisfies i). Furthermore, Meinardus ([3, theorem 30]) has shown that the assumptions ii) and iii) together imply the existence of exactly N extreme points of the minimax approximation i). Since one of these extreme points must be on the main lobe itself, and every other extreme point corresponds to a sidelobe, without both these assumptions an optimum field pattern satisfying i) need not possess precisely $N-1$ sidelobes.

A nearly identical result holds for an even number of elements, but an explicit relation between beamwidth and the parameter u_0 is not available. Meinardus' result applies only to Chebyshev systems containing the constant 1 as a basis function, a circumstance which occurs for symmetric line arrays for odd M only. Thus, the strongest statement possible is that the beamwidth is perhaps (slightly) smaller than $\arcsin(u_0/\pi)$.

These results can be extended in two directions. First, the synthesis of steered arrays can be performed by computing the approximation i) on intervals extending as far beyond π as desired. Secondly, the fact that ii) and iii) are cosine bases is never used in the proof of the results. Ma [4, p. 215] gives an example of concentric continuous rings whose field pattern is a linear combination of basis functions of the form $J_0(\zeta_k u)$. Therefore, the theorem above also yields an optimality result for this case.

TABLE I
OPTIMUM ELEMENT CURRENTS FOR 25-ELEMENT RANDOM ARRAY

Element Position	u_0 = .151	u_0 = .249	u_0 = .341	u_0 = .434
.000	.031	.136	.427	1.165
.777	.256	1.041	3.140	8.313
1.930	.122	.500	1.503	3.964
2.590	.193	.775	2.277	5.895
3.867	.226	.889	2.555	6.440
4.849	.118	.444	1.223	2.966
5.695	.195	.721	1.939	4.567
6.871	.158	.549	1.380	3.035
7.531	.116	.393	.963	2.051
8.675	.206	.644	1.437	2.785
9.941	.146	.420	.844	1.457
10.780	.125	.329	.594	.911
12.000	1.000	1.000	1.000	1.000
Sidelobe level(dB)	-9.998	-20.062	-29.906	-39.950
Beamwidth (deg)	2.759	4.541	6.236	7.936

TABLE II
OPTIMUM ELEMENT CURRENTS FOR 25-ELEMENT GAUSS ARRAY

Element Position	u_0 = .151	u_0 = .251	u_0 = .341	u_0 = .441
.000	.263	1.453	5.858	-58.734
1.481	.259	1.423	5.693	-58.571
2.939	.249	1.337	5.231	-50.580
4.353	.234	1.208	4.550	-42.064
5.701	.213	1.049	3.755	-32.639
6.963	.189	.878	2.950	-23.741
8.119	.164	.709	2.213	-16.252
9.152	.137	.554	1.596	-10.636
10.046	.113	.415	1.092	-6.289
10.788	.078	.306	.769	-4.871
11.366	.099	.193	.364	.359
11.772	-.106	.132	.474	-6.678
12.000	1.000	1.000	1.000	1.000
Sidelobe level(dB)	-9.994	-20.189	-29.507	-39.371
Beamwidth (deg)	2.759	4.587	6.236	8.074

II. Application to Arrays With Odd Number of Elements

Three 25-element arrays called herein (for convenience) Random, Dolph, and Gauss are considered. The Dolph array is equispaced. In the Random array, all the element positions except the first and the last are displaced strictly toward the origin from equal spacings by no more than 0.5, but otherwise in a random fashion. The Gauss array has elements located at positions proportional to a 25-point Gaussian quadrature formula. Thus the Gauss array is substantially more perturbed from equal spacing than is the Random array. All three arrays have the same aperture.

The Dolph positions satisfy the theorem statement (see [2]). However, for reasons that will be stated later, the Random array does not satisfy condition iii), whereas the Gauss array does not satisfy condition ii). Therefore, computational difficulties can be expected for the Gauss array. Also, the field patterns presented for the Gauss array may not be optimum.

The application of the method to the Random and Gauss arrays is detailed in Tables I and II, respectively, and follows the development in [2]. Since in each case the proper choice of the subintervals $[u_0,\pi)$ could not be made beforehand, an arbitrary starting point of $u_0 = 0.1$ was picked and the Remes exchange algorithm employed to compute the minimax approximation i) on the subinterval $[0.1,\pi)$. Then u_0 was incremented by 0.01 and the Remes exchange algorithm was used again on the slightly smaller interval $[0.11,\pi)$. Continuing in this fashion gave a family of optimum element currents. Four sets of element currents for both arrays are given in Tables I and II. Notice that the beamwidth-sidelobe level tradeoff is as expected in both cases. Also, the element currents appear to be continuous functions of the sidelobe parameter u_0 except in the Gauss array. For the Gauss array, nonuniqueness is the consequence of violating the second condition of the theorem.

The accuracy of the results can be estimated by comparing similar results for the Dolph array with those that can be obtained explicitly. It was found that the currents were correct to 4 or 5 significant decimals. More accuracy was not attained because a discrete version, and not a continuous version, of the Remes exchange algorithm was implemented in the computer program.

The field patterns for the element currents in Tables I and II are shown in Fig. 1. Even though the element currents and positions are quite dissimiliar, all three sets of field patterns are

nearly identical and there is very little difference in the beamwidths to be had for the same sidelobe level. However, it can be seen that the Random array is slightly superior to the Dolph array. To the author's knowledge, this is the first explicit example of an optimum unequally spaced array which can be shown to be superior to an optimum equispaced array of exactly the same length and the same number of elements. Practically speaking, however, the field patterns are virtually identical in all three cases.

The Random array does not satisfy condition iii) because the field pattern for -40 dB has 13 zeros. Since the field pattern is a linear combination of the 13 functions in iii), the Chebyshev condition fails. That the Gauss array fails to satisfy condition ii) is not as straightforward. It can be shown that the smaller the interval of approximation, the less the error of the minimax approximation on that interval. The four given approximations (Table II) satisfy the requirements of Chebyshev's theorem ([3, theorem 23]) and therefore should be minimax approximations; however, the error of these approximations does not decrease with increasing u_0. Hence, the Gauss array cannot satisfy condition ii).

III. Application to Arrays With Even Number of Elements

The chief difference, numerically, between even numbered arrays and odd numbered arrays is that the field pattern function $P(u)$ does not contain the constant 1 as a basis function because no element lies at the array center. Meinardus' result does not apply, and the problem that develops in the even case is that one too many sidelobes develops so that not all of them can be suppressed.

The Remes exchange algorithm establishes one linear algebraic equation for each sidelobe and one equation corresponding to a point on the mainlobe down at the sidelobe level. Because of Chebyshev's theorem, there must be *at least* N equations in exactly N unknowns when the minimax approximation i) has been found. For arrays with an odd number of elements, there is never any difficulty because there are always *exactly* N equations in N unknowns ([3, theorem 30]). The difficulty with even arrays, then, is that eventually it develops that $N + 1$ equations in N unknowns must be solved exactly, and these equations prove to be inconsistent.

Three arrays which are the 24-element analogs of the same named 25-element arrays were considered. The Dolph array satisfied the theorem statement, the Random array did not satisfy condition iii), and the Gauss array did not satisfy either condition ii) or iii). The application of the Remes exchange algorithm to these three arrays proceeded exactly as in Section III, and field patterns with sidelobe levels of -10 dB, -15 dB, and -20 dB were obtained. Too many sidelobes never develop for the Dolph array because of the fact that $P(u)$ must be identically zero at $u = \pi$ (i.e., $\theta = 90°$). The other two arrays did not have this feature, so that inevitably a sidelobe appeared at $u = \pi$. For small u_0, this sidelobe either did not exist or it was small, but for larger u_0, the sidelobe at $u = \pi$ appeared and increased in size until it equaled in magnitude the other sidelobes. At this point, the difficulty of the overdetermined system of equations developed. In consequence, the minimax approximation i) does not change until u_0 has increased to the point that the first sidelobe is included as part of the main lobe. The extra equation can then be dropped and further reduction of the remaining sidelobes is then possible. Dropping the first equation is equivalent to losing control of the first sidelobe. The remaining sidelobes may well be reduced in amplitude, but only at the expense of a bad first lobe. Field patterns with *all* the sidelobes at, say, -30 dB were not computed because no such field patterns existed for the Random and Gauss arrays. Since this procedure is unavoidable, the conclusion must be that the equispaced Dolph array is much superior to either of the other two arrays. For as long as the extra sidelobe was not a problem, the three arrays were indistinguishable from the beamwidth sidelobe level tradeoff standpoint.

The development of the extra sidelobe seems inevitable in the general even case. Since the extra sidelobe prevents a uniform suppression of all sidelobes, it follows that optimized arrays with an odd number of elements have a substantially better character than optimized arrays with an even number of elements.

IV. Summary

A generalization of Dolph's method is applied to six different arrays. Based on these examples, it is concluded that optimized equispaced arrays are as good as any other optimized symmetric line array with the same number of elements and aperture. Another conclusion is that arrays with an odd number of elements have better behavior than arrays with an even number of elements.

References

[1] C. L. Dolph, "A current distribution of broadside arrays which optimizes the relationship between beamwidth and sidelobe level," *Proc. IRE, Waves Electron.*, vol. 34, pp. 335–348, June 1946.
[2] R. L. Streit, "Sufficient conditions for the existence of optimum beam patterns for unequally spaced linear arrays with an example," *IEEE Trans. Antennas Propagat.*, vol. AP-23, pp. 112–115, Jan. 1975.
[3] G. Meinardus, *Approximation of Functions: Theory and Numerical Methods*. New York: Springer-Verlag, 1967.
[4] M. T. Ma, *Theory and Application of Antenna Arrays*. New York: Wiley-Interscience, 1974.
[5] R. L. Streit, "Extremals and zeros in Markov systems are monotone functions of one end point," in *Proc. Conf. on the Theory of Approximation*, Calgary, Alta., Canada, 1975, and in *Theory of Approximation*. New York: Academic, to appear.

Real Excitation Coefficients Suffice For Sidelobe Control In Linear Array

J. T. Lewis and R. L. Streit

Real Excitation Coefficients Suffice for Sidelobe Control in a Linear Array

JAMES T. LEWIS AND ROY L. STREIT

Abstract—**Minimax design of a linear antenna array with arbitrary fixed elements leads to the following mathematical problem.**

$$\underset{w_k \text{ complex}}{\text{minimize}} \quad \max_{u_0 \le |u| \le u_1} |T(u)|$$

subject to $T(0) = 1$ where $T(u) = \sum_{k=1}^{N} w_k \exp(-id_k u)$ and d_k are real numbers. It is proven that this problem has a solution with *real* excitation coefficients w_k. In the antenna application this shows that there is no need to allow phasing at the individual elements of the array; amplitude control alone will achieve all the sidelobe reduction possible. An analogous result can be proved for a more general complex approximation problem.

We consider a linear antenna array with N omnidirectional elements located at arbitrary fixed positions $\{x_k\}$ receiving a plane wave of wavelength λ from the direction θ_a, $-\pi/2 \le \theta_a \le \pi/2$, relative to a normal to the array. If the array is steered to look in the direction θ_l, $-\pi/2 \le \theta_l \le \pi/2$, then the complex transfer function of the beamformer is given by

$$T(u) = \sum_{k=1}^{N} w_k \exp(-id_k u)$$

where $\{w_k\}$ are the element excitation coefficients, $d_k = 2\pi x_k/\lambda$, and $u = \sin\theta_a - \sin\theta_l$. The coefficients w_k may be complex in general. The peak response should occur at $u = 0$; we make the usual normalization

$$T(0) = \sum_{k=1}^{N} w_k = 1.$$

To effect small sidelobes we wish to minimize $|T(u)|$ for $|u| \ge u_0$ where $u_0 > 0$ is chosen small.

The total range of u depends on the look direction θ_l. First, let us consider only the case of the array steered broadside. Thus $\theta_l = 0$ and the range of u becomes $-1 \le u \le 1$ corresponding to $-1 \le \sin\theta_a \le 1$ for $-\pi/2 \le \theta_a \le \pi/2$. Hence the problem of selecting excitation coefficients to effect minimum overall sidelobe level becomes a minimax problem.

$$\underset{w_k \text{ complex}}{\text{minimize}} \quad \max_{u_0 \le |u| \le 1} |T(u)|$$

subject to

$$T(0) = \sum_{k=1}^{N} w_k = 1. \tag{1}$$

The case where the array is steered through the same number of degrees either side of broadside is very similar mathematically to the case $\theta_l = 0$ and is discussed below.

Manuscript received October 30, 1981; revised January 4, 1982.

J. T. Lewis was with the Naval Underwater Systems Center, New London Laboratory, New London, CT 06320 during the summer of 1981, on leave from the Department of Mathematics, University of Rhode Island, Kingston, RI 02881.

R. L. Streit is with the Naval Underwater Systems Center, New London Laboratory, New London, CT 06320.

A standard argument shows that a solution to problem (1) exists; however, it may not be unique. In general the excitation coefficients w_k are allowed to be complex; we now prove that a solution of (1) exists with w_k all *real*. First, denoting complex conjugates by an overbar,

$$\begin{aligned}
&\max_{u_0 \le |u| \le 1} \left| \sum_{k=1}^{N} \bar{w}_k e^{-id_k u} \right| \\
&= \max_{u_0 \le |u| \le 1} \overline{\left| \sum_{k=1}^{N} \bar{w}_k e^{-id_k u} \right|} \\
&= \max_{u_0 \le |u| \le 1} \left| \sum_{k=1}^{N} w_k e^{id_k u} \right| \\
&= \max_{u_0 \le |u| \le 1} \left| \sum_{k=1}^{N} w_k e^{-id_k u} \right|.
\end{aligned}$$

The last equality follows from the fact that $u_0 \le |-u| \le 1$ if and only if $u_0 \le |u| \le 1$; i.e., the range of u is symmetric about $u = 0$. Now

$$\begin{aligned}
&\max_{u_0 \le |u| \le 1} \left| \sum_{k=1}^{N} (\operatorname{Re} w_k) e^{-id_k u} \right| \\
&= \max_{u_0 \le |u| \le 1} \left| \sum_{k=1}^{N} \frac{1}{2} (w_k + \bar{w}_k) e^{-id_k u} \right| \\
&\le \tfrac{1}{2} \max_{u_0 \le |u| \le 1} \left| \sum_{k=1}^{N} w_k e^{-id_k u} \right| \\
&\quad + \tfrac{1}{2} \max_{u_0 \le |u| \le 1} \left| \sum_{k=1}^{N} \bar{w}_k e^{-id_k u} \right| \\
&= \max_{u_0 \le |u| \le 1} \left| \sum_{k=1}^{N} w_k e^{-id_k u} \right| \text{ from above.}
\end{aligned}$$

This guarantees the existence of a real solution of problem (1) as asserted, since

$$\sum_{k=1}^{N} w_k = 1 \text{ implies } \sum_{k=1}^{N} (\operatorname{Re} w_k) = \operatorname{Re} \sum_{k=1}^{N} w_k = 1.$$

We now note that, since $|T(-u)| = |\overline{T(u)}| = |T(u)|$ when $w_1, \cdots, w_N$ are real, we can further simplify problem (1) to

$$\underset{w_k \text{ real}}{\text{minimize}} \quad \max_{u_0 \le |u| \le 1} |T(u)| \tag{1a}$$

subject to

$$T(0) = \sum_{k=1}^{N} w_k = 1.$$

Hence, we can find a solution to problem (1) by solving the easier problem (1a). This has important practical implications for the design of an antenna array. It indicates that there is no need to allow phasing at the individual elements; amplitude of excitation alone will achieve all the sidelobe reduction that is possible.

The above analysis was for the look angle $\theta_l = 0$. Now let us regard θ_l as not being fixed; then the range of u becomes $-2 \le u \le 2$. The problem corresponding to (1) with θ_l bounded

away from endfire is

minimize

$$w_k \quad \max_{u_0 \le |u| \le 2-u_1} |T(u)|$$

subject to

$$T(0) = \sum_{k=1}^{N} w_k' = 1. \tag{2}$$

As above, we can show the existence of a solution of (2) with *real* excitation coefficients w_k.

Now, let us consider a more general complex approximation problem. Let $f, h_1, \cdots, h_N$ be continuous complex valued functions defined on a closed and bounded set Q in the complex plane. (Q can be finite or infinite.) The minimax approximation problem is

minimize

$$a_k \quad \max_{z \in Q} \left| f(z) - \sum_{k=1}^{N} a_k h_k(z) \right|. \tag{3}$$

Here the a_k are allowed to be complex. If for all $z \in Q$, $f(\bar{z}) = \overline{f(z)}$, $h_k(\bar{z}) = \overline{h_k(z)}$, $k = 1, \cdots, N$ and Q is symmetric with respect to the real axis, i.e., $\bar{q}$ in Q if and only if q in Q, then a solution of (3) with real coefficients exists. We omit the details of the verification.

Finally, we note that real excitation coefficients are not adequate for every use of a linear array or for every pattern desired. For example, if a null is required in the pattern at a point $\hat{u} \neq 0$, the equation $T(\hat{u}) = 0$ would be added to problem (1). However, now a solution with real coefficients would not necessarily exist.

Two Exponential Approximation Methods

R. L. Streit

Abstract

Two different constructive techniques for approximating positive definite functions by means of finite exponential sums are explored. One technique constructs the coefficients and the exponents. The other technique constructs the exponents when the coefficients are all required to be equal. Both approximation techniques appear to be suitable for numerical computation. The techniques extend to completely monotonic functions as well. Error bounds are proved using elementary methods.

In an application, these error bounds can be used to eliminate some of the effort and guesswork previously necessary in two procedures for the design and synthesis of sparse broadband linear arrays.

Two Exponential Approximation Methods

I. Introduction

Two design procedures for aperiodic, or space tapered, linear arrays are investigated in this report in a setting much more general than the usual setting. One procedure, due to Bruce and Unz [1], gives both element excitations ("shadings") and positions. The other procedure, due to Maffett [2], gives element positions under the condition that all excitations are unity. Both seek desirable radiation patterns minimizing grating lobes. These methods synthesize sparse broadband arrays that are less sensitive to frequency changes than periodic (equispaced) arrays.

Using either of these procedures, the designer must guess the number of elements required, perform the appropriate numerical computations, examine the resulting radiation pattern, and then decide if more elements are required or if fewer elements will suffice. In this report, error bounds are derived that provide estimates on the number of elements necessary for a given degree of approximation of the desired radiation pattern. Thus, some of the effort and guesswork inherent in these procedures can be eliminated.

Neither of these two methods is intrinsically limited to aperiodic array design. Generalizations turn out to be worthwhile and of independent interest. Therefore, this report addresses only the general setting from this point on.

A complex valued function f of a real variable is defined to be positive definite if and only if, for each integer $n \geq 1$, the inequality

$$\sum_{i,j=1}^{n} f(x_i - x_j)\, a_i \bar{a}_j \geq 0 \tag{1.1}$$

holds for all $x_1, \ldots, x_n \in \mathbf{R}$ (the real numbers) and $a_1, \ldots, a_n \in \mathbf{C}$ (the complex numbers). Bochner's Theorem states the following: If f is a continuous function on $\mathbf{R}$, then f is positive definite if, and only if, there exists a bounded non-decreasing function V on $\mathbf{R}$ such that f is the Fourier-Stieltjes transform of V; that is,

$$f(x) = \int_{-\infty}^{\infty} e^{i\alpha x}\, dV(\alpha), \qquad x \in \mathbf{R}\,. \tag{1.2}$$

The recent paper of Stewart [3] gives references to various proofs of Bochner's Theorem and its generalizations. We point out, for future use, that (1.2) immediately implies that the total variation of V equals f(0), and for all real x, f(-x) equals the complex conjugate of f(x). Goldberg [4] proves that any positive definite function f is such that $|f(x)| \leq f(0)$ for all real x.

In this report, we restrict our attention, for the most part, to continuous positive definite functions f on $\mathbf{R}$ that can be written

$$f(x) = \int_{-\lambda}^{\lambda} e^{i\alpha x}\, dV(\alpha), \qquad x \in \mathbf{R}, \tag{1.3}$$

for some real number λ such that $0 < \lambda < \infty$. In other words, we have assumed that $V(\alpha)$ is constant for $|\alpha| > \lambda$. For functions satisfying (1.3), we develop in an elementary manner an approximation to f(x) of the form

$$S_n(x) = f(0) \sum_{k=1}^{n} a_k e^{i\alpha_k x}, \tag{1.4}$$

where $|\alpha_k| < \lambda$, $k = 1, \ldots, n$. We give an error bound for this approximation in Theorem 1. This approximation always gives positive coefficients and exponents $\alpha_k \varepsilon \mathbf{R}$ that are located at the roots of an appropriate orthogonal polynomial. We suspect that these approximations are near-optimal in some well-defined sense. (See Schabach [5, p. 1018] for a relevant conjecture about a particular function f.)

Under various additional assumptions concerning V, we develop an approximation to f(x) of the form

$$Q_n(x) = \frac{f(0)}{n} \sum_{k=1}^{n} e^{i\alpha_k x}, \tag{1.5}$$

where $|\alpha_k| < \lambda$, $k = 1, \ldots n$, and we give an error bound in Theorem 2. The approximation $Q_n(x)$ cannot be as efficient in general as the approximation $S_n(x)$; however, $Q_n(x)$ has the advantage of being much more easily constructed in practice for almost any reasonable n (say, $n < 10^6$).

Note that both the approximations $S_n(x)$ and $Q_n(x)$ are readily written in the form (1.3) and, therefore, are positive definite. Hence, we must have

$$|f(x)-S_n(x)| \leq |f(x)| + |S_n(x)| \leq 2f(0), \qquad x \varepsilon \mathbf{R}, \tag{1.6}$$

since $S_n(0) = f(0)$. Similarly, it is always the case that $|f(x) - Q_n(x)| \leq 2 f(0)$.

It will be shown that Prony's method can be used to compute $S_n(x)$. Although Prony's method in this problem must become numerically ill-conditioned for n sufficiently large, it may nonetheless be useful for small n (say, $n \leq 10$). Numerically stable methods for computing $S_n(x)$ suitable for all n would require an algorithm other than Prony's method. This is discussed at the end of Section II.

The computation of approximations $Q_n(x)$ is shown to depend upon the ability to compute the numerical value of the inverse function of V (guaranteed to exist by additional assumptions) at specific points. The level of difficulty involved depends on V, of course, but the interval is finite, so the problem seems to encounter no inherent numerical difficulties.

An excellent bibliography of references to the literature on exponential approximation is contained in [6].

Note that if V in (1.3) is continuously differentiable on the interval $(-\lambda, \lambda)$, then $V'(\alpha) \geq 0$, and

$$f(x) = \int_{-\lambda}^{\lambda} e^{i\alpha x} V'(\alpha)\, d\alpha. \tag{1.7}$$

From the Paley-Wiener Theorem (see, e.g., [7, p. 134]), this equation uniquely extends the domain of f to all C and that this extension of f is an entire function of exponential type at most λ.

We close this section with a small collection of positive definite functions. According to [3], Schoenberg proved that $f_r(x) = \exp[-|x|^r]$ is positive definite if and only if $0 \leq r \leq 2$, and Polya proved that any real, even, continuous function f that is convex on the interval $(0, \infty)$, that is, $f((x+y)/2) \leq (f(x)+f(y))/2$, and satisfies $\lim_{x \to \infty} f(x) = 0$, is positive definite. Goldberg [4, p. 61] proves that if f(x) is positive definite and $a > 0$, then the function $h(x) = f(x) \exp(-ax^2)$ is also positive definite. Finally, if the function f has a Fourier transform that is nonnegative and integrable, then the function f is positive definite. Specific examples of functions satisfying this latter property are

$$\frac{\sin \lambda x}{x} = \frac{1}{2}\int_{-\lambda}^{\lambda} e^{i\alpha x}\, d\alpha , \tag{1.8}$$

$$\frac{2(1-\cos \lambda x)}{\lambda x^2} = \int_{-\lambda}^{\lambda} e^{i\alpha x}\left(1 - \frac{|\alpha|}{\lambda}\right) d\alpha \tag{1.9}$$

$$e^{-|x|} = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{e^{i\alpha x}}{1+\alpha^2}\, d\alpha \tag{1.10}$$

$$e^{-ax^2} = \frac{1}{\sqrt{4a\pi}}\int_{-\infty}^{\infty} e^{i\alpha x} e^{-\alpha^2/4a}\, d\alpha \qquad (a > 0) \tag{1.11}$$

$$\frac{J_\nu(\lambda x)}{x^\nu} = \frac{(2\lambda)^{-\nu}}{\pi^{1/2}\Gamma(\nu+1/2)}\int_{-\lambda}^{\lambda} (\lambda^2-\alpha^2)^{\nu-1/2} e^{i\alpha x}\, d\alpha \quad (\nu > -1/2) \tag{1.12}$$

where $J_\nu(x)$ is the usual Bessel function of order ν. A final example, one that finds application in antenna design ([8], [9], and [10]), is

$$\cos\left(\sqrt{(\lambda x)^2 - a^2}\right) = \int_{-\lambda}^{\lambda} e^{i\alpha x}\, dV(\alpha) , \qquad a \geq 0, \tag{1.13}$$

where $V(-\lambda) = -1/2\lambda$,

$$V(\alpha) = \frac{a}{2}\int_{-\lambda}^{\alpha} \frac{I_1\left(\frac{a}{\lambda}\sqrt{\lambda^2-t^2}\right)}{(\lambda^2-t^2)^{1/2}}\, dt, \qquad -\lambda < \alpha < \lambda ,$$

$V(+\lambda) = 1/2\lambda + \lim_{\alpha \to \lambda^-} V(\alpha)$, and where $I_1(x)$ is the modified Bessel function of order one. This function is interesting because, for $|x| \geq a/\lambda$, it has magnitude not exceeding 1, while for $|x| < a/\lambda$, it exhibits very rapid growth achieving a maximum magnitude of cosh(a) at x = 0. Other examples can be discerned in various tables of integral transforms, such as [11].

3/4
Reverse Blank

II. Exponential Approximation With Arbitrary Coefficients

The idea developed in this section for constructing approximations of the form $S_n(x)$ is simply Gaussian quadrature. A glance at equation (1.3) reveals that we are particularly interested in Gaussian quadrature with respect to the measure $dV(\alpha)$. From Szegö [12, p. 25], a system of orthogonal polynomials exists for the measure $dV(\alpha)$ if $V(\alpha)$ has infinitely many points of increase in the interval $[-\lambda, -\lambda]$ and if the moments

$$c_m = \int_{-\lambda}^{+\lambda} \alpha^m \, dV(\alpha) \, , \; m = 0,1,2, \ldots \tag{2.1}$$

exist. Since V is bounded above, the moments c_m certainly exist. If V has finitely many points of increase, then f can be written explicitly as a finite sum of exponentials. Although this special case is not uninteresting (in the context of economizing large finite exponential sums), we will avoid it by assuming that V has infinitely many points of increase.

Let $\alpha_1, \ldots, \alpha_n$ be the abscissas and let $b_1, \ldots, b_n$ be the corresponding Cotes numbers of the n-th order Gaussian quadrature formula with respect to the measure $dV(\alpha)$. Since

$$\sum_{k=1}^{n} b_k = \int_{-\lambda}^{\lambda} 1 \, dV(\alpha) = f(0) \, ,$$

we rewrite the Cotes numbers in the form $b_k = a_k f(0)$, $k = 1, \ldots, n$. Using this notation, and applying the quadrature formula blindly to (1.3) gives the approximation

$$S_n(x) = f(0) \sum_{k=1}^{n} a_k e^{i\alpha_k x} \, , \tag{2.2}$$

where

$$0 < a_k < 1, \; k = 1, \ldots, n \tag{2.3}$$

$$a_1 + \ldots + a_n = 1 \tag{2.4}$$

$$|\alpha_k| < \lambda, \; k = 1, \ldots, n \, . \tag{2.5}$$

These three properties are immediate consequences of well known results on Gaussian quadrature. (See Szegö [12, pp. 47-49].) In addition, these properties imply that $S_n(x)$ possesses good numerical round-off error behavior when the sum is evaluated numerically.

We seek an error bound such that

$$|f(x) - S_n(x)| \leq R_n(x) \, , \qquad x \varepsilon \mathbf{R} \, . \tag{2.6}$$

It is clear from (1.3) and the Riemann-Lebesgue Lemma that $f(x) \to 0$ as $x \to \infty$. On the other hand, it is not hard to see from (2.2) that $S_n(x)$ cannot tend to zero as $x \to \infty$. The most that can be expected is that $R_n(x)$ becomes "small" for any fixed x. We will show that as $n \to \infty$, S_n converges to f uniformly on any finite real interval.

Let $n \geq 1$. For each $x \varepsilon \mathbf{R}$, let

$$\varepsilon_n(x) = \min \max_{-\lambda \leq \alpha \leq \lambda} | e^{i\alpha x} - \pi_{2n-1}(\alpha)| \, , \tag{2.7}$$

where the minimum is taken over all polynomials $\pi_{2n-1}(\alpha)$ of degree at most 2n-1 with complex coefficients. We always have $\varepsilon_n(x) \leq 1$ for all x, as can be seen by considering the case $\pi_{2n-1}(\alpha) \equiv 0$ in (2.7).

Lemma 1. For $n \geq 1$,

$$\varepsilon_n(x) \leq \sqrt{2} \, \frac{(\lambda x)^{2n}}{2^{2n-1}(2n)!} \, , \quad x \varepsilon \mathbf{R} \, . \tag{2.8}$$

Proof. From a theorem given in [13, p.78], for any real valued function $p(\alpha)$ defined on the interval $[-1, +1]$ and possessing $n+1$ continuous derivatives on $(-1, +1)$, we have

$$E_n(p) \equiv \min \max_{-1 \leq \alpha \leq 1} | p(\alpha) - \pi_n(\alpha) | = \frac{|p^{(n+1)}(\zeta)|}{2^n (n+1)!}$$

for some ζ, $-1 < \zeta < +1$, where the minimum is taken over all real polynomials π_n of degree at most n. For $p(\alpha) = \cos \alpha \lambda x$ defined for α in the interval $[-1, 1]$ and for fixed real numbers λ and x, we have

$$E_{2n-1}(p) = \frac{(\lambda x)^{2n}}{2^{2n-1}(2n)!} \, |\cos \zeta \lambda x|$$

$$\leq \frac{(\lambda x)^{2n}}{2^{2n-1}(2n)!} \, .$$

For $q(\alpha) = \sin \alpha \lambda x$ on $[-1, 1]$, we have similarly

$$E_{2n-1}(q) \leq \frac{(\lambda x)^{2n}}{2^{2n-1}(2n)!} \, .$$

From the definition of $\varepsilon_n(x)$, we have

$$\varepsilon_n(x) = \min \max_{-1 \leq \alpha \leq 1} |e^{i\alpha \lambda x} - \pi_{2n-1}(\lambda \alpha)|$$

$$\leq \{E^2_{2n-1}(p) + E^2_{2n-1}(q)\}^{1/2} \, .$$

Substituting the estimates for $E_{2n-1}(p)$ and $E_{2n-1}(q)$ completes the proof.

We remark, but do not prove, that an example in Meinardus [13, p. 96] can be extended and used to show that for fixed x,

$$\varepsilon_n(x) \leq \frac{(\lambda x)^{2n}}{2^{2n-1}(2n)!} \, (1 + o(1)) \, , n \to \infty \, .$$

It seems reasonable to conjecture that this asymptotic inequality is actually an asymptotic equality. In any event, we use only (2.8) in this report.

Theorem 1. Let f(x) be a continuous complex valued positive definite function of a real variable such that

$$f(x) = \int_{-\lambda}^{\lambda} e^{i\alpha x}\, dV(\alpha)\,, \qquad x\varepsilon\mathbf{R}, \tag{2.9}$$

where V is a bounded non-decreasing function having infinitely many points of increase in the finite closed interval $[-\lambda, \lambda]$. Then, for each integer $n \geqslant 1$, there exists distinct real numbers $\alpha_1, \ldots, \alpha_n$ and real numbers $a_1, \ldots, a_n$ satisfying (2.3), (2.4), and (2.5) and the additional condition

$$|f(x) - S_n(x)| \leqslant \sqrt{2}\ f(0)\ \frac{(\lambda x)^{2n}}{2^{2n-2}\,(2n)!}\,, \quad x\varepsilon\mathbf{R}\,, \tag{2.10}$$

where $S_n(x)$ is given by (2.2). Furthermore, the left-hand side of (2.10) is never larger than $2f(0)$ for all $x\varepsilon\mathbf{R}$ and every integer $n \geqslant 1$.

Proof. Let $\alpha_1, \ldots, \alpha_n$ be the distinct abscissas of an n point Gaussian quadrature formula with respect to the measure $dV(\alpha)$, and let $b_1, \ldots, b_n$ be the corresponding Cotes numbers. Let the numbers $a_1, \ldots, a_n$ be defined by the relationship $b_k = f(0)\, a_k$, $k = 1, \ldots, n$. Equations (2.3), (2.4), and (2.5) are then satisfied. Fix $x\varepsilon\mathbf{R}$. Let $p^*(\alpha)$ be any polynomial of degree at most $2n-1$ such that

$$\max_{-\lambda \leqslant \alpha \leqslant \lambda} |e^{i\alpha x} - p^*(\alpha)| = \varepsilon_n(x)\,,$$

where $\varepsilon_n(x)$ is defined by (2.7). Then, defining $S_n(x)$ as in (2.2), we have

$$\begin{aligned}
|f(x) - S_n(x)| &\leqslant |f(x) - f(0)\sum_{k=1}^{n} a_k\, p^*(\alpha_k)| \\
&\quad + |f(0)\sum_{k=1}^{n} a_k\, p^*(\alpha_k) - S_n(x)| \\
&\leqslant \int_{-\lambda}^{\lambda} |e^{i\alpha x} - p^*(\alpha)|\, dV(\alpha) \\
&\quad + f(0)\sum_{k=1}^{n} a_k\, |p^*(\alpha_k) - e^{i\alpha_k x}| \\
&\leqslant \varepsilon_n(x) \int_{-\lambda}^{\lambda} dV(\alpha) + f(0)\,\varepsilon_n(x)\sum_{k=1}^{n} a_k \\
&= 2f(0)\,\varepsilon_n(x)\,.
\end{aligned}$$

Since $\varepsilon_n(x) \leq 1$ is always true, recalling Lemma 1 completes the proof.

Corollary 1.1. Any sequence of approximations $S_n(x)$, $n = 1,2, \ldots$, satisfying Theorem 1 converges uniformly to f(x) on every finite interval.

Proof. Immediate.

Corollary 1.2. If, in addition to the requirements of Theorem 1, f(x) is real valued, then for each integer $n \geq 1$, there exists distinct real numbers $\beta_1, \ldots, \beta_n$ and real numbers $d_1, \ldots, d_n$ that satisfy

$$0 < d_k < 1 , \quad k = 1, \ldots, n \tag{2.11}$$

$$d_1 + d_2 + \ldots + d_n = 1 \tag{2.12}$$

$$0 < \beta_k < \lambda , \quad k = 1, \ldots, n \tag{2.13}$$

and are such that

$$|f(x) - f(0) \sum_{k=1}^{n} d_k \cos \beta_k x| \leq f(0) \frac{(\lambda x)^{4n}}{2^{4n-2}(4n)!} \tag{2.14}$$

for all $x \varepsilon \mathbf{R}$. Furthermore, the left-hand side of (2.14) is bounded from above by 2f(0) for all $x \varepsilon \mathbf{R}$ and every integer $n \geq 1$.

Proof. Since f(x) is real valued, by conjugating (1.1) we see that it must be even. From $f(x) = (f(x) + f(-x))/2$ and (2.9), we get

$$f(x) = \int_{-\lambda}^{\lambda} \cos \alpha x \, dV(\alpha) . \tag{2.15}$$

Furthermore, the measure $dV(\alpha)$ can be taken to be symmetric about 0. For each $n \geq 1$, and for each fixed $x \varepsilon \mathbf{R}$, define

$$\tilde{\varepsilon}_n(x) = \min_{-\lambda \leq \alpha \leq \lambda} \max |\cos \alpha x - \pi_{2n-1}(\alpha)| ,$$

where the minimum is taken over all polynomials $\pi_{2n-1}(\alpha)$ of degree at most 2n-1 with real coefficients. Hence, we always have $\tilde{\varepsilon}_n(x) \leq 1$ by considering the case $\pi_{2n-1}(\alpha) \equiv 0$. From the proof of Lemma 1,

$$\tilde{\varepsilon}_n(x) \leq \frac{(\lambda x)^{2n}}{2^{2n-1}(2n)!} , \quad x \varepsilon \mathbf{R} .$$

Duplicating the proof of Theorem 1 with 2n replacing n gives

$$|f(x) - f(0) \sum_{k=1}^{2n} a_k \cos \alpha_k x| \leq 2f(0) \tilde{\varepsilon}_{2n}(x)$$

for the distinct real numbers $\alpha_1, \ldots, \alpha_{2n}$ and real numbers $a_1, \ldots, a_{2n}$ that are the abscissas and Cotes numbers, respectively, of the Gaussian quadrature of order 2n with respect to the measure $dV(\alpha)$. These abscissas and Cotes numbers satisfy

(2.3), (2.4), and (2.5). Since the measure $dV(\alpha)$ is symmetric about zero, it must be that $\alpha_1 = -\alpha_{2n}$, $\alpha_2 = -\alpha_{2n-1}$, etc., and that $a_1 = a_{2n}$, $a_2 = a_{2n-2}$, etc. Inequality (2.14) follows immediately by taking $d_k = 2a_{n+k}$ and $\beta_k = \alpha_{n+k}$ for $k = 1, 2, \ldots, n$. The properties (2.11), (2.12), and (2.13) follow from (2.3), (2.4), and (2.5). This completes the proof.

Example 1. The real valued function

$$f(x) = 2\,\frac{\sin x}{x} = \int_{-1}^{1} e^{i\alpha x}\, dV(\alpha) \tag{2.16}$$

with $V(\alpha) = \alpha$, $-1 \leqslant \alpha \leqslant 1$, is a positive definite function on **R**. In this case, Gaussian quadrature with respect to the measure $dV(\alpha)$ is Gauss-Legendre quadrature. Thus, from the proof of Corollary 1.2, for each $n \geqslant 1$, we have

$$\left| \frac{\sin x}{x} - \sum_{k=1}^{n} d_k \cos \beta_k x \right| \leqslant \frac{x^{4n}}{2^{4n-2}(4n)!}\,, \tag{2.17}$$

where $\beta_1, \ldots, \beta_n$ are the positive abscissas of a 2n point Gauss-Legendre quadrature and $d_1, \ldots, d_n$ are the corresponding Cotes numbers. This example and some computation provides one test of the quality of the error term. Let $R_n(x)$ be the smaller of the two numbers $2f(0) = 2$ and

$$\frac{x^{4n}}{2^{4n-2}(4n)!} \approx \left(\frac{xe}{8n}\right)^{4n} \sqrt{\frac{2}{\pi n}}\,. \tag{2.18}$$

From Table 1, it appears that $R_n(x)$ is an excellent error bound provided $|x| << 8n/e$. (Table 1 was computed on a DEC VAX 11/780 on which the double precision unit round-off error is only 4×10^{-17}.)

Table 1. Comparison of (2.17) for $n = 10$ ($8n/e \approx 29.43$)

x	$R_{10}(x)$	$f(x) - S_{10}(x)$	$\max_{0 \leqslant y \leqslant x} \lvert f(y) - S_{10}(y) \rvert$,
5	$.407 \times 10^{-31}$	underflow	underflow
10	$.447 \times 10^{-19}$	underflow	underflow
15	$.494 \times 10^{-12}$	$.491 \times 10^{-13}$	$.491 \times 10^{-13}$
20	$.491 \times 10^{-7}$	$.164 \times 10^{-8}$	$.164 \times 10^{-8}$
25	$.370 \times 10^{-3}$	$.290 \times 10^{-5}$	$.290 \times 10^{-5}$
30	.543	$.664 \times 10^{-3}$	$.664 \times 10^{-3}$
35	$.200 \times 10^{1}$	$.302 \times 10^{-1}$	$.302 \times 10^{-1}$
40	$.200 \times 10^{1}$	.309	.309
45	$.200 \times 10^{1}$	.501	.569
50	$.200 \times 10^{1}$	-.364	.569

This section is concluded by showing that Prony's method (see, e.g., [14, p. 378] or [15, p. 340]) can be used to compute numerically the approximations of Theorem 1. We need only find the Gaussian quadrature formula of order 2n with respect to the measure $dV(\alpha)$, which is equivalent to solving the equations

$$c_m = \int_{-\lambda}^{\lambda} \alpha^m \, dV(\alpha) = \sum_{k=1}^{n} b_k \alpha_k^m, \; m = 0,1, \ldots, 2n-1 \qquad (2.19)$$

for $b_1, \ldots, b_n$ and $\alpha_1, \ldots, \alpha_n$. Since α_k must be real, write $s_k = \ln \alpha_k$, if $\alpha_k \neq 0$, and $s_k = 0$ if $\alpha_k = 0$. The required equations can now be written as

$$c_m = \sum_{k=1}^{n} b_k e^{ms_k}, \; m = 0,1, \ldots, 2n-1 .$$

This form is precisely the form required for Prony's method. (The use of Prony's method to compute Gaussian quadrature formulas was pointed out to the author by Marvin J. Goldstein.) In principle, we require 2n quadratures, the solutions of two systems of linear equations each of rank n, and the roots of a polynomial of degree n (in this case, all its roots are known to be real, distinct, and have multiplicity one) to compute one approximation for which Theorem 1 holds. Unfortunately, it is known [16] that any procedure that relies upon the moments must become increasingly numerically ill-conditioned as n increases. Fortunately, the use of modified moments (i.e., replacing the α^m in (2.19) with some classical system of orthogonal polynomials) together with an algorithm other than Prony's method often results in a numerically well-conditioned problem for finite intervals. See [17] and [18] for details.

III. Exponential Approximation With Uniform Coefficients

The idea developed in this section for constructing approximations of the form $Q_n(x)$ in which each exponential term enters the approximation with equal weight is basically probabilistic in nature. The integral representation (1.3) of f(x) is approximated by a Riemann sum whose subintervals are equally probable according to the "probability" measure $dV(\alpha)$. In this interpretation, $V(\alpha)$ is a cumulative probability integral that is used to transform n uniformly distributed points in the range of $V(\alpha)$ into n abscissas on the real line distributed according to the measure $dV(\alpha)$. (See [19, p. 314] or [15, p. 389].)

Theorem 2. Let f(x) be a continuous complex valued positive definite function of a real variable such that

$$f(x) = \int_{-\lambda}^{\lambda} e^{i\alpha x}\, dV(\alpha) , \qquad x\varepsilon\mathbf{R}, \tag{3.1}$$

where V is a continuous and strictly monotone increasing function throughout the finite closed interval $[-\lambda, \lambda]$. Then, for each integer $n \geqslant 1$, there exists distinct real numbers $\alpha_1, \ldots, \alpha_n$ in the open interval $(-\lambda, \lambda)$ such that

$$|f(x) - Q_n(x)| \leqslant 2\sqrt{2}\ f(0)\, \lambda |x|/n , \qquad x\varepsilon\mathbf{R} , \tag{3.2}$$

where

$$Q_n(x) = \frac{f(0)}{n} \sum_{k=1}^{n} e^{i\alpha_k x} . \tag{3.3}$$

Furthermore, from the remark following (1.6), the left-hand side of (3.2) is bounded from above by 2f(0) for all $x\varepsilon\mathbf{R}$.

Proof. Let the real number x be fixed throughout this proof. Define, for k = 0,1,2, . . . , 2n,

$$\begin{aligned} u_k &= V(-\lambda) + (V(\lambda) - V(-\lambda))\ k/2n \\ v_k &= V^{-1}(u_k) . \end{aligned} \tag{3.4}$$

Under the hypotheses on $V(\alpha)$, it is clear that V^{-1} exists and is continuous and strictly monotone on the closed interval $[V(-\lambda), V(\lambda)]$. Hence, the numbers v_k in (3.4) are well defined and are distinct. It will be shown that inequality (3.2) holds for

$$\alpha_k = v_{2k-1} , \quad k = 1,2, \ldots , n . \tag{3.5}$$

Since

$$V(v_{k+1}) - V(v_k) = (V(\lambda) - V(-\lambda))/2n = f(0)/2n \quad , k = 0,1, \ldots , 2n-1 , \tag{3.6}$$

it follows from the definition (3.3) of $Q_n(x)$ that

$$Q_n(x) = \sum_{k=1}^{n} \int_{v_{2k-2}}^{v_{2k}} \exp(i\, v_{2k-1} x)\, dV(\alpha) . \tag{3.7}$$

By the Mean Value Theorem, there exists ζ_k in the interval between α and v_{2k-1} such that

$$\cos \alpha x - \cos v_{2k-1} x = -x(\alpha - v_{2k-1}) \sin \zeta_k x . \tag{3.8}$$

Thus, $\alpha < v_{2k-1}$ implies $\alpha < \zeta_k < v_{2k-1}$ and $v_{2k-1} < \alpha$ implies $v_{2k-1} < \zeta_k < \alpha$. From (3.1) and (3.7),

$$\begin{aligned} \mathrm{Re}(f(x) - Q_n(x)) &= \sum_{k=1}^{n} \int_{v_{2k-2}}^{v_{2k}} (\cos \alpha x - \cos v_{2k-1} x)\ dV(\alpha) \\ &= -x \sum_{k=1}^{n} \int_{v_{2k-2}}^{v_{2k}} (\alpha - v_{2k-1}) \sin \zeta_k x\ dV(\alpha) \end{aligned} \tag{3.9}$$

and so, taking absolute values,

$$\begin{aligned} |\mathrm{Re}(f(x) - Q_n(x))| &\leqslant |x| \sum_{k=1}^{n} \int_{v_{2k-2}}^{v_{2k}} |\alpha - v_{2k-1}|\, dV(\alpha) \\ &\leqslant |x| \sum_{k=1}^{n} (v_{2k} - v_{2k-2})\,(V(v_{2k}) - V(v_{2k-2})) \\ &= 2\lambda |x|\, f(0)/n . \end{aligned} \tag{3.10}$$

where (3.6) was used in the last step. Similarly,

$$|\mathrm{Im}\,(f(x) - Q_n(x))| \leqslant 2\lambda |x|\, f(0)/n . \tag{3.11}$$

Clearly, (3.10) and (3.11) together complete the proof.

Corollary 2.1. Any sequence of approximations $Q_n(x)$, $n = 1,2, \ldots$, satisfying Theorem 2 converges uniformly to $f(x)$ on every finite interval.

Proof. Immediate.

Corollary 2.2. If, in addition to the requirements of Theorem 2, $f(x)$ is real valued, then for each integer $n \geqslant 1$, there exist distinct real numbers $\beta_1, \ldots, \beta_n$ in the open interval $(0,\lambda)$ such that

$$\left| f(x) - \frac{f(0)}{n} \sum_{k=1}^{n} \cos \beta_k x \right| \leqslant f(0)\, \lambda |x| / n . \tag{3.12}$$

Furthermore, the left hand side of (3.12) is never larger than $2f(0)$ for all $x \varepsilon \mathbf{R}$ and integer $n \geqslant 1$.

Proof. Recalling (2.15), follow the proof of Theorem 2 with $2n$ replacing n throughout. Half of the resulting $2n$ α_k's are positive. Set the β_k's equal to the positive α_k's. The details are immediate. This concludes the proof.

The proof of Theorem 2 requires that V be continuous and strictly monotone increasing. It is not clear whether the hypotheses on V can be weakened. On the other hand, the convergence rate of the approximations Q_n can apparently be improved by making further assumptions concerning V. In general, however, better than n^{-2} convergence rates cannot be expected. Consider Example 1, where $V(\alpha)=\alpha$ and $f(x) = x^{-1} \sin x$. From the construction indicated in Corollary 2.2, f(x) is approximated by

$$Q_n(x) = \frac{1}{n} \sum_{k=1}^{n} \cos (2k-1)x/2n = \frac{\sin x}{2n \sin x/2n} . \tag{3.13}$$

It can be shown directly that

$$\frac{|x \sin x|}{24n^2} \leqslant \left| \frac{\sin x}{x} - Q_n(x) \right| \leqslant \frac{x^2}{24n^2} \quad , x \varepsilon \mathbf{R} , \tag{3.14}$$

and

$$\lim_{n \to \infty} n^2[x^{-1} \sin x - Q_n(x)] = \frac{-x \sin x}{24} . \tag{3.15}$$

Hence, in this example, the correct convergence rate is precisely n^{-2} for each fixed x. We point out that the upper bound in (3.14) and the limit (3.15) follow by a trivial application of a suggestive result in Pólya-Szegö [20, Pt. 2, Ch. 1, Pr. 11]. Regrettably, their method seems applicable in this application only to the special measure $V(\alpha) = \alpha$.

A further example seems to indicate that the convergence rate of the approximations Q_n can lie between n^{-1} and n^{-2}.

Example 2. [4, p. 22] Let λ be a finite positive real number. Define V for non-negative arguments $\alpha \geqslant 0$ by

$$V(\alpha) = \begin{cases} \alpha(1-\alpha/2\lambda) & , 0 \leqslant \alpha \leqslant \lambda, \\ \lambda/2 & , \lambda < \alpha, \end{cases}$$

and for negative arguments by the relation $V(-\alpha) = -V(\alpha)$, $\alpha > 0$. Thus, V is an odd function whose derivative $V'(\alpha) = 1 - |\alpha|/\lambda$. Obviously, for all $x \neq 0$,

$$f(x) = \int_{-\lambda}^{\lambda} \cos \alpha x \, dV(\alpha) = \frac{2(1-\cos \lambda x)}{\lambda x^2} ,$$

and $f(0) = \lambda$. Now V^{-1} exists on the interval $[-\lambda, \lambda]$ and, for non-negative arguments, is given by

$$V^{-1}(t) = \lambda(1 - \sqrt{1 - 2t/\lambda}) , \quad 0 \leqslant t \leqslant \lambda/2 .$$

From the construction indicated in Corollary 2.2,

$$Q_n(x) = \frac{\lambda}{n} \sum_{k=1}^{n} \cos \left[\lambda(1-\sqrt{1 - (k - \tfrac{1}{2})/n}\ ) x \right], \tag{3.16}$$

and

$$|f(x) - Q_n(x)| \leq \lambda^2 |x|/n , \quad x \varepsilon \mathbf{R} .$$

This estimate is not even close to the truth. In fact, an examination of Table 2 indicates that for sufficiently large n the best error bound may take the form

$$|f(x) - Q_n(x)| \leq \frac{Kx^2}{(2n)^{3/2}} \tag{3.17}$$

for some constant K. In general, we speculate that if V^{-1} satisfies a Lipschitz condition of order r, $0 < r \leq 1$, then the convergence rate is of order $1/n^{1+r}$.

Table 2. Inequality (3.17) for $x = 10$, $\lambda = 1$, $K = 10^{-2}$

n	$Kx^2/(2n)^{3/2}$	$f(x) - Q_n(x)$
5	$.316 \times 10^{-1}$	$-.618 \times 10^{-1}$
10	$.112 \times 10^{-1}$	$-.117 \times 10^{-1}$
20	$.395 \times 10^{-2}$	$-.153 \times 10^{-2}$
40	$.140 \times 10^{-2}$	$.649 \times 10^{-4}$
80	$.494 \times 10^{-3}$	$.163 \times 10^{-4}$
160	$.175 \times 10^{-3}$	$.907 \times 10^{-4}$
320	$.618 \times 10^{-4}$	$.400 \times 10^{-4}$
640	$.218 \times 10^{-4}$	$.160 \times 10^{-4}$
1280	$.772 \times 10^{-5}$	$.614 \times 10^{-5}$
2560	$.273 \times 10^{-5}$	$.229 \times 10^{-5}$
5120	$.965 \times 10^{-6}$	$.837 \times 10^{-6}$
10240	$.341 \times 10^{-6}$	$.303 \times 10^{-6}$
20480	$.121 \times 10^{-6}$	$.109 \times 10^{-6}$
40960	$.426 \times 10^{-7}$	$.389 \times 10^{-7}$

IV. Concluding Remarks

The proofs in this report depend heavily on the finite support of the measure $dV(\alpha)$ even though the construction of the approximations $S_n(x)$ and $Q_n(x)$ can be carried out without modification on infinite intervals as well, provided $V(\alpha)$ is bounded. Since these proofs cannot be adapted for infinite intervals, the effectiveness of the resulting approximations theoretically remains an open question. Intuitively, however, it would seem that only our proofs are limited and that the underlying approximation process is generally valid.

In computational practice the function V is usually unknown. In many cases, however, the given function f does possess a nicely behaved Fourier transform from which V can be readily constructed. The Fourier transform of f can, of course, be computed accurately and efficiently in many situations using fast Fourier transform (FFT) methods.

If in the above, V was not monotonic, but of bounded variation on **R**, exponential approximations can be constructed as follows. In this case, there exist monotone increasing functions V^+ and V^- such that $V = V^+ - V^-$. For each $\lambda > 0$, define the "bandlimiting" operator B_λ by

$$B_\lambda f(x) = \int_{-\lambda}^{\lambda} e^{i\alpha x}\, dV(\alpha)\ ,\quad x\varepsilon\mathbf{R}\ .$$

Let I be any finite interval. Given $\varepsilon > 0$, choose $\lambda > 0$ so that $||f - B_\lambda f|| < \varepsilon$, where the norm is the uniform norm over the interval I. Now, let the two functions

$$B_\lambda^{\pm} f(x) \equiv \int_{-\lambda}^{\lambda} e^{i\alpha x}\, dV^{\pm}(\alpha)\ ,\quad x\varepsilon\mathbf{R}\ ,$$

be approximated using either of the methods of this paper by the two finite exponential sums, say, $E^{\pm}(x)$, so that

$$||B_\lambda^{\pm} f - E^{\pm}|| < \varepsilon\ .$$

Let $E(x) = E^+(x) - E^-(x)$. Since $B_\lambda f = B_\lambda^+ f - B_\lambda^- f$, we have

$$\begin{aligned} ||f - E|| &= ||(f - B_\lambda f) + (B_\lambda^+ f - E^+) - (B_\lambda^- f - E^-)|| \\ &\leq ||f - B_\lambda f|| + ||B_\lambda^+ f - E^+|| + ||B_\lambda^- f - E^-|| < 3\varepsilon\ . \end{aligned}$$

Thus, exponential sums of degree not greater than $\deg(E^+) + \deg(E^-)$ may be constructed to approximate f(x) with specified accuracy on the interval I.

It is well known [4, p. 60] that if f(x) is measurable, then Bochner's Theorem still holds for almost all x, but not necessarily all x as in (1.2). Results similar to the results of this report can also be proven for measurable f with careful attention to certain details; however, this generalization is not pursued here. Similarly, Bochner's Theorem has been generalized to locally compact abelian groups [4, p. 72], and perhaps the basic approaches to approximation used here can be extended to this much more general setting.

We conclude by commenting that Bernstein's Theorem [21, p. 160] states that a necessary and sufficient condition for f(x) to be completely monotonic on the interval $(0,\infty)$ is that

$$f(x) = \int_0^\infty e^{-\alpha x}\, dV(\alpha)\,, \quad 0 \leq x < \infty\,,$$

where $V(\alpha)$ is bounded and nondecreasing. It is evident that the methods employed in this report can be used in a manner entirely analogous to the proof of Theorem 1 to develop exponential approximations. That is to say, whenever f(x) can be expressed as

$$f(x) = \int_0^\lambda e^{-\alpha x}\, dV(\alpha)$$

for some finite $\lambda > 0$, there exist approximations of the form

$$T_n(x) = f(0) \sum_{k=1}^{n} a_k\, e^{-\alpha_k x}$$

where

$$a_k > 0\,,\ k = 1, \ldots, n \tag{4.1}$$

$$a_1 + \ldots + a_n = 1 \tag{4.2}$$

$$\lambda > \alpha_k > 0\,,\ k = 1, \ldots, n \tag{4.3}$$

and

$$|f(x) - T_n(x)| \leq 2f(0)\, \frac{(\lambda x)^{2n}}{2^{2n}}\, e^{-\lambda x/2}, \quad 0 \leq x < \infty\,. \tag{4.4}$$

An alternate approach for approximation of completely monotonic functions can be found in [22].

V. References

1. J. D. Bruce and H. Unz, "Mechanical Quadratures to Synthesize Nonuniformly-Spaced Antenna Arrays," *Proc. of the IRE (Correspondence)*, vol. 50, Oct. 1962, p. 2128.

2. A. L. Maffett, "Array Factors with Nonuniform Spacing Parameter," *IRE Trans. on Ant. and Prop.*, vol. AP-10, March 1962, pp. 131-136.

3. J. Stewart, "Positive Definite Functions and Generalizations, an Historical Survey," *Rocky Mountain J. Math.*, vol. 6, No. 3, 1976, pp. 409-434.

4. R. R. Goldberg, *Fourier Transforms*, Cambridge University Press, 1970.

5. R. Schabach, "Suboptimal Exponential Approximations," *SIAM J. Numer. Anal.*, vol. 16, no. 6, December 1979, pp. 1007-1018.

6. D. W. Kammler and R. J. McGlinn, "A Bibliography for Approximation With Exponential Sums," *J. Comp. and App. Math.*, vol. 4, No. 2, 1978, pp. 167-173.

7. N. I. Achieser, *Theory of Approximation*, Ungar Publishing Co., 1956.

8. G. J. van der Maas, "A Simplified Calculation of Dolph Tchebycheff arrays," *J. of Applied Physics*, vol. 25, January 1954, pp. 121-124.

9. V. Barcilon and G. C. Temes, "Optimum Impulse Response and the van der Maas Function," *IEEE Trans. on Circuit Theory*, vol. CT-19, July 1972, pp. 336-342.

10. R. J. Duffin and A. C. Schaeffer, "Some Properties of Functions of Exponential Type," *Bull. Amer. Math. Soc.*, vol. 44, 1938, pp. 236-240.

11. A. Erdelyi, Editor, *Tables of Integral Transforms*, vol. 1, Bateman Manuscript Project, McGraw-Hill, 1954.

12. G. Szegö, *Orthogonal Polynomials*, American Mathematical Society Colloquium Publications, vol. 23, Fourth Edition, 1978.

13. G. Meinardus, *Approximation of Functions: Theory and Numerical Methods*, Springer-Verlag, 1967.

14. F. B. Hildebrand, *Introduction to Numerical Analysis*, First Edition, McGraw-Hill, 1956.

15. R. W. Hamming, *Numerical Methods for Scientists and Engineers*, First Edition, McGraw-Hill, 1962.

16. W. Gautschi, "Construction of Gauss-Christoffel Quadrature Formulas," *Math. Comp.*, vol. 22, 1968, pp. 251-270.

17. W. Gautschi, "On the Construction of Gaussian Quadrature Rules From Modified Moments," *Math. Comp.*, vol. 24, 1970, pp. 245-260.

18. R. A. Sack and A. F. Donovan, "An Algorithm for Gaussian Quadrature Given Modified Moments," *Numer. Math.*, vol. 18, 1972, pp. 465-478.

19. E. Parzen, *Modern Probability Theory and Its Applications*, John Wiley and Sons, Inc., 1960.

20. G. Pólya and G. Szegö, *Problems and Theorems in Analysis I*, English Edition, Springer-Verlag, 1972.

21. D. V. Widder, *The Laplace Transform*, Princeton University Press, 1946.

22. D. W. Kammler, "Prony's Method for Completely Monotonic Functions," *J. of Math. Anal. and Appl.*, vol. 57, 1977, pp. 560-570.

FREQUENCY LINE DETECTOR/TRACKERS

Foreword

Hidden Markov models (HMMs) are well known for their application to automatic speech recognition problems, where they are used to characterize the time variation of short term Fourier spectra of the broad band speech signal. HMMs are useful outside the speech application as well. The application of HMMs to the problem of detecting and tracking time varying frequency lines is presented in detail in paper [13]. One unique aspect of this approach is that tracks are automatically initiated and terminated as an intrinsic function of the underlying HMM algorithm For this reason, the tracker is referred to as an HMM detector/tracker. Another interesting aspect is that the finite state HMM enables the non-Gaussian nature of the measurement process to be modeled exactly. Papers [14] and [15] are the first to present the application of HMMs to frequency line detection and tracking.

Papers [16] and [17] describe extensions of the HMM detector/tracker presented in [13] to include the exploitation of the phase and amplitude information in the received signal. The inclusion of this information affects the state conditional measurement likelihood functions, but it does not alter the fundamental character of the HMM detector/tracker algorithm.

The ability of HMM detector/trackers to estimate signal-to-noise ratio (SNR) is documented in [18]. The SNR estimation algorithm is a maximum likelihood algorithm, and it is derived from a variation of the Baum-Welch training algorithm for general HMMs.

Detection performance can be studied by using an HMM as a signal source model, either matched or mismatched to the HMM of the detector/tracker. The detection capability of HMM detector/trackers was first studied in this way in [19]. Paper [20] also discusses the use of HMMs as signal sources.

Various advanced and special purpose computer architectures have been proposed for implementation of HMMs for the speech application. Paper [21] studies HMM detector/tracker implementations on the Connection Machine supercomputer.

Frequency Line Tracking Using Hidden Markov Models

R. L. Streit and R. F. Barrett

Frequency Line Tracking Using Hidden Markov Models

ROY L. STREIT, SENIOR MEMBER, IEEE, AND ROSS F. BARRETT

Abstract—**This paper demonstrates how the problem of frequency line tracking can be formulated in terms of hidden Markov models (HMM's). Frequency cells comprising a subset, or gate, of the spectral bins from FFT processing are identified with the states of the hidden Markov chain. An additional zero state is included to allow for the possibility of track initiation and termination. Analytic expressions are obtained for the basic parameters of the HMM in terms of physically meaningful quantities, and optimization of the HMM tracker is carefully discussed. A measurement sequence based on a simple threshold detector forms the input to the tracker. The outputs of the HMM tracker are a discrete Viterbi track, a gate occupancy probability function, and a continuous mean cell occupancy track. The latter provides an estimate of the mean signal frequency as a function of time. The performance of the HMM tracker is evaluated for two sets of simulated data and is found to be remarkably good, comparing favorably to results from inspection of the signal spectrograms. A comparison of the HMM tracker to earlier, related trackers is presented, and possible extensions are discussed.**

I. Introduction

THE estimation of the frequency of isolated tones embedded in a noise background is a problem that is of interest in diverse fields (e.g., seismology, radar, sonar, radioastronomy, etc.), and is currently receiving considerable attention in the signal processing literature (e.g., see [1]–[4]). In the case where the frequency of the tone is changing as a function of time, a related problem is that of accurately tracking these changes in frequency.

One obvious approach is to divide the time series into finite-sized blocks, and to apply one of the many new frequency estimation techniques to the data in each block. The result is a sequence of independent frequency estimates which, if the signal-to-noise ratio (SNR) is reasonably high, provides an accurate estimate of the underlying frequency variations. However, as the SNR is reduced, the scatter in the frequency estimates becomes large, and "outliers," or estimates far from the true frequency track, become common. *A priori* knowledge of the extent and rapidity of the likely frequency changes can be incorporated into an algorithm that rejects the highly improbable outliers and produces smoothed frequency estimates as a function of time. Such an algorithm is designated here a "frequency tracker."

The purpose of this paper is to show that the problem of frequency tracking lends itself readily to formulation in terms of a hidden Markov model (HMM). These models are used in speech applications to characterize the time variation of the short-term spectra of spoken words. The basic principles of HMM's are reviewed in Section II. For more detailed discussions of HMM's, the reader is referred to [5], [6] and to the references cited therein.

The HMM's utilized in this paper are comprised of two basic parts: a Markov chain, and a set of discrete finite-outcome random variables. The Markov chain has a finite number of states and is characterized by its transition probability matrix A. The elements of the A matrix are the probabilities of transitioning between the states of the Markov chain. The set of random variables is characterized by a measurement probability matrix B. The elements of the B matrix are the probabilities defining the probability density functions (pdf's) of the finite-outcome random variables. Each state of the Markov chain is uniquely associated with one of the random variables.

The relevance of the HMM to frequency tracking is easy to see. The range of frequencies over which the track is allowed to wander is divided into a finite number of frequency cells, and each cell is associated with a state of the Markov chain. In addition, a zero state is included to allow for the possibility of the track wandering outside the allowed frequency range or terminating altogether. The A matrix represents our knowledge, based on past experience, of the likely extent of the frequency fluctuations, or of the track terminating, or of it restarting after a previous termination. The inclusion of the zero state is an important feature, and its presence precludes a simple characterization of track variation as a Gaussian statistic.

The B matrix characterizes the connection between the underlying state at time t and the measurement at time t. For the HMM frequency tracker presented in this paper, the measurement takes the form of a detection. A detection is said to have occurred in a particular frequency cell at time t if the spectral power in that cell at that time exceeds a certain threshold D and is larger than the power in all other cells within the allowed frequency range. If the power in each cell is less than D, a detection in the zero state is said to have occurred. Since the B matrix connects the underlying states with the noisy measurements, it depends on the SNR and the nature of the back-

Manuscript received August 11, 1988; revised March 28, 1989.

R. L. Streit is with the Naval Underwater Systems Center, New London, CT 06320, on leave at the Weapons Systems Research Laboratory, Maritime Systems Division, Defence Science and Technology Organisation, Salisbury, South Australia, Australia.

R. F. Barrett is with the Weapons Systems Research Laboratory, Maritime Systems Division, Defence Science and Technology Organisation, Salisbury, South Australia, Australia.

IEEE Log Number 9034283.

ground noise. For the HMM tracker presented here, the B matrix is computed analytically. The threshold detector applied in this way results in a measurement sequence that is highly non-Gaussian in character.

Once the connection between the HMM and the frequency tracking problem is correctly formulated, the wide body of existing knowledge on HMM's is exploited to yield both discrete and continuous tracker outputs. The highly efficient Viterbi algorithm is used to obtain the maximum likelihood frequency track, conditioned on a given set of measurements. We refer to this track as the Viterbi track; it is the discrete output of the HMM tracker. The forward-backward algorithm is used to compute the mean cell occupancy track and the probability that no frequency track is present; both are conditioned on the measurement sequence. These are the continuous outputs of the HMM tracker.

The HMM tracker presented in this paper minimizes the effect of noise by looking at the overall track time history for global structure. In practice, only a fixed length T of the measured track is utilized; thus, T measurements are stored before the output HMM track is calculated. It does not follow, however, that the HMM tracker is a fixed lag tracker with lag $T - 1$. It is shown in Section III-C that the HMM tracker can be used as a fixed lag tracker with any lag from 0 to $T - 1$.

Section III describes the HMM frequency tracking algorithm in detail, and discusses how the HMM tracker can be optimized for a particular application. The performance of the HMM tracker on simulated data is discussed in Section IV. Section V compares the HMM tracker and related work by Kopec [7] for formant tracking using HMM's in the field of speech processing. Two earlier frequency trackers described by Scharf *et al.* [8] and Jaffer *et al.* [9] are also based on HMM's, although it was not recognized at the time, and they are also discussed in Section V. Possible extensions of the HMM tracker are discussed in Section VI. The conclusions of the paper are presented in Section VII.

II. Elements of Hidden Markov Models

A finite Markov chain has a finite number $n + 1$ of states where $n \geq 0$, and is characterized by its transition probability matrix, denoted $A = [a_{ij}]$ where $i, j = 0, 1, \cdots, n$. Let π be the initial state probability vector of the Markov chain. Thus, at the start time $t = 1$, the probability that the Markov chain is in state i is π_i. The probability that the chain transitions from state i at time t to state j at time $t + 1$ is a_{ij} where $t = 1, 2, 3, \cdots$. Note that the transition probabilities a_{ij} are independent of time t. Markov chains with an infinite number of states are not considered.

The tracking application presented in this paper requires only HMM's with a finite number of different possible outcomes or measurements. However, we also consider HMM's with measurements that are arbitrary complex-valued vectors because this kind of HMM is useful in some applications (see below in Section V). The pdf of the random variable uniquely associated with state i of the Markov chain is denoted by $b_i(z)$ where z is a measurement. Let B denote the vector $[b_i(z)]$. For finite-outcome HMM's, we also use B to denote the measurement probability matrix $B = [b_{ij}]$ where $b_{ij} = b_i(z_j)$ and z_j runs through the finite measurement set. This abuse of notation should not cause confusion. (What we call a measurement is referred to as a "symbol" in the speech literature [5], [6].)

Simulation of an HMM measurement sequence of length T, given π, A, and B, is straightforward. The initial state of the Markov chain is chosen according to the initial state probability vector π. The initial state uniquely determines the first pdf. The first measurement, say $z(1)$, is chosen according to this first pdf. Next, the Markov chain transitions to another (or the same) state according to its transition probability matrix A. This state determines the second pdf, and the second measurement, say $z(2)$, is chosen according to this second pdf. Continuing in this fashion up to time $t = T$ generates the measurement sequence

$$Z_T = \{z(1), z(2), \cdots, z(T)\}. \tag{2.1}$$

It is important to note that the only output from such an HMM simulator is the measurement sequence Z_T. The state sequence of the Markov chain is not an output.

From the HMM simulation procedure, it is clear that the total probability of a given measurement sequence is the sum

$$P[Z_T] = \sum_I P[Z_T | I] \tag{2.2}$$

where $I = \{I(1), I(2), \cdots, I(T)\}$ denotes an arbitrary Markov chain state sequence of length T, and $P[Z_T | I]$ denotes the probability of Z_T, conditioned on knowledge of the state sequence. Explicitly, we have (consider the simulation procedure)

$$P[Z_T | I] = \{\pi_{I(1)} b_{I(1)}[z(1)]\}\{a_{I(1),I(2)} b_{I(2)}[z(2)]\} \cdots \{a_{I(T-1),I(T)} b_{I(T)}[z(T)]\}. \tag{2.3}$$

As is intuitively clear from the HMM simulation procedure, some state sequences I are, in general, more likely than other state sequences to correspond to the given Z_T. In other words, some terms in the summation (2.2) are larger than others. Since the "true" state sequence corresponding to Z_T is not observable, we define an optimality criterion and use it to select an "optimal" state sequence. We define an optimal state sequence to be any state sequence for which the probability $P[Z_T | I]$ is a maximum. Optimal state sequences are not necessarily unique because the maximum $P[Z_T | I]$ may not be uniquely attained; however, the HMM's developed in this paper appear to give unique optimal state sequences, except in situations which are of little or no importance in the application. Other definitions of optimality are possible and potentially useful (see [5]), but only the definition above is used in this paper.

We refer to an optimal state sequence corresponding to a given Z_T as the Viterbi track, denoted $I_V[Z_T]$, and to the probability

$$P_V[Z_T] = \max_I P[Z_T|I] \tag{2.4}$$

as the Viterbi score of Z_T. Both the Viterbi track and the Viterbi score are easily computed using the so-called Viterbi dynamic programming algorithm. The Viterbi algorithm gives the globally optimal state sequence as required by (2.4). Furthermore, the computational complexity of the Viterbi algorithm is linear in T, the length of the measurement sequence. This makes it a very efficient algorithm in many applications.

For a given observation sequence (2.1), the Viterbi algorithm is defined as follows. For $t = 1$, define

$$\delta_1(j) = \ln \pi_j + \ln b_j(z(1)), \quad 0 \le j \le n$$
$$\psi_1(j) = \text{arbitrary} \tag{2.5}$$

and, for $t = 2, 3, \cdots, T$, define

$$\delta_t(j) = \ln b_j(z(t)) + \max_{0 \le i \le n} \{\delta_{t-1}(i) + \ln a_{ij}\} \tag{2.6a}$$

$$\psi_t(j) = \operatorname*{argmax}_{0 \le i \le n} \{\delta_{t-1}(i) + \ln a_{ij}\} \tag{2.6b}$$

where the argmax function gives the smallest index i for which the maximum is attained. The Viterbi score is

$$P_V[Z_T] = \max_{0 \le j \le n} \{\delta_T(j)\} \tag{2.7}$$

and the Viterbi track is given by

$$I_V[Z_T] = \{I_V(1), I_V(2), \cdots, I_V(T)\}$$

where

$$I_V(T) = \operatorname*{argmax}_{0 \le j \le n} \{\delta_T(j)\} \tag{2.8a}$$

and for $t = T-1, T-2, \cdots, 1$,

$$I_V(t) = \psi_{t+1}(I_V(t+1)). \tag{2.8b}$$

For greater computational efficiency, the natural logarithms of the components of π, A, and B are usually precomputed and stored for finite-outcome HMM's.

The so-called forward–backward algorithm is used to provide state occupancy information on the hidden Markov chain state sequence. We define the forward probabilities $\alpha_t(j)$ by

$$\alpha_t(j) = P[z(1), \cdots, z(t) \text{ and } I(t) = j], \quad 1 \le t \le T \tag{2.9a}$$

and the backward probabilities $\beta_t(j)$ by

$$\beta_t(j) = P[z(t+1), z(t+2), \cdots, z(T)|I(t) = j], \quad 1 \le t \le T-1. \tag{2.9b}$$

The probabilities $\alpha_t(j)$ are calculated with the recursion

$$\alpha_1(j) = \pi_j b_j(z(1))$$
$$\alpha_t(j) = b_j(z(t)) \sum_{i=0}^{n} \alpha_{t-1}(i) a_{ij}, \quad t = 2, \cdots, T \tag{2.10a}$$

and the probabilities $\beta_t(j)$ are calculated with the recursion

$$\beta_T(j) = 1$$
$$\beta_t(j) = \sum_{i=0}^{n} a_{ji} b_i(z(t+1)) \beta_{t+1}(i). \tag{2.10b}$$

We define the state occupancy probabilities at time t by

$$\gamma_t(i) = \alpha_t(i) \beta_t(i)/P[Z_T] \tag{2.11}$$

so that

$$\sum_{i=0}^{n} \gamma_t(i) = 1.$$

The state occupancy probability $\gamma_t(i)$ is interpreted as the probability that the Markov chain occupies state i at time t, conditioned on the measurement sequence. We shall see in Section III how the state occupancy probabilities are used to define the continuous output of the HMM frequency tracker.

The computational complexity of the Viterbi algorithm is $(n+1)^2T$ additions if the natural logarithm of the components of the B matrix can be stored. If the measurement pdf vector B must be computed for each symbol in Z_T, then the complexity is $[(n+1)^2 + c_1]T$ additions where c_1 is the complexity (measured in units equivalent to addition) of computing the natural logarithm of the components of the vector B for an arbitrary measurement.

The computational complexity of the forward–backward algorithm is $(n+1)^2T$ multiplications if the B matrix is stored and $[(n+1)^2 + c_2]T$ multiplications if the measurement pdf vector B is computed for each measurement in Z_T where c_2 is the complexity (measured in units equivalent to multiplication) of computing the components of the vector B for an arbitrary measurement. The imperative need to rescale to prevent underflow (discussed in [6]) requires an additional $(n+1)T$ divisions.

An important concept in the application of HMM's is "training." In the case when many different measurement sequences are known for the same HMM, maximum likelihood estimates of the model parameters π, A, and B can be computed using the so-called Baum–Welch reestimation algorithm. Training is not used for the frequency tracker presented in this paper. Further background information on HMM's, including a discussion of training, is found in [5], [6], [10] and in the references cited therein.

III. Frequency Tracking with Hidden Markov Models

A permissible frequency track is defined to be any state sequence that is a realization of the Markov chain char-

acterized by π and the A matrix. The states representing frequency cells are numbered from 1 to n and are referred to as the nonzero states. The collection of nonzero states is called the gate, and the gate size is said to be n. The unique state representing the absence of the frequency track is numbered 0 and is referred to as the zero state. Track initiation and track termination are defined to occur whenever the state sequence transitions out of and into the zero state, respectively. The SNR does not affect the transition probabilities between the nonzero states because these transitions are related only to possible frequency track variations inside the gate; however, the SNR does influence track initiation and termination. An important issue in optimizing the performance of the HMM tracker is the definition of the row and column of the A matrix corresponding to transitions out of and into the zero state. This issue is discussed in Section III-D.

The frequency track is not directly observable, except at infinite SNR, and is inferred from measured data. The measurements are random functions of the frequency state, and the pdf's of these random functions constitute the B matrix of the HMM tracker. As discussed in Section I, a simple threshold detector is used to estimate which frequency cell, if any, in the gate is occupied by the frequency track. The measurements are, therefore, estimates of the state of the Markov chain. By setting a detection threshold D to control false alarms, it is possible to observe the zero state. The size of D depends on the SNR, but it also affects track initiation and termination. For example, if D is too large, no detections are made and only the zero state track (i.e., no track) is measured. Consequently, setting the detection threshold is an important issue in optimizing the performance of the HMM tracker. This issue is discussed in Section III-D.

The threshold detector is not the only detector possible, but it is one that is commonly used in practice when frequency cells (usually FFT bins) constitute the tracker input. Besides its simplicity, our main purpose in using the threshold detector is to show that the HMM tracker is appropriate whenever it is possible to estimate (analytically or otherwise) the B matrix from some underlying model of the physical process. The performance of the frequency tracker is obviously tied closely to the particular detector selected.

The HMM tracker accepts as input a measured track (i.e., a sequence of measured states) and produces both discrete and continuous output tracks. The discrete output is the Viterbi track or maximum likelihood estimate of the state sequence. The Viterbi track is necessarily a valid track, i.e., the Viterbi track is a realization of the Markov chain characterized by π and A. However, the Viterbi track is not a typical random walk because it is conditioned on the measurements. For example, the Viterbi track, the true track, and the measured track all coincide for infinite SNR. In this paper, the HMM tracker is set up so that "smooth" tracks have a higher likelihood than "rough" tracks at all finite SNR's.

The continuous outputs of the HMM tracker are comprised of the mean cell occupancy (MCO) track, denoted $I_M(t)$, and the gate occupancy probability (GOP) function, denoted $G_O(t)$. They are defined by

$$G_O(t) = 1 - \gamma_t(0) = \sum_{i=1}^{n} \gamma_t(i) \tag{3.1}$$

and

$$I_M(t) = \sum_{i=1}^{n} \gamma_t(i) \bar{f}_i / G_O(t) \tag{3.2}$$

where the $\gamma_t(i)$ are the state occupancy probabilities given by (2.11) and the $\bar{f}_i$ are the center frequencies of the cells. Note that the sums in (3.1) and (3.2) do not include the term $i = 0$; hence, the MCO track is conditioned on the track at no time occupying the zero state. The MCO track is continuously variable in the frequency range spanned by the gate. The standard deviation $\sigma_M(t)$ associated with the MCO track is defined by

$$\sigma_M^2(t) = \sum_{i=1}^{n} \gamma_t(i) [\bar{f}_i - I_M(t)]^2 / G_O(t). \tag{3.3}$$

The MCO track is undefined whenever the GOP function is identically zero because, in this case, the track occupies the zero state with unit probability, i.e., the track has terminated. In practice, the MCO track should be terminated whenever the GOP function is near zero. This can be done by setting a threshold for $G_O(t)$ or by setting a threshold for the MCO track standard deviation $\sigma_M(t)$. Alternatively, the decision can be based on the state of the Viterbi track at time t. Examples of the Viterbi track, the MCO track, and the GOP function are given in Section IV.

Once a transition into the zero state occurs, the track is terminated. If the track is reinitiated in the gate at some later time, the question arises as to whether or not these two tracks correspond to the same frequency track. The resolution of this question depends on the particular application and on how much additional information is available. It is outside the scope of the present paper.

In the remainder of this section, we discuss in turn the detailed mathematical structure of each of the components of the HMM tracker. The important topic of optimization of the tracker is fully discussed in Section III-D.

A. Definition of the Transition Probability Matrix

In terms of the A matrix, the probability of track initiation into frequency cell j is given by the transition probability a_{0j}, and the probability of track termination out of frequency cell j is given by a_{j0}. For the HMM tracker presented here, it is assumed that the track initiation and termination probabilities, denoted u and v, respectively, are independent of the state j within the gate, i.e., for $j = 1, 2, \cdots, n$, we have

$$a_{0j} = u/n \tag{3.4a}$$

$$a_{j0} = v. \tag{3.4b}$$

If the A matrix is to be a valid transition probability matrix, each of its rows must sum to unity. Thus, from

(3.4a), we have

$$a_{00} = 1 - u. \tag{3.4c}$$

This completes the definition of row and column 0 of the A matrix. The best choice of u and v depends on the particular application and, in principle, these parameters can be determined by training the HMM. Alternatively, in Section III-D, it is shown how to choose u and v to optimize the performance of the HMM tracker without training. In this section, however, u and v are treated as free parameters.

Let the ith cell in the frequency domain be denoted by

$$[f_i, f_{i+1}], \qquad i = 1, 2, \cdots, n$$

where $-\infty < f_1 < f_2 < \cdots < f_{n+1} < +\infty$. The center frequency $\bar{f}_i$ of the ith cell is then given by $\bar{f}_i = (f_i + f_{i+1})/2$. If the frequency track lies in the ith cell at the current time step, the location of the track at the next time step is assumed to be characterized by a Gaussian distribution with mean $\bar{f}_i$ and standard deviation d where d is a measure of the "process" noise. Hence, the probability that the frequency shifts from the ith cell to the jth cell at the next time step is g_{ij} where

$$g_{ij} = (2\pi d)^{-1/2} \int_{f_j}^{f_{j+1}} \exp\left\{-(1/2)\left((f - \bar{f}_i)/d\right)^2\right\} df. \tag{3.5}$$

Note that g_{ij} is not a function of the SNR.

The natural definition of the transition probabilities between the nonzero states of the Markov chain is

$$\bar{a}_{ij} = (1 - v) g_{ij} \Big/ \sum_{k=1}^{n} g_{ik}, \qquad i, j = 1, 2, \cdots, n. \tag{3.6}$$

However, this definition results in an "unbalanced" gate, i.e., the diagonal elements $\bar{a}_{ii}$ are not independent of i for $i > 0$. If the gate is sufficiently unbalanced, then in certain instances, the Viterbi track can be skewed toward the outer cells in the gate. A moment's reflection shows that the problem is caused by the finite gate size and that frequency cells at the edge of the gate have the largest self-transition probabilities. The problem is not significant when the standard deviation d of the process noise is fairly small compared to a cell width, but it grows progressively more severe as d gets larger.

To overcome the unbalanced gate problem, the transition probabilities between the nonzero states of the HMM tracker are derived from the natural probabilities $\bar{a}_{ij}$ and the termination probability v in the following manner. Define

$$a_{\min} = \min_{1 \le i \le n} \bar{a}_{ii}.$$

Now, for $i > 0$, elements $\{a_{i1}, \cdots, a_{in}\}$ of the ith row of the A matrix are obtained from the row vector $\bar{c} = (\bar{a}_{i1}, \cdots, \bar{a}_{in})$ as follows. Replace the "inner" element $\bar{a}_{ii}$ with $a_{\min}$ and normalize the "outer" $n - 1$ elements so that $\bar{c}$ sums to $1 - v$; denote this vector by c_1. If no element of c_1 exceeds $a_{\min}$, then stop. Otherwise, replace the "inner" elements $\bar{a}_{i,i-1}$, $\bar{a}_{ii}$, and $\bar{a}_{i,i+1}$ of $\bar{c}$ with $a_{\min}$ and normalize the "outer" $n - 3$ elements so that $\bar{c}$ sums to $1 - v$; denote this vector by c_2. If no component of c_2 exceeds $a_{\min}$, then stop. Otherwise, continue the algorithm until a vector, say c_s, is found whose components do not exceed $a_{\min}$. The ith row of the A matrix is then defined in partitioned form by (v, c_s). In this algorithm, if indexes $i < 1$ or $i > n$ are encountered, the corresponding element is ignored. Note that the A matrix defined in this fashion is balanced and the transition probabilities a_{ij} are as nearly equal to the natural probabilities $\bar{a}_{ij}$ as possible without unbalancing the gate.

B. Derivation of the Measurement Probability Matrix

The fact that at a given time the signal frequency lies in a prescribed frequency cell does not necessarily mean that a detection occurs in that cell. The presence of random noise can result in some other cell within the gate fortuitously recording greater spectral power than the power in the correct cell. Alternatively, if no cell within the gate records a power greater than the preset threshold D, then a detection is registered in the zero state. The B matrix, therefore, depends on the background noise characteristics, the SNR, and the threshold D. In this section, the SNR and the threshold D are treated as free parameters. How they are chosen to optimize tracker performance is discussed in Section III-D.

It is assumed that the data time series is of the form

$$z(t_0 + kT_s) = A \sin\left[p(t_0 + kT_s) + \xi\right] + n_k \tag{3.7}$$

where t_0 is the initial time and T_s is the sampling period. It is also assumed that the signal amplitude A, phase ξ, and angular frequency p remain constant over the period NT_s, which is the data acquisition time for a Fourier transform of size N. The noise is taken to be zero-mean and Gaussian in nature so that

$$\langle n_k n_j \rangle = \delta_{kj} \sigma^2 \tag{3.8}$$

where the angular brackets denote an ensemble average, δ denotes the Kronecker delta function, and σ^2 is the variance of the noise.

The discrete Fourier transform, denoted $\chi(q)$, at angular frequency q of the time series in (3.7), is given by

$$\chi(q) = (1/N) \sum_{k=0}^{N-1} z(t_0 + kT_s) \exp(-jqkT_s). \tag{3.9}$$

Transforming the complex variable $\chi(q)$ into polar coordinates gives

$$\chi(q) = Re^{j\eta} \tag{3.10a}$$

$$= Ce^{j\phi} + De^{j\theta} \tag{3.10b}$$

where (R, η) denotes the amplitude and phase of $\chi(q)$. The amplitudes and phases of the signal and noise components of χ are denoted by (C, ϕ) and (D, θ), respec-

tively. From (3.7), (3.9), and (3.10), it follows that

$$C = (A/2N)\sin[N(p-q)T_s/2]/\sin[(p-q)T_s/2] \tag{3.11a}$$

and

$$\eta = (N-1)(p-q)T_s/2 + pt_0 + \xi - \pi/2. \tag{3.11b}$$

The pdf of the amplitude R is given by

$$P(R) = (2RN/\sigma^2)I_0(2RCN/\sigma^2) \cdot \exp[-N(R^2+C^2)/\sigma^2] \tag{3.12}$$

where I_0 is the modified Bessel function. Note that $P(R)$ is a noncentral Rayleigh density function.

The Fourier transform in (3.9) is normally calculated only at the discrete values q_i, $i = 1, 2, \cdots, n$ where n is the gate size. For a detection to be registered for a particular observation frequency q_m, two requirements must be met. First, the amplitude R_m of the Fourier transform at frequency q_m must be larger than the amplitudes $R_{i\neq m}$ at all other frequencies $q_{i\neq m}$ within the gate, and second, the amplitude R_m must exceed the prescribed threshold D. If $R_i < D$ for all i within the gate, a detection is registered in the zero state (i.e., $m = 0$).

To simplify the problem, we assume the case where the discrete angular frequencies q_i are given by

$$q_i = 2\pi i/(NT_s) \tag{3.13}$$

and where the true signal angular frequency p lies close to one of the q_i (say, q_m), i.e.,

$$p \cong 2\pi m/(NT_s). \tag{3.14}$$

Substituting (3.13) and (3.14) into (3.11a), we have in this case

$$C = (A/2)\delta_{im}. \tag{3.15}$$

Substitution of (3.15) into (3.12) leads to two separate expressions for the pdf:

$$i = m;\ P_1(R_i) = (2NR_i/\sigma^2)I_0(AR_iN/\sigma^2) \cdot \exp[-N(4R_i^2 + A^2)/(4\sigma^2)] \tag{3.16a}$$

$$i \neq m;\ P_2(R_i) = (2NR_i/\sigma^2)\exp[-NR_i^2/\sigma^2]. \tag{3.16b}$$

Equation (3.16a) describes the pdf of R_i in the case when the true signal lies in the ith frequency cell, and (3.16b) represents the situation when no frequency is present in that cell.

We are now in a position to calculate the various elements B_{m_i} of the B matrix. We consider first the case where there is no signal present in the frequency cells within the gate (i.e., $m = 0$). If the amplitudes measured in all cells lie below the threshold (i.e., $R_i < D$ for all i), then no detection is registered. The probability of this event defines the B-matrix element B_{00}. From (3.16), we have

$$B_{00} = \prod_{i=1}^{n}\int_0^D P_2(r)\,dr = [1 - \exp(-D^2N/\sigma^2)]^n. \tag{3.17}$$

It is possible, however, that the random noise contribution to the time series results in the amplitude in one or more of the cells being larger than the threshold D. This situation corresponds to a false alarm. The probability of a false alarm in the ith cell is given by

$$B_{0i} = (1 - B_{00})/n, \quad i = 1, 2, \cdots, n. \tag{3.18}$$

In the case where a signal is present in one of the cells within the gate (i.e., $m = 1, 2, \cdots, n$), we have three disjoint cases:

1) a detection is registered in the correct cell (i.e., $i = m$);

2) a detection is registered in an incorrect cell (i.e., $i \neq m$, $i \neq 0$);

3) no detection is registered in any cell (i.e., $i = 0$).

The probabilities B_{mm}, $B_{m,i\neq m}$, and B_{m0} corresponding to these three cases are given by the expressions

$$B_{mm} = \int_D^\infty P_1(r)[1 - \exp(-r^2N/\sigma^2)]^{n-1}\,dr \tag{3.19a}$$

$$B_{m0} = [1 - \exp(-D^2N/\sigma^2)]^{n-1}\int_0^D P_1(r)\,dr \tag{3.19b}$$

$$B_{m,i\neq m} = [1 - B_{m0} - B_{mm}]/(n-1). \tag{3.19c}$$

This completes the definition of the B matrix.

C. The Initial State Probability Vector

The final component of the HMM tracker is the initial state probability vector π. The best choice for π depends on the application. For instance, when the entire measurement set Z_T is utilized, as in the examples in Section IV, it is appropriate to choose π to be independent of the measurements. In this case, a good strategy is to force automatic track initiation by starting in the zero state; thus, $\pi_i = A_{0i}$ for $i = 0, \cdots, n$. Alternatively, π can be taken equal to the long-term state occupancy probability vector μ for the A matrix where μ is defined by the matrix equation $\mu A = \mu$. (The vector μ exists uniquely and is nonnegative because the A matrix is positive.) In this paper, however, we utilize the former choice because of its simplicity and because it makes the HMM tracker perform like a kind of detector.

In applications where measurements are taken of an ongoing track, it is reasonable to suppose that the measurement set Z_T is comprised of the most recent T measurements where T is fixed. As indicated in the previous paragraph, for the first data set, the HMM tracker should

use an initial probability vector that is independent of the data. For subsequent data sets, however, π should be updated so that it is dependent on earlier measurements. In effect, the updated π characterizes the impact of track history of the HMM track estimates for the current measurement set. We describe two updating methods that depend on the fact that time $t + 1$ for the previous measurement set is identical to time t for the current set. The simplest update assumes that the Viterbi track for the previous measurement set is correct at time $t = 1$. If $I_V(1)$ denotes the state of the Viterbi track at time $t = 1$, then the π update for use with the current measurement set is taken to be row $I_V(1)$ of the A matrix. Alternatively, the state occupancy probability vector $\gamma_2(i)$ from the previous measurement set at time $t = 2$ can be used as the π update for the current measurement set. This method is computationally less efficient than the former method; however, based on simulated data, it seems to give slightly more accurate estimated MCO tracks.

The HMM tracker is a fixed interval tracker, i.e., for each input measurement sequence Z_T, the output sequence is an estimated track at each time $t = 1, 2, \cdots, T$. When overlapped measurement sets are utilized, as in the preceding paragraph, any time T_E in the output HMM track sequence can be chosen to correspond to the track estimate for the current measurement set. For example, we define the GOP function with lag $T_L = T - T_E$ by

$$G_O(t; T_E) = 1 - \gamma_{T_E}(0).$$

Similar definitions can be made for the MCO track and the Viterbi track. If we choose $T_E = 1$, the HMM tracker functions as a fixed lag tracker with a lag $T_L = T - 1$. If $T_E = T$, the HMM tracker has no lag, i.e., $T_L = 0$. Based on simulated data, it would appear that the variance in the MCO track increases as T_E increases from 1 to T; however, this subject is outside the scope of the present paper and is not discussed further.

D. Optimization of the HMM Tracker

The performance of the HMM tracker is completely determined by the parameters u, v, and d characterizing the A matrix and the parameters D and SNR characterizing the B matrix. However, it is not intuitively obvious how to go about setting reasonable numerical values for all these parameters. We propose the following approach. The process noise parameter d and the SNR parameter are each selected independently of the other parameters in the straightforward manner discussed below. The remaining three parameters, however, are interdependent and are selected by solving three nonlinear equations in three unknowns. One equation is derived from an optimal detection criterion, and the other two equations are derived from optimal tracking criteria. All three optimality criteria are discussed below.

The process noise standard deviation d is similar to the process noise term in a standard Kalman filter. The smaller the value d, the straighter the frequency track is assumed to be; the larger d becomes, the more the frequency track looks like uniformly distributed noise in the gate (even at infinite SNR). For frequency line tracking, an estimate of d can be derived from an estimate of the stability of the line. If such estimates are not available, the best value to use for d can be assessed by trial and error. Examples in Section IV show the effect of different values of d on the output of the HMM tracker.

The HMM tracker is a time-invariant optimal tracker of an intermittent signal that has a specified SNR whenever it is present. We refer to the specified SNR as the tracker SNR. If the true SNR is greater than the tracker SNR, the HMM tracker may interpret genuine frequency changes in the measurement sequence as random noise so that the estimated tracks may be too smooth and may persist after the true signal has terminated. On the other hand, if the true SNR is smaller than the tracker SNR, the HMM tracker may interpret random noise in the measurement sequence as being incompatible with the assumed process noise, and the net result may be premature track termination. Both effects may occur if the true SNR is fluctuating above and below the tracker SNR over the measurement sequence Z_T. For robust HMM tracking, the tracker SNR should be set somewhat smaller than the estimated mean of the true SNR when avoidance of premature termination is critical in the application, and greater than this estimate when belated termination is more important. Alternatively, the tracker SNR can be set by trial and error just as the process noise standard deviation d is often selected. Examples in Section IV show the effects of different tracker SNR's on the HMM tracker outputs. It is seen that the HMM tracker is reasonably insensitive to SNR mismatch.

To define an optimal detection criterion for the HMM tracker and to set the detection threshold, we use the long-term state occupancy probability vector, denoted $\mu = (\mu_0, \mu_1, \cdots, \mu_n)$ of the A matrix. We then show that, for optimal detection, the threshold D is a function of the other parameters defining the HMM tracker. The value of D influences the frequency of occurrence of false alarms and false dismissals in the detection process. We define a false alarm as a measurement (i.e., a detection) in a nonzero state when the zero state is the true state. Thus, the probability of a single false alarm in state $i > 0$ is B_{0i}. Since we have assumed that the A matrix characterizes the frequency track, the long-term total false alarm probability P_{FA} is given by

$$P_{FA} = \mu_0 \sum_{i=1}^{n} B_{0i} = \mu_0(1 - B_{00}) \qquad (3.20)$$

where the last expression in (3.20) follows because the B-matrix rows sum to unity. Similarly, a false dismissal is defined as a measurement in the zero state when a nonzero state is the true state. Thus, if the true state is $i > 0$, the probability of a single false dismissal is B_{i0}, and the long-term total false dismissal probability P_{FD} is given by

$$P_{FD} = \sum_{i=1}^{n} \mu_i B_{i0} = (1 - \mu_0) B_{i0}. \qquad (3.21)$$

The last expression follows from the fact that B_{i0} is independent of the index i for $i > 0$.

Let α and β be nonnegative numbers that add to unity and that represent the relative importance (in the specific application) of false alarms and false dismissals. The error detection criterion is defined by

$$C_{ED} = \alpha P_{FA} + \beta P_{FD}. \tag{3.22}$$

The optimal threshold is therefore that value of D which minimizes C_{ED}; thus, since μ_0 is independent of D, we require D to satisfy

$$\frac{\partial C_{ED}}{\partial D} = -\alpha\mu_0 \frac{\partial B_{00}}{\partial D} + \beta(1 - \mu_0)\frac{\partial B_{i0}}{\partial D} = 0. \tag{3.23}$$

For the A matrices used in this paper, it is easy to show, using (3.4), that

$$\mu_0 = v/(u + v). \tag{3.24}$$

Substituting (3.24) into (3.23) and then differentiating (3.17) and (3.19b), it follows that the optimal threshold satisfies the nonlinear equation

$$P_1(D) = \frac{2nND}{\sigma^2}\frac{\alpha v}{\beta u}\left[1 - \frac{n-1}{n}\frac{\beta u}{\alpha v}\frac{B_{i0}}{B_{ii}}\right] \cdot \exp\left(-D^2N/\sigma^2\right) \tag{3.25}$$

where $P_1(D)$ is given by (3.16a). Equation (3.25) can be solved by a variety of simple iteration procedures, e.g., the bisection method. Obvious modifications are required in (3.25) if either $\alpha v = 0$ or $\beta u = 0$.

An interesting special case of (3.25) occurs for $\alpha = u/(u + v)$ and $\beta = v/(u + v)$. The optimal threshold is then independent of u and v and dependent only on the tracker SNR. For these values of α and β, the criterion C_{ED} is physically meaningful because it emphasizes false alarms when the frequency track is unlikely to be in the zero state (i.e., μ_0 is small) and emphasizes false dismissals when the track is unlikely to be in the gate (i.e., μ_0 is large). This choice of α and β is not necessarily the best choice for the specific application, however, and we do not pursue the matter further here.

As the tracker SNR goes to infinity, the ratio B_{i0}/B_{ii} goes to zero. Neglecting this ratio in (3.25) gives the simpler approximate expression

$$P_1(D) = (2nND/\sigma^2)(\alpha v/\beta u)\exp\left[-D^2N/\sigma^2\right]. \tag{3.26}$$

This expression must be solved iteratively for D, but it does not require the evaluation of integrals as does (3.25). The threshold satisfying (3.26) is independent of the tracker SNR, and dependent on u and v.

To obtain a threshold from (3.25), we must first specify u and v. We use the GOP function to define optimal tracker initiation and termination criteria. We then show that, for optimal tracking, the parameters u and v are functions of the other parameters defining the HMM tracker. An exemplar tracker initiation measurement sequence is defined by

$$z_t = [(n+1)/2], \quad \text{for } t = [T/2] - [L_B/2] + 1, \cdots, [T/2] - [L_B/2] + L_B$$
$$z_t = 0, \quad \text{otherwise} \tag{3.27}$$

where $[x]$ denotes the greatest integer less than or equal to x. This exemplar sequence is comprised of midgate measurements of duration L_B in the center of a string of zero state measurements. Let $G_{OB}(t)$ denote the GOP function corresponding to (3.27), with the HMM tracker started in the zero state. Intuitively, $G_{OB}(t) = 0$ at time $t = 1$, rises monotonically to some maximum value at time $t = T/2$, and thereafter decreases monotonically to time $t = T$. The tracker initiation criterion is defined by

$$C_{OB} = \max_{1 \le t \le T} G_{OB}(t) \tag{3.28}$$

and its optimal value is defined by $C_{OB} = 1/2$. With this optimality criterion, the HMM tracker gives a 50% probability that the exemplar sequence (3.27) is identified as a track by the GOP. In other words, sequences of measurements in the gate of duration less than L_B are treated by the tracker as likely false alarms, and sequences of duration greater than L_B are treated as likely new tracks.

Similarly, an exemplar tracker termination measurement sequence is defined by

$$z_t = 0, \quad \text{for } t = [T/2] - [L_E/2] + 1, \cdots, [T/2] - [L_E/2] + L_E$$
$$z_t = [(n+1)/2], \quad \text{otherwise.} \tag{3.29}$$

This exemplar sequence experiences a "drop out" of duration L_E in the center of a midgate measurement sequence. Let $G_{OE}(t)$ denote the GOP function corresponding to (3.29), with the HMM tracker started in state $[(n+1)/2]$. Intuitively, $G_{OE}(t) = 1$ at time $t = 0$, falls monotonically to some minimum value at time $t = T/2$, and thereafter increases monotonically to time $t = T$. The tracker termination criterion is defined by

$$C_{OE} = \min_{1 \le t \le T} G_{OE}(t) \tag{3.30}$$

and its optimal value is defined by $C_{OE} = 1/2$. HMM trackers satisfying this optimality criterion give a 50% probability that the exemplar sequence (3.29) is not considered to be a track by the GOP. Thus, sequences of zero state measurements of duration less than L_E are treated by the tracker as likely false dismissals, and sequences of duration greater than L_E are treated as likely terminated tracks.

Optimal values of u and v are determined by first selecting durations L_B and L_E that are suitable for the spe-

cific application, and then solving the three nonlinear equations (3.25), $C_{OB} = 1/2$, and $C_{OE} = 1/2$ simultaneously for D, u, and v. This set is equivalent to a system in only u and v because D is given as a function of u and v by (3.25) and because d and the tracker SNR are already specified. C_{OB} and C_{OE} are readily computed for any given pair of (u, v) values using the HMM tracker. Consequently, straightforward iteration procedures can be used to find optimal values for u and v. In practice, we proceed by using the HMM tracker to compute the right-hand sides of (3.28) and (3.30) for a small grid of (u, v) pairs after first computing D using (3.25) at each gridpoint. The grid is adjusted by inspection until near optimal values of C_{OB} and C_{OE} are found.

IV. Application of HMM Frequency Tracker to Simulated Data

The performance of the HMM frequency tracker is evaluated by application to simulated data. The advantage of simulated data over real data is that the underlying, "hidden" state sequence in this case is precisely known, thus enabling the objective assessment of the tracker performance.

For the purposes of evaluation, two sets of simulated data are generated. Each set consists of a frequency-modulated sine wave added to white Gaussian noise. The frequency excursions of the modulation spans five of the frequency cells employed in the Markov model. In the examples considered here, the gate size is set to nine, and the gate is centered about the mean signal frequency. The total number of states in the Markov chain is thus ten, counting the zero state.

The frequency modulation characteristics for the two data sets are displayed in Fig. 1. The y axis is divided into the nine discrete frequency cells employed in the HMM, and the x axis is divided into 100 time steps. The true signal frequency as a function of time is indicated by the cell occupancy track. For Fig. 1(a), the signal is present throughout the total time period, while in Fig. 1(b), it is present only intermittently.

Intensity-modulated "spectrograms" for the two data sets are presented in Fig. 2. The signals of Fig. 1 are added to white noise and the power spectra of the resultant sums are then calculated. The power spectra are shown as a function of time in the spectrograms of Fig. 2. The SNR values in Fig. 2(a) and (b) are -23 and -20 dB, respectively. The underlying, hidden state sequence is difficult to discern in Fig. 2.

The various outputs of the HMM tracker for the continuous signal frequency are displayed, together with the measurement sequence, in Fig. 3. The measurement sequence, which indicates the cell containing the maximum spectral power if the power exceeds a prescribed threshold or the zero state if the power in no cell exceeds the threshold, is shown in Fig. 3(c). The zero state is shown slightly displaced from the other frequency cells, and in this example remains unoccupied at all times. The measurement sequence forms the only data input to the HMM tracker.

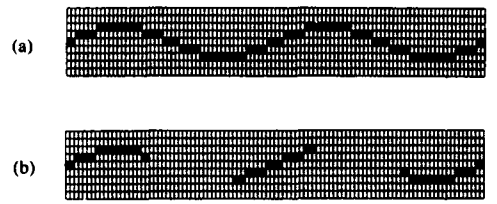


Fig. 1. The continuous (a) and intermittent (b) true signals used for the investigation of the HMM tracker. The frequency cells are marked along the y axis and the time divisions along the x axis.

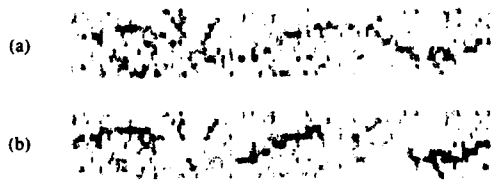


Fig. 2. Intensity-modulated "spectrograms" for the signals of Fig. 1. The signals are embedded in white Gaussian noise. SNR values of -23.0 and -20.0 dB are used for (a) and (b), respectively.

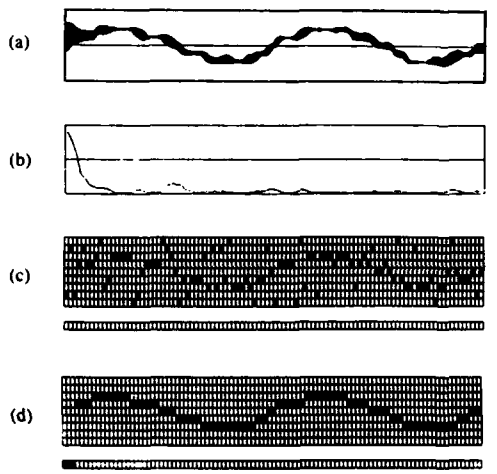


Fig. 3. Results of applying the HMM tracker to the data of Fig. 2(a). Parameters employed are: tracker SNR = -23 dB, $d = 0.333$, $u = 0.24$, $v = 0.016$, $D = 0.0278$. (a) The output of the MCO tracker; the shaded area contains all the tracks within one standard deviation of the mean track. (b) The probability of zero state occupancy [i.e., $\gamma_t(0) = 1 - G_O(t)$] as a function of time. (c) The measurement sequence; the cells corresponding to the zero state are shown underneath the cells of the gate. (d) The Viterbi track; the cells corresponding to the zero state are shown underneath the cells of the gate.

The MCO track is shown in Fig. 3(a). The shaded area marks the bounds lying one standard deviation estimate $\sigma_M(t)$ on either side of the optimal track. The MCO track does not directly estimate the possibilities of track initiation and termination. However, track initiation and termination can be included by defining a threshold on the MCO track standard deviation or by calculating the GOP function and setting an appropriate threshold there. The probability of zero state occupancy, i.e., $1 - G_O(t)$, is shown in Fig. 3(b). This probability starts high because the HMM tracker is initialized in the zero state.

The Viterbi track for this data set is displayed in Fig.

3(d), and is seen to provide an excellent reconstruction of the true state sequence of Fig. 1(a). The Viterbi track does not initiate until three time steps after the start of the data sequence. The delay is explained by the initiation of the HMM tracker in the zero state and by the large fluctuations in the measurement sequence at the start of the sequence. This delay is consistent with the high probability of zero state occupancy in Fig. 3(b) and the large variance in the MCO track over the corresponding period. The parameters used in the HMM tracker for the continuous signal frequency are listed in the caption of Fig. 3.

The results from the application of the HMM tracker for the intermittent signal frequency [Fig. 1(b)] are displayed in Fig. 4, and Fig. 4(a)–(d) correspond to the analogous results shown in Fig. 3(a)–(d) for the continuous signal. For the intermittent signal, the Viterbi track is seen once again to provide an excellent reconstruction of the true state sequence of Fig. 1(b). In this case, the capability of the Viterbi track to terminate and initiate as the signal drops out and reappears is clearly demonstrated. From Fig. 4(a), it is seen that the MCO track follows the true state sequence closely during the periods when the signal is present, and exhibits a large variance when the signal is absent. Similarly, the probability of zero state occupancy is also large when the signal is absent. The setting of a threshold on the MCO track standard deviation or on the probability of zero state occupancy could be used to implement track termination and initiation for the MCO track. The results would then agree closely with those obtained from the Viterbi track. The HMM parameters used for these examples are listed in the caption to Fig. 4.

As we have indicated earlier, for optimal performance of the HMM tracker, the parameters of the HMM should represent as closely as possible the characteristics of the line being tracked. The process noise parameter d is a measure of the likelihood of the track changing frequency. In Fig. 5, the Viterbi track for the continuous signal is presented as the value of the process noise parameter d is decreased from 0.667 to 0.167. For the highest value of d, the track tends to follow the measurement sequence too closely, with the result that the Viterbi track exhibits a fine structure that is not present in the true state sequence. On the other hand, when d is small, the tracker fits the measurement sequence by a series of straight line segments, separated by track terminations and reinitiations. In this case, the tracker finds it less "costly" to terminate and reinitiate the track than to allow the track to step to an adjacent frequency cell. We refer to this phenomenon of termination and reinitiation as "punctuation." The value of 0.333 for d seems to provide the optimal track for this data set.

In Fig. 6, the effect of varying the tracker SNR is investigated for the intermittent signal of Fig. 1(b). In this case, the SNR of the true signal is −20 dB, and the Viterbi track exhibits optimal results when the tracker SNR is equal to the true SNR. If the tracker SNR is less than the true SNR [e.g., Fig. 6(a)], the tracker is less likely to terminate, even when the true signal is absent. If the tracker SNR is considerably larger than the true SNR, the

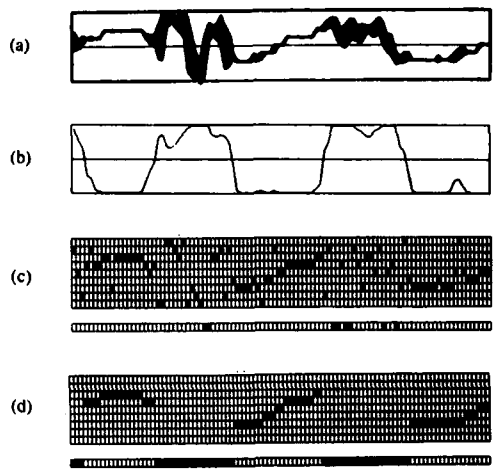


Fig. 4. As for Fig. 3, but for the data of Fig. 2(b). Parameters employed are: tracker SNR = −20 dB, d = 0.333, u = 0.3, v = 0.0092, D = 0.0386.

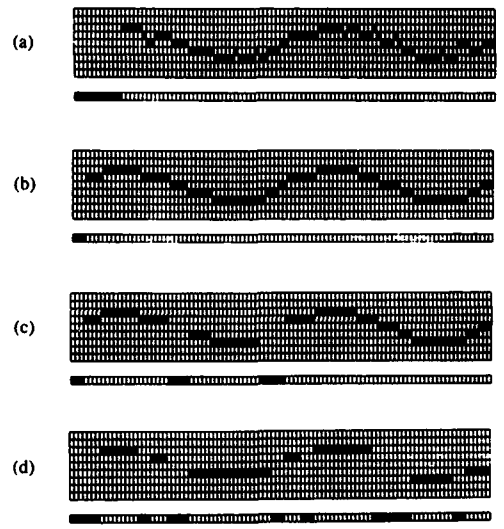


Fig. 5. The Viterbi track for the data of Fig. 2(a), showing the effects of varying the process noise d. Values of d are: (a) 0.667, (b) 0.333, (c) 0.222, (d) 0.167. Other parameters are the same as in Fig. 3.

tracker is likely to terminate, even when the signal is still present (as the HMM was designed for a stronger signal), and is more likely to follow the measurement sequence too closely (as the HMM attaches an overimportance to each measurement). These two modes of behavior can be seen in Fig. 6(d).

The dependence of the probability of zero state occupancy on the tracker SNR exhibits a similar range of behavior to that of the Viterbi track. In Fig. 7(a)–(d), the probabilities of zero state occupancy are shown for the same range of HMM parameters as was used in Fig. 6(a)–(d). For a tracker SNR of −25.2 dB, the probability of zero state occupancy is always less than 0.5 (except near the start of the data sequence) and the Viterbi track terminates only once. This termination is a punctuation caused by the necessity of reducing the cost of making a

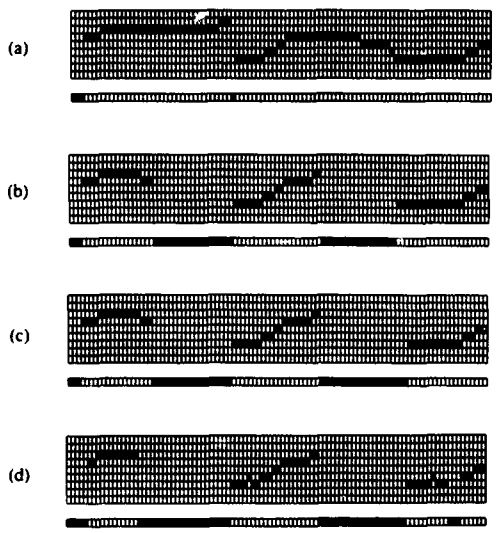


Fig. 6. The Viterbi tracks for the data of Fig. 2(b), showing the effects of varying the tracker SNR. Values of the tracker SNR are: (a) −25.2 dB, (b) −23.0 dB, (c) −20.0 dB, (d) −17.0 dB.

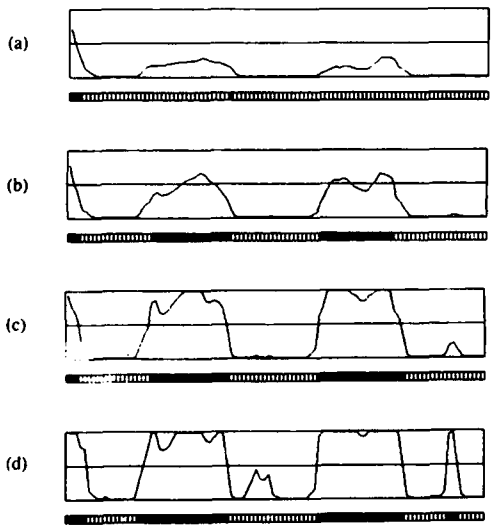


Fig. 7. The probability of zero state occupancy for the data of Fig. 2(b), showing the effects of varying the tracker SNR. The parameter values are the same as in Fig. 6. The zero state occupancies associated with the Viterbi tracker are also shown for comparison.

spurious five-cell frequency change. As the tracker SNR increases, so does the probability of zero state occupancy, until in the region where the tracker SNR is considerably larger than the true SNR, this probability becomes excessively high, and spurious terminations occur in the Viterbi track.

From the data shown in Figs. 1-7, it is apparent that both the Viterbi and MCO tracks provide very good reconstructions of the hidden, true signal behavior. The two tracks are mutually consistent. Track termination and initiation capabilities are intrinsic to the Viterbi track and can be built into the MCO track. Figs. 6-7 indicate that optimal performance of the trackers is dependent on the tracker parameters being suitably matched to the signals under investigation. However, it has been our experience that even in a case where some mismatch of parameters is unavoidable (e.g., the tracking of real signals), the two tracks still provide remarkably good reconstructions of the underlying signal, and agree consistently with what would be obtained from careful inspection of the original spectrograms.

V. Comparisons to Related Trackers

A. Formant Tracking

Kopec [7] studies the problem of tracking formants in speech using HMM's. Formant tracking is similar to frequency line tracking, and Kopec's paper and this paper have much in common. Both papers use finite-state, finite-outcome HMM's, and both use the same kinds of states. Moreover, Kopec uses a "distinguished nonnumerical state" to represent the absence of a formant, just as we have used the zero state to indicate the absence of a track in the gate. The different applications, however, require different definitions of the measurement sequence. Kopec uses the codebook vectors that result from vector quantization, whereas we use the frequency state estimates of a threshold detector.

A significant difference between Kopec's paper and our paper is that he uses the MCO track, but not the Viterbi track. One reason he rejects the Viterbi track is that his frequency cells are from 50 to 100 Hz wide, and so any discrete track estimate is inherently unacceptable. In our application, however, the frequency cells can be made as small as desired by increasing the FFT resolution. Another reason he does not use the Viterbi track is that it does not provide a way to define or set a formant detection threshold. In our application, we are able to define the HMM tracker so that there is excellent correlation between zero state occupancy for the Viterbi track and low values of the GOP function; therefore, there is no necessity to set a threshold on the GOP to determine the presence or absence of the track. Nonetheless, in practice, we would still set such a threshold for the GOP.

B. Relationship with a Dynamic Programming Tracking Method

In a set of earlier papers (e.g., see [8]), Scharf *et al.* present a dynamic programming method for tracking frequency and phase. Although they do not identify it as such, their algorithm is equivalent to an HMM using real-valued continuous measurement vectors. They assume the frequency is constant for the duration of each block of time series data, and then allow a transition to another of a set of discrete frequencies. These discrete frequencies correspond to the states of the underlying HMM. They do not, however, include a zero state to indicate the absence of a signal. They use the Viterbi track exlcusively and do not discuss the MCO track.

The fundamental equation of Scharf *et al.* is the loga-

rithmic likelihood function, denoted LLF, which is given by

$$\text{LLF} = -\sum_{t=1}^{T} (1/2\sigma_t^2)\left|z_t - s_t\right|^2 + \sum_{t=1}^{T} \ln p(x_t|x_{t-1}). \tag{5.1}$$

Here, $\{x_1, \cdots, x_T\}$ denotes a sequence of discrete frequency states, the vector z_t is a block of time series data commencing at time t, the vector s_t is the complex exponential signal vector corresponding to a frequency x_t, and σ_t is the standard deviation of the random background noise at time t. The measurements in the HMM of Scharf *et al.* are the observed time series data blocks; thus, the first term in the LLF is equivalent to the B matrix in the HMM, and the second term in the LLF is equivalent to the A matrix of the HMM. The dynamic programming algorithm that Scharf *et al.* present for maximizing (5.1) is equivalent to the Viterbi algorithm (2.5)-(2.6).

C. Bayes-Markov Tracking

Jaffer *et al.* [9] present a recursive Bayesian technique for tracking dynamic signals in noise. Although they do not present it as an HMM, their technique is equivalent to a finite-state HMM using real-valued continuous measurement vectors. It uses a one time-step recursion to update the posterior pdf of the state conditioned on the measured data, but it does not treat sequences of measurements collectively. The MCO track, if used with zero lag, is similar to the tracker of Jaffer *et al.*

Jaffer *et al.* define the states of their model to be FFT resolution cells just as we have done, but they do not define a zero state to denote the absence of a signal. A measurement is a (real) vector whose components are the magnitudes of the output FFT's in the resolution cells. Let $P[X_t = j|Z_t]$ denote the posterior pdf of the signal location in cell j conditioned on the entire measurement sequence Z_t. The fundamental equation of their method (neglecting scale factors) is

$$P[X_t = j|Z_t] = b(z_t|X_t = j)\sum_{i=1}^{n} a_{ij}P[X_{t-1} = i|Z_{t-1}] \tag{5.2}$$

where $b(z_t|X_t = j)$ is the pdf of the current measurement vector z_t conditioned on state j. Using Bayes' theorem, they show that

$$b(z_t|X_t = j) = f_{S+N}(z_t(j))/f_N(z_t(j))$$

where $f_{S+N}(z_t(j))$ and $f_N(z_t(j))$ are the signal-plus-noise and noise only pdf's, respectively, of the data in cell j. They define the initial state probability vector to be uniform. Bayes' theorem can be used to show that (5.2) is identical to one step of the HMM forward algorithm (see (2.10a)).

Jaffer *et al.* do not explicitly define the A matrix, but the kind of matrix they have in mind is clear from the context and from the two interesting examples they present. (One example is pulsed radar tracking of range and Doppler, and the other is passive sonar tracking of Doppler and delay.) They also give a detection statistic that they claim enhances signal detection despite target motion, but do not present examples of its use or discuss the false alarm rates that might be anticipated.

VI. Possible Extensions of the HMM Tracker

It is seen in Section IV that the HMM tracker is an excellent algorithm for frequency line tracking, provided that the underlying HMM is optimized for the line under study. A number of extensions to the present work that could enhance the performance of the HMM tracker even further are now discussed.

In the application of HMM's to speech processing, the training of HMM's is a well-established concept. Training is mentioned briefly in Section II, but it is unnecessary for the analyses carried out here for simulated data. With real data, however, training the HMM will enable the determination of the optimal values of the A matrix, B matrix, and π for the line being considered. These parameters are likely to depend on the SNR, and on the nature and amplitude of the frequency modulation of the line. Suitable training of the HMM should result in better frequency tracking for real data.

Extending the present tracker to include the possibility of more than one line being present in the frequency gate would enable the tracking of lines whose frequencies are close together, and would include the possibility of frequency tracks crossing. One implementation of such an extension is to allow multiple detections when the spectral power in more than one frequency cell lies above the detection threshold D. Each state is then no longer uniquely associated with a frequency cell or the zero state, but describes one of the following possibilities: 1) no detection in any frequency cell, 2) a detection in only one frequency cell, or 3) detections in two (or more) frequency cells. The A and B matrices would have to be reformulated in a manner consistent with this interpretation.

The concept of an HMM state can also be extended to incorporate both the frequency and its time derivative. The advantage of such an extension is that the dynamic characteristics of the line tracks can be more completely represented by the A matrix, with a consequent improvement in the tracker performance; the disadvantage is that the number of states is increased by a factor equal to the number of time derivative resolution cells, with a consequent increase in the required computing time. The proposed extension would enable a more meaningful comparison of the HMM tracker to existing alpha-beta and Kalman trackers (see, e.g., [11]-[13]) which typically use a track derivative model.

The examples presented in Section IV are investigated using a finite-time window of length $T = 100$. The possibility of sliding windows is discussed in Section III-C. For frequency tracks that change substantially in frequency over a long time, it would clearly be computationally advantageous to employ smaller windows and an "adaptive" gate whose center frequency and width

change as the window slides over the data sequence. The use of an adaptive gate, combined with the treatment of multiple tracks within a gate, would significantly extend the range of applicability of the HMM tracker.

Finally, it is emphasized that the data input to the current HMM tracker is in the form of a measurement sequence obtained from a simple threshold detector. The tracker is therefore denied access to important frequency estimation information, e.g., the amplitude and phase of the complex FFT's associated with each frequency cell. Extensions that allow for measurement sequences of a more sophisticated nature than the output of a simple threshold detector clearly warrant further investigation. A possible alternative measurement sequence for input to the tracker could be the instantaneous frequencies obtained from a Wigner–Ville time-frequency analysis of the data.

VII. Conclusions

In this paper, the application of HMM's to the problem of frequency tracking is presented and discussed, and the interpretation of several earlier tracking algorithms in terms of HMM's is pointed out for the first time. It is demonstrated in Section III how to formulate the frequency tracking problem in terms of HMM's and how to optimize the HMM parameters for this problem. In Section IV, the HMM tracker is tested by application to simulated data. The resultant tracks are comparable to those obtained by inspection of the spectrograms, and are found to be robust with respect to variations in the HMM parameters.

Tracker optimization is important and is responsible for the observed consistency between the Viterbi and the MCO tracks, and for the reliable initiation and termination of the tracks. We propose three different track initiation and termination criteria based on the Viterbi track, the standard deviation of the MCO track, and the GOP function. The mutual consistency of the results obtained by using these criteria is a testimony to the successful optimization of the HMM parameters.

The present work is capable of extension in several directions. A number of possible extensions are outlined in Section VI. These issues are the subject of current investigation, and will be addressed in subsequent publications.

References

[1] D. C. Rife and R. R. Boorstyn, "Single-tone parameter estimation from discrete-time observations," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 591-598, Sept. 1974.

[2] S. M. Kay and S. L. Marple, Jr., "Spectrum analysis—A modern perspective," *Proc. IEEE*, vol. 69, pp. 1380-1419, Nov. 1981.

[3] D. R. A. McMahon and R. F. Barrett, "An efficient method for estimation of the frequency of a single tone in noise from the phases of discrete Fourier transforms," *Signal Processing*, vol. 11, pp. 169-177, Sept. 1986.

[4] R. F. Barrett and D. R. A. McMahon, "Comparison of frequency estimators for underwater acoustic data," *J. Acoust. Soc. Amer.*, vol. 79, pp. 1461-1471, May 1986.

[5] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, vol. 3, pp. 4-16, Jan. 1986.

[6] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst. Tech. J.*, vol. 62, pp. 1035-1074, Apr. 1983.

[7] G. E. Kopec, "Formant tracking using hidden Markov models and vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 709-729, Aug. 1986.

[8] L. L. Scharf and H. Elliott, "Aspects of dynamic programming in signal and image processing," *IEEE Trans. Automat. Contr.*, vol. AC-26, pp. 1018-1029, Oct. 1981.

[9] A. G. Jaffer, R. L. Stoutenborough, and W. B. Green, "Improved detection and tracking of dynamic signals by Bayes-Markov techniques," in *Proc. ICASSP'83*, vol. 2, pp. 575-578.

[10] L. E. Baum, T. Petrie, G. Soules, and N. Wiess, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Statist.*, vol. 41, no. 1, pp. 164-171, 1970.

[11] T. R. Benedict and G. W. Bordner, "Synthesis of an optimal set of radar track-while-scan smoothing equations," *IEEE Trans. Automat. Contr.*, vol. AC-7, pp. 27-32, July 1962.

[12] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. New York: Academic, 1988.

[13] R. F. Barrett, A. K. Steele, and R. L. Streit, "Frequency line tracking algorithms," in *Proc. NATO Adv. Study Inst. Underwater Acoustic Data Processing*, Kingston, Ont., Canada, July 1988.

Roy L. Streit (SM'84) was born in Guthrie, OK, on October 14, 1947. He received the B.A. degree (with Honors) in mathematics and physics from East Texas State University, Commerce, in 1968, the M.A. degree in mathematics from the University of Missouri, Columbia, in 1970, and the Ph.D. degree in mathematics from the University of Rhode Island, Kingston, in 1978.

He was a Visiting Scholar in the Department of Operations Research, Stanford University, Stanford, CA, during 1981-1982, and an Exchange Scientist in the Signal Processing and Classification Group at the Defence Science and Technology Organisation, Adelaide, South Australia, from 1987 to 1989. He joined the staff of the Naval Underwater Systems Center (then the Navy Underwater Sound Laboratory), New London, CT, in 1970. He is an Applied Mathematician and has published work in several areas, including towed array design, complex function approximation, semi-infinite programming, and applications of hidden Markov models. His current interests include image analysis, tracking problems, and training algorithms for neural networks.

Ross F. Barrett was born in Fremantle, Western Australia, on August 19, 1942. He received the B.Sc. (Honours) degree in physics in 1964 and the Ph.D. degree in physics in 1969, both from the University of Western Australia.

He was an Alexander Von Humboldt Fellow at the University of Frankfurt, West Germany (1969-1972), a Research Fellow at the University of Melbourne (1972-1974), a Research Fellow at the Australian National University, Canberra (1974-1978), and a Lecturer in Physics at the University of Western Australia (1978-1982). Until this period, his primary research interests were in theoretical and experimental nuclear physics. In 1982 he joined the staff of the Defence Science and Technology Organisation, Salisbury, South Australia, where he is presently a Principal Research Scientist. His current research interests are in signal processing techniques and passive sonar classification.

Nonlinear Frequency Line Tracking Algorithms

A. K. Steele, R. L. Streit and R. F. Barrett

Nonlinear Frequency Line Tracking Algorithms

A.K. STEELE
Maritime Systems Div., Weapons Systems Research Laboratory, Salisbury, S.A.

R.L. STREIT
Naval Underwater Systems Center, New London, Connecticut, U.S.A.

R.F. BARRETT
Maritime Systems Div., Weapons Systems Research Laboratory, Salisbury, S.A.

1 INTRODUCTION

The problem of producing accurate target tracks from noisy measurements is of great current interest for the automation of signal processing systems. Trackers are classified as either linear or nonlinear systems. Examples of linear trackers are the alpha-beta and Kalman trackers. Examples of nonlinear trackers are the probabilistic data association (PDA) and hidden Markov model (HMM) trackers. This paper deals exclusively with these two nonlinear trackers.

Linear trackers estimate the track using simple deterministic or statistical dynamic target motion models to develop filters for the target position estimates. The problem with linear trackers is that they are sensitive to outliers and false measurements. The nonlinear PDA methodology can be applied to (single input) linear trackers to overcome the problems caused by outliers and multiple input measurements. Application of the PDA methodology to the Kalman tracker results in the PDA-Kalman tracker studied in this paper.

The HMM tracker, which is a recent development, models the target measurement sequence as a probabilistic function of a Markov chain. The states of the Markov chain define the target states, and the transition probability matrix of the Markov chain defines possible target motion. The probabilistic function describes exactly the non-Gaussian nature of the measurement process. The HMM tracker provides a unified mathematical framework for describing important tracking problem issues. In particular, the HMM tracker initiates and terminates tracks automatically as an intrinsic feature of the tracking algorithm. It does this by incorporating a special state into the Markov chain to designate the absence of a target. The HMM tracking algorithm is equivalent to a sequence of matrix-vector products.

A new development is the HMM/A tracker which is a HMM tracker that also uses the amplitude of the input measurements as additional information. The inclusion of amplitude information does not significantly increase the complexity of the HMM/A tracker over that of the HMM tracker. The HMM/A tracker is the only tracker in this paper that uses amplitude information. (For a discussion of the performance of the HMM/A tracker as a signal detector see Barrett & Streit, 1989.)

This paper compares the PDA, HMM, and HMM/A tracking algorithms when used for frequency line tracking on two different simulated data sets. These examples (and others not given here) show that quantitative comparisons of the frequency line tracking algorithms (FLTA) require careful statistical analysis. At high SNRs adequate comparisons can be made using simple error measures (e.g. rms tracking error) on a few data sets, but at low SNRs, such measures can be misleading and one must resort to ensemble statistical measures of tracker performance. Such statistical performance comparisons are especially important when discussing track initiation and track termination.

2 DEFINITIONS

In this paper, we treat the problem of tracking the time variation of the instantaneous frequency of an isolated tone embedded in additive white noise as a post-detection process applied to the evolving short term Fourier spectra of the sampled time series. Separating the tracking problem from the detection problem can lead to suboptimal tracking performance, but it is an approach commonly used in practice.

The choice of detector is important because the output of the detector determines the tracker input measurements. Throughout this paper we use a simple threshold detector because of its widespread usage and ease of implementation. No interpolation is employed to smooth the intrinsic quantization effects of this detector because interpolation is not justified at low SNR. Other well known problems associated with the threshold detector are outliers (false detections) and missed detections. These problems cause serious tracking errors in linear trackers; however, as the examples will show, our PDA, HMM and HMM/A trackers are robust against outliers and missed detections and are capable of tracking down to the input quantization level.

All the trackers considered here can, in principle, accept multiple measurements as input; i.e., the tracker input is the set (possibly empty) of centre frequencies of all FFT cells whose amplitude exceeds the detection threshold; however, if the detection threshold is not exceeded, no frequency measurement is made for the current scan, or block of time series data. On the other hand, the tracker input can be a single measurement, i.e. the FFT cell having the largest amplitude if the amplitude exceeds the threshold.

The PDA trackers considered here use either single or multiple measurements and ignore the amplitude of the measurements. The only HMM tracker considered here accepts just single measurements, while the only HMM/A tracker uses single measurements together with their associated amplitudes.

3 PROBABILISTIC DATA ASSOCIATION TRACKING

We present PDA as a method for converting a single-

input-single-output (SISO) tracker into a multiple-input-single-output tracker. PDA assumes that only one of the multiple input measurements corresponds to the target being tracked. It further assumes that measurements in the next scan will be normally distributed with the mean and covariance predicted from the current scan. PDA is thus applicable to any SISO tracker in which measurement mean and covariance at the next scan are predicted.

Track input measurements are gated, thus creating the possibility of false dismissal of the target measurement. Using the PDA assumptions, the probability of false dismissal is easily evaluated. The usual gated PDA method uses a variable gate to achieve constant probability of false dismissal. For the FLTA, we use a fixed gate so the false dismissal probability varies from scan to scan.

Let a_i be the probability that the i-th measurement corresponds to the target, and let a_0 be the probability that none of the measurements corresponds to the target. Similarly, let f_i be the track frequency estimate generated by using the i-th measurement in the underlying SISO tracker, and let f_0 be the predicted track frequency when no measurement is made. Then the PDA tracker output is given by

$$f_{PDA} = a_0 f_0 + \sum_{i=1}^{N} a_i f_i \quad , \qquad (1)$$

where N is the total number of detections in the gate. For the PDA Kalman tracker, the error covariance associated with f_{PDA} is also computed (Bar-Shalom & Fortmann, 1988), and it is used with f_{PDA} to make the measurement mean and covariance predictions in the underlying SISO Kalman filter. The probabilities (a_i) are easily computed and require only the evaluation of a truncated Gaussian density function. The nonlinear dependence of all PDA trackers on the measured track input data is due to the nonlinear dependence of the probabilities (a_i) on the data.

4 HIDDEN MARKOV MODEL TRACKING

HMM's are probabilistic models that are commonly used in speech applications. Their utility in tracking applications seems not to be recognised in the general literature, except for a paper by Kopec (1986) who uses them to track formants, or resonances, in spoken words. The HMM tracker presented in this paper is similar to Kopec's formant tracker; however, the FLTA application permits the analytical development of the parameters defining the underlying HMM.

The HMM tracker is a fixed interval FLTA; i.e., it takes a fixed length sequence, or "window", of measurements and outputs a track estimate for each time in the window. By sliding the window along as new data are collected, the track estimate evolves in time. Alternatively, one may simply increase the window size. Either way, the HMM tracker is used only to compute the output track estimate for each given tracking window. The quantised frequency track is modelled as a finite state Markov chain. A "faded" or "zero" state represents a track whose SNR is less than the tracker SNR (see below); the remaining "active" states represent a track occupying an FFT cell inside a fixed gate and having an SNR greater than the tracker SNR. Track initiation is defined as a transition from the zero state to any active state, while track termination is defined as a transition from any active state into the zero state. Initiation and termination of tracks, as well as movement of the track within the gate, are therefore governed by the transition probability matrix, A, of the Markov chain.

Measurements of the frequency track are characterised by a detection probability matrix, B. Thus, for each possible target state, the threshold detector outputs are measured target states with probabilities that are computed analytically from the SNR and the threshold. The SNR assumed for this B-matrix calculation is called the tracker SNR; in effect, it is the lowest SNR at which tracks are initiated and estimated.

The HMM tracker is fully specified by the A- and B-matrices, together with the initial state probability vector π. Initially, π corresponds to a target in the zero state. This forces automatic track initiation. If the tracking window slides along as new data are collected, π is updated using current HMM tracker output.

The HMM/A tracker cannot utilize a B-matrix because amplitude is a continuous quantity. Instead, it is necessary to compute the likelihoods of the measurements, conditioned on each possible target state. There are a finite number of states, so these conditional likelihoods can be stored as a matrix.

The HMM and HMM/A trackers output both a discrete (quantised) track and a continuous track. The quantised track is the Viterbi track; i.e., of all possible tracks (realisations of the Markov chain), the Viterbi track is the one most likely to account for the measurement sequence. The continuous track is essentially the expected track, with the expectation taken over all possible realisations of the Markov chain. Strictly speaking, the expected track is conditioned on the track not occupying the zero state, as well as on the measurements. The total probability of the zero state track (conditioned only on the measurements) at each point in the window is thus a necessary complement to the continuous track estimate. The gate occupancy probability (GOP) is defined as one minus the probability of the zero state track, and it is the GOP that is plotted in the examples in the next section.

The discrete output of the HMM tracker is computed using only n^2T additions, while the continuous output uses n^2T multiplications, where n is the number of Markov chain states and T is the number of scans in the window. The discrete and continuous algorithms are easily vectorised. Similar remarks hold for the HMM/A tracker once the necessary conditional likelihoods have been computed.

5. EXAMPLES

The simulated data were obtained by generating a sine wave triangularly swept in frequency with a period of 132 scans, a centre frequency of 5, and a maximum deviation of 3. Uncorrelated noise is then added to the sine wave, Fourier transformed, and then passed to a threshold detector. One hundred scans of the output of the threshold detector are then used as the input for the various trackers. The signal is absent until scan 15 when the SNR is increased instantaneously to 3 dB (in an FFT cell). The SNR is kept at that level until scan 79, when the signal ceases. All the trackers use a gate width of 9 FFT cells.

Figure 1 illustrates the cell occupancy of the true track. This track was obtained using an HMM tracker, with data similar to that above, except that the SNR was increased until no false

detections were obtained when the signal was present. The track initiates out of and terminates into the zero state which is indicated by the dots appearing in the window below the 9 active states in the gate. Note in this figure the last three scans for which the signal is present. Any tracker will have difficulty tracking these three measurements.

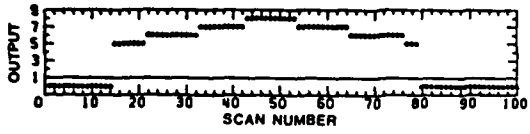


FIGURE 1. Track output at high SNR.

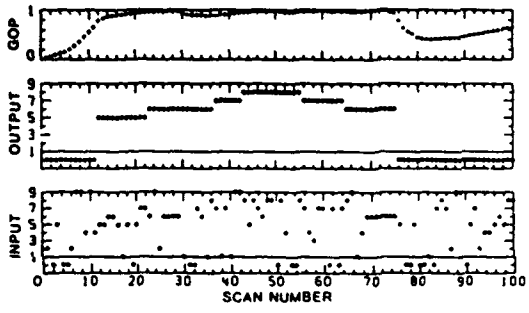


FIGURE 2. Input data, discrete track output & GOP for HMM tracker; data set 1.

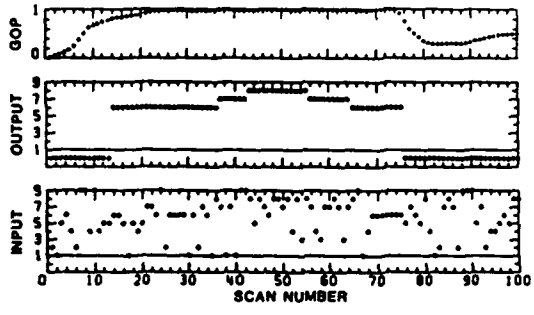


FIGURE 3. Input data, discrete track output & GOP for HMM/A tracker; data set 1.

Figures 2-5 show the input data, the discrete (Viterbi track) output, and the gate occupancy probability of the HMM and HMM/A trackers using the two different data sets. For the first data set both the trackers initiate early due to spurious data, but the HMM initiates in the correct state while the HMM/A does not. Both tracks terminate three cells early. For the second data set, the HMM fails to track the variation, but this is hardly surprising when one looks at the input data. The HMM/A on the other hand has additional information, viz. the amplitude of the measurements, and this information is critical to good tracking in this data set. The HMM/A initiates correctly, but terminates six scans early.

Figures 2-5 show very clearly the effects of statistical variations between the two data sets which can result in significantly different performances. This shows that both the HMM and HMM/A trackers need to be optimized for good performance in the ensemble statistical sense. Such an optimization procedure will require the development of ensemble statistical tracker performance measures which could then be used to compare the performance of different trackers. This is the subject of future work.

The input measurement data used for the HMM and HMM/A trackers are different even when they originated from the same data set. This arises because of the different detection thresholds used for HMM and HMM/A. A finite detection threshold is essential for the proper operation of the HMM tracker since the threshold level is an input to the HMM algorithm and indicates the significance of an individual detection. For these examples the best threshold for the HMM tracker was found to be 1.8 times the average noise level. For the HMM/A

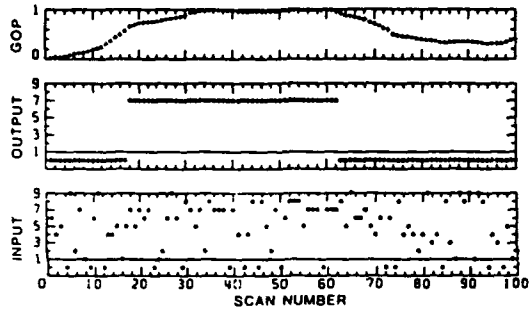


FIGURE 4. Input data, discrete track output & GOP for HMM tracker; data set 2.

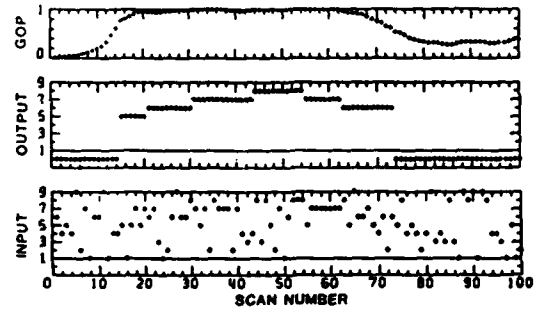


FIGURE 5. Input data, discrete track output & GOP for HMM/A tracker; data set 2.

tracker, the significance of each measurement is contained in the amplitude information. Raising the detection threshold in this case therefore tends to reduce the information available to the tracker, and the threshold is best kept small (Barrett & Streit, 1989).

Figures 6-9 show the input data, the track output, and the logarithm (base 10) of the estimated covariance of the frequency estimate for PDA trackers that use single and multiple gate measurements for the same two data sets. Note that even for a single data set there are slight differences in the single gate measurement input data used for the PDA and HMM. This is due to the use of different detection thresholds. It was not possible to find a common detection threshold for use with both PDA and HMM because PDA required a higher threshold than HMM for successful tracking.

The PDA tracker implemented here is a causal tracker, i.e. has no lag, and consequently one

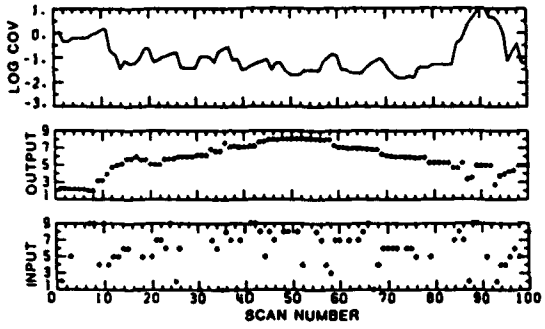


FIGURE 6. Input data, track output & log covariance for PDA tracker with single gate measurements; data set 1.

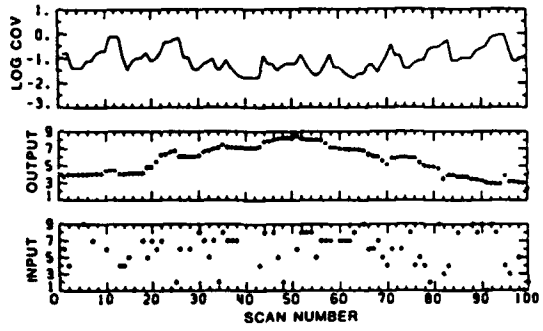


FIGURE 8. Input data, track output & log covariance for PDA tracker with single gate measurements; data set 2.

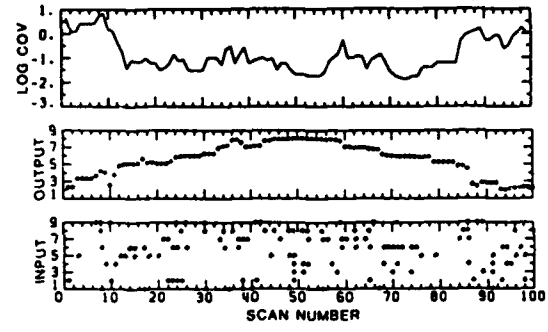


FIGURE 7. Input data, track output & log covariance for PDA tracker with multiple gate measurements; data set 1.

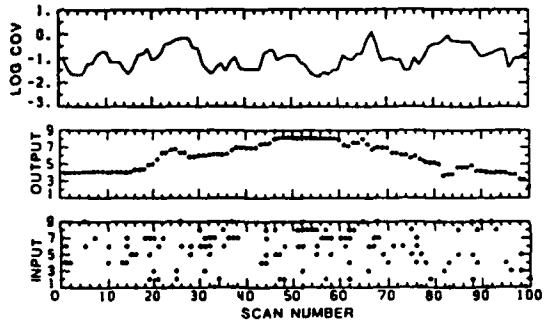


FIGURE 9. Input data, track output & log covariance for PDA tracker with multiple gate measurements; data set 2.

expects some delay in following dynamic track variations. The reluctance of the PDA tracker to change state resulted in tracks that lingered in the wrong state for up to five scans before switching to the correct state. The severity of this problem might be reduced by putting some lag into the PDA tracker (Mahalanabis, Prasad & Garg, 1986).

The most significant problems with our PDA tracker are the issues of track initiation and track termination. In these examples the tracker was initiated in cell five, but the signal did not exist so the tracker followed noise until scan 15. Similarly, the tracker followed noise after the signal terminated after scan 78. When the PDA tracker is following noise one would expect the covariance of the frequency estimate to be large. However, as can be seen in these figures there is no way to set a threshold on the estimated covariance that would satisfactorily indicate the presence or absence of a signal.

The multiple gate measurement PDA tracker shows little evidence of greater stability over the single gate measurement PDA for these two data sets. However, other data sets have shown that multiple measurements appear to give some robustness to the PDA tracker, making it less responsive to outliers, but at the cost of rendering it somewhat less responsive to state changes.

CONCLUSIONS

The development of ensemble statistical tracker performance measures must be accomplished in order to:

(i) optimise the tracking performance of any tracker, especially the HMM and HMM/A trackers;
(ii) provide a basis for comparison of all frequency line tracking algorithms; and
(iii) provide a means of comparing the automatic track initiation and termination characteristics of different algorithms.

These issues are especially important at low SNRs because then only ensemble statistical tracking behaviour is meaningful.

Our PDA algorithm as currently configured cannot satisfactorily initiate or terminate a track using PDA covariance estimates. The HMM and HMM/A trackers appear to offer significantly improved capabilities in this area, but ensemble performance measures are necessary to quantify track initiation and termination in a meaningful way.

From the discussion of the examples it is clear that the incorporation of amplitude information into our PDA tracker could offer some improvement in performance. On the other hand the incorporation of multiple gate measurements into both the HMM and HMM/A trackers could also enhance their performance. These extensions to our trackers would make them more directly comparable.

Finally, the incorporation of phase information into FLTAs is a most promising way of improving their performance. Similarly, the incorporation of derivative information into HMM based FLTAs should improve their performance because of better signal modelling.

REFERENCES

Barrett, R.F., Steele, A.K., and Streit, R.L. (1988). Frequency line tracking algorithms. Proc. of the NATO Advanced Study Institute on Underwater Acoustic Data Processing. Kingston, Ontario, Canada, July 18-29.

Barrett, R.F. and Streit, R.L. (1989). Automatic Detection of Frequency Modulated Spectral Lines. Proc. Australian Symposium on Signal Processing and its Applications. Adelaide, Australia April 17-19.

Bar-Shalom, Y. and Fortmann, T.E.(1988). Tracking and Data Association. Orlando, Florida, Academic Press.

Kopec, G.E. (1986). Format tracking using hidden Markov models and vector quantisation. IEEE Trans. Acoustics, Speech and Signal Processing. Vol. ASSP-34, August 1986, pp.709-728.

Mahalanabis, A.K., Prasad, S., and Garg, A. (1986). A Smoothing Algorithm for Improved Tracking in Clutter and Multitarget Environment. Proc. 1986 American Control Conference. Seattle, Washington, pp.908-910.

Streit, R.L. and Barrett, R.F. Frequency Line Tracking Using Hidden Markov Models. (Submitted for publication.)

Frequency Line Tracking Algorithms

R. F. Barrett, A. K. Steele and R. L. Streit

FREQUENCY LINE TRACKING ALGORITHMS

R.F. BARRETT & A.K. STEELE

Weapons Systems Research Laboratory, Defence Science and Technology Organisation, PO Box 1700, Salisbury, South Australia 5108, Australia.

R.L. STREIT

Naval Underwater Systems Center, New London, Connecticut 06320-5594, USA. (On exchange to Weapons Systems Research Laboratory).

1. INTRODUCTION

This paper has several purposes. The first purpose is to compare via simulation the performance of six different frequency line tracking algorithms (FLTA's) when used in conjunction with a simple threshold detector. The second purpose is show that the probabilistic data association (PDA) method for handling multiple detections is not limited to the Kalman filter context in which it has hitherto been presented. In particular, we present a PDA alpha-beta tracker that handles multiple detections without sacrificing algorithmic simplicity. The third purpose is to discuss a new tracker based on hidden Markov models (HMM's). An important and intrinsic feature of the HMM tracker is that it initiates and terminates tracks automatically.

In this paper, we treat the problem of tracking the time variation of the (instantaneous) frequency of an isolated tone embedded in additive white noise as a post-detection process applied to the evolving short term Fourier spectra of the sampled time series. Separating the tracking problem from the detection problem can lead to suboptimal tracking performance, but it is an approach commonly used in practice.

The choice of detector is also important, but throughout this paper we use a simple threshold detector because of its widespread usage and ease of implementation. No interpolation is employed to smooth the intrinsic quantization effects of this detector because interpolation is not justified at low SNR. Other well known problems associated with the threshold detector are outliers (false detections) and missed detections. These problems cause serious tracking errors in the conventional trackers studied in this paper; however, as the examples will show, PDA trackers and the HMM tracker are robust against outliers and missed detections and are capable of tracking down to the input quantization level.

The three conventional trackers studied in this paper are the alpha-beta [1], Kalman [2], and fixed lag Kalman smoothing trackers [3]. Each requires as input a single detection. The tracker input is the centre frequency of the FFT cell with the largest amplitude; however, if the detection threshold is not exceeded, no frequency measurement is made for the current scan, or block of time series data.

The two PDA trackers studied in this paper are the PDA alpha-beta and the PDA Kalman [4] trackers. (A PDA fixed lag Kalman smoothing tracker is also possible [5], but we have not yet implemented it). All PDA trackers accept multiple detections as input; i.e., the tracker input is the set (possibly empty) of centre frequencies of all FFT cells whose amplitude exceeds the detection threshold.

The last tracker studied in this paper is the HMM tracker [6]. It can accept as input either multiple detections or the strongest detection; however, the version studied here uses the same input as the conventional trackers mentioned above. It is unique among the trackers studied in this paper in its ability to initiate and terminate tracks automatically.

2. BRIEF DESCRIPTION OF THE TRACKERS

2.1 The Conventional Trackers

The conventional tracking algorithms accept a single input frequency measurement and generate two output quantities: the track frequency estimate $\hat{f}$ and the time derivative $\dot{\hat{f}}$ for the current scan. The mathematical form of the track dynamic models assumes that f is constant over the scan update interval, a reasonable assumption if the change in f over the scan update interval is small. The two Kalman trackers modify this simple dynamic model for f by corrupting it with additive Gaussian process noise; however, the alpha-beta tracker does not explicitly include a process noise term. Process noise accounts for mismatch between the assumed track dynamic model and true track dynamics. The fixed lag Kalman smoothing tracker [3] differs from the other two conventional trackers in its utilisation of track measurements from scans in advance of the (fixed lag) estimation point to improve the output track estimate.

The problem with conventional trackers is that they are linear systems. Consequently, their response to outliers is governed by their impulse response function, while their response to missed detections is by comparison much less important. Outliers and missed detections are common at low SNR, so optimising a conventional tracker is essentially equivalent to optimising its impulse response function. One way to avoid the impulse response function issue is to avoid trackers that are linear functions of the measurement sequence. The PDA trackers and the HMM tracker discussed below are nonlinear systems, and their response to outliers is much superior to that of the conventional trackers.

2.2 The PDA Trackers

We present PDA as a method for converting a single-input-single-output (SISO) tracker into a multiple-input-single-output tracker. PDA assumes that only one of the multiple input measurements corresponds to the target being tracked. It further assumes that measurements in the next scan will be normally distributed with the mean and covariance predicted from the current scan. PDA is thus applicable to any SISO tracker in which measurement mean and covariance at the next scan are predicted.

The conventional Kalman tracker predicts the target state and error covariance, and the predicted measurement mean and covariance follow from the general Kalman system equations. On the other hand, the alpha-beta tracker predicts the target state, but not the error covariance. For the FLTA, we interpret the target state as the predicted measurement mean and supplement the constants α and β with another constant σ^2 denoting the covariance of the measurements in the next scan. The PDA alpha-beta tracker is completely specified by (α, β, σ).

Track input measurements are gated, thus creating the possibility of false dismissal of the target measurement. Using the PDA assumptions, the probability of false dismissal is easily evaluated. The usual gated PDA method uses a variable gate to achieve constant probability of false dismissal. For the FLTA, we use a fixed gate so the false dismissal probability varies from scan to scan.

Let β_i be the probability that the i-th measurement corresponds to the target, and let β_o be the probability that none of the measurements

corresponds to the target. Similarly, let $\hat{f}_i$ be the track frequency estimate generated by using the i-th measurement in the underlying SISO tracker, and let $\hat{f}_o$ be the predicted track frequency when no measurement is made. Then the PDA tracker output is given by $\hat{f}_{PDA} = \beta_o \hat{f}_o + \Sigma_i \beta_i \hat{f}_i$.

For the PDA alpha-beta tracker, the necessary predicted measurement mean and covariance are the obvious ones obtained from $\hat{f}_{PDA}$ and (α, β, σ). For the PDA Kalman tracker, the error covariance associated with $\hat{f}_{PDA}$ is also computed (see [4]), and it is used with $\hat{f}_{PDA}$ to make the measurement mean and covariance predictions in the underlying SISO Kalman filter.

The probabilities $\{\beta_i\}$ are easily computed and require only the evaluation of a truncated Gaussian density function. The nonlinear dependence of all PDA trackers on the measured track input data is due to the nonlinear dependence of the probabilities $\{\beta_i\}$ on the data.

2.3 The HMM Tracker

HMM's are probabilistic models that are commonly used in speech applications. Their utility in tracking applications seems not to be recognised in the general literature, except for a paper by Kopec [7] who uses them to track formants, or resonances, in spoken words. The HMM tracker presented in this paper is similar to Kopec's formant tracker; however, the FLTA application permits the analytical development of the parameters defining the underlying HMM.

The HMM tracker is a fixed interval FLTA; i.e., it takes a fixed length sequence, or "window", of measurements and outputs a track estimate for each time in the window. By sliding the window along as new data are collected, the track estimate evolves in time. Alternatively, one may simply increase the window size. Either way, the HMM tracker is used only to compute the output track estimate for each given tracking window.

The quantised frequency track is modelled as a finite state Markov chain. A "faded" state represents a track whose SNR is less than the tracker SNR (see below); the remaining "active" states represent a track occupying an FFT cell inside a fixed gate and having an SNR greater than the tracker SNR. Track initiation is defined as a transition from the faded state to any active state, while track termination is defined as a transition from any active state into the "faded" state. Initiation and termination of tracks, as well as movement of the track within the gate, are therefore governed by the transition probability matrix, A, of the Markov chain.

Measurements of the frequency track are characterised by a detection probability matrix, B. Thus, for each possible target state, the threshold detector outputs are measured target states with probabilities that are computed analytically from the SNR and the threshold. The SNR assumed for this B-matrix calculation is called the tracker SNR; in effect, it is the lowest SNR at which tracks are initiated and estimated.

The HMM tracker is fully specified by the A- and B- matrices, together with the initial state probability matrix, π. Initially, π corresponds to a target in the faded state. This forces automatic track initiation. If the tracking window slides along as new data are collected, π is updated using current HMM tracker output.

The HMM tracker outputs both a discrete (quantised) track and a continuous track. The quantised track is the Viterbi track; i.e., of all possible tracks (realisations of the Markov chain), the Viterbi track is the one most likely to account for the measurement sequence. The continuous track is essentially the expected track, with the expectation taken over all possible realisations of the Markov chain. Strictly

speaking, the expected track is conditioned on the track not having faded, as well as on the measurements. The total probability of a faded track (conditioned only on the measurements) at each point in the window is thus a necessary complement to the continuous track estimate.

The discrete output of the HMM tracker is computed using only n^2T additions, while the continuous output uses n^2T multiplications, where n is the number of Markov chain states and T is the number of scans in the window. The discrete and continuous algorithms are easily vectorised.

3. EXAMPLES

All six trackers are compared in Figure 1, and the nonlinear trackers are compared in Figure 2. The gated input measurement is indicated by a dot, and the tracker output is the continuous curve. The data are obtained by generating a sine wave triangularly swept in frequency with a period of 400 scans, a centre frequency of 10, and a maximum deviation of 5. The added uncorrelated noise level and the detection threshold are set to give a reasonable number of false detections and missed detections. The conventional and HMM trackers use the largest detection in the gate; the fixed lag Kalman smoother has a lag of 10 scans; the PDA trackers use all detections in the gate; the HMM tracker uses a window of 250 scans.

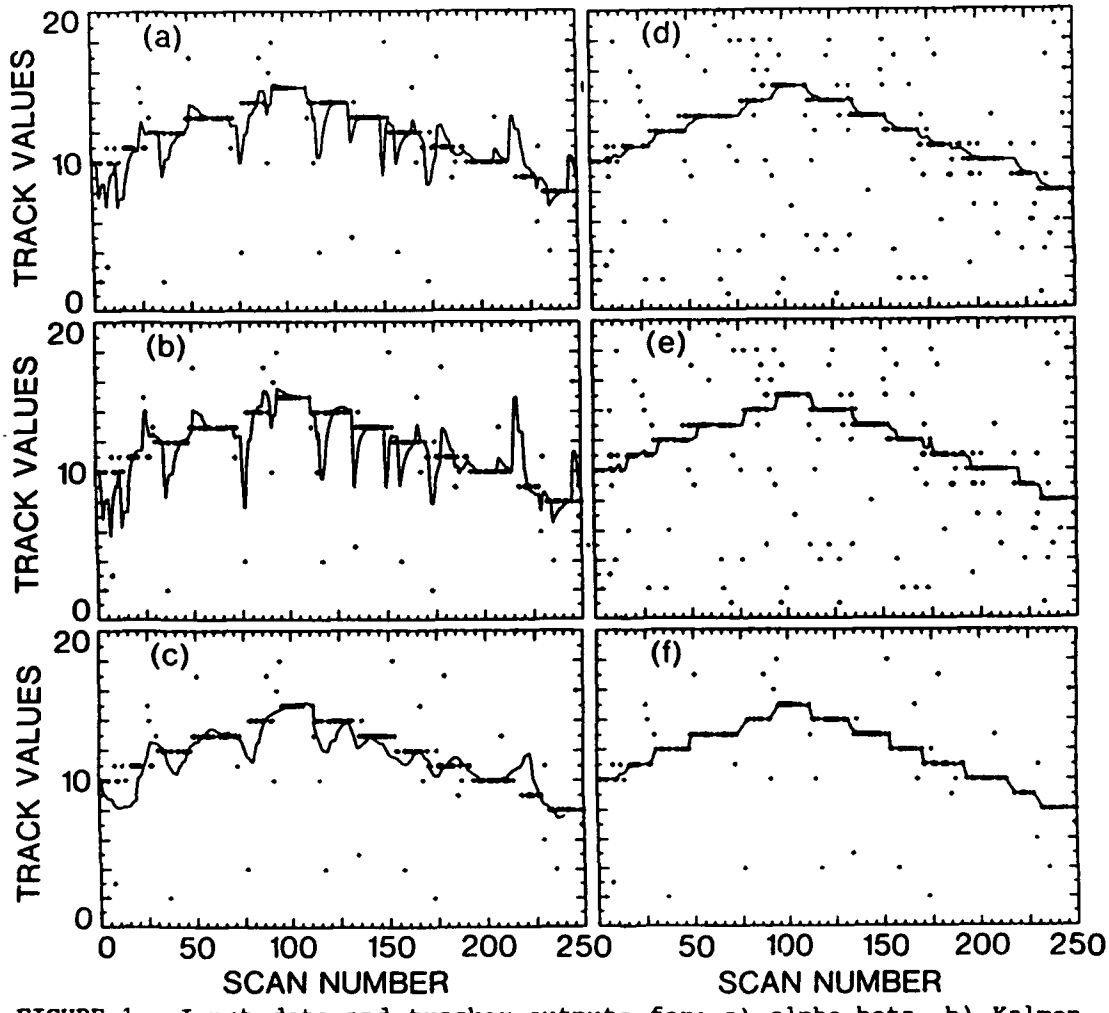


FIGURE 1. Input data and tracker outputs for: a) alpha-beta, b) Kalman, c) fixed lag Kalman smoother, d) PDA alpha-beta, e) PDA Kalman, f) HMM.

The PDA trackers are initiated at the correct track value and the HMM tracker is initiated in the faded state. In Figure 1 the SNR is constant over all scans. It shows that the nonlinear trackers are robust against outliers, whereas the conventional trackers are not. In Figure 2 the SNR starts low, increases linearly to scan 100, is constant to scan 150, and decreases linearly to scan 250. The track is the same as in Figure 1, but the false alarm rate is increased by decreasing the detection threshold. The PDA trackers are incorrect at low SNRs. The automatic track initiation and termination of the HMM tracker is clearly evident.

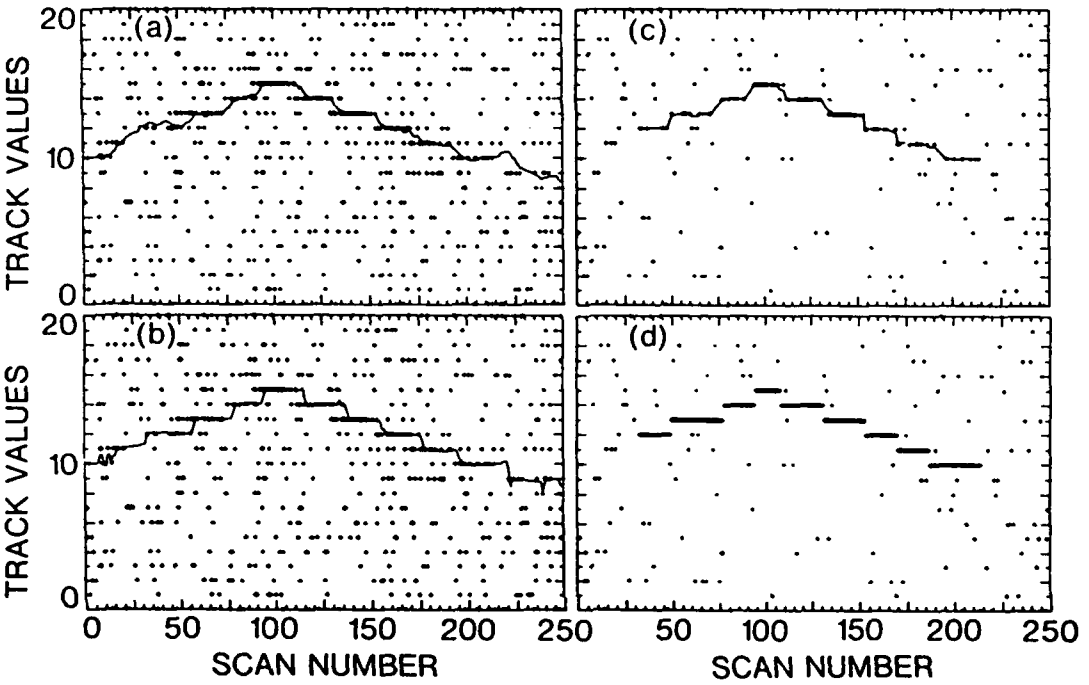


FIGURE 2. Input data and tracker outputs for: a) PDA alpha-beta, b) PDA Kalman, c) continuous output HMM, d) discrete output HMM.

REFERENCES

1. Benedict TR, Bordner GW: Synthesis of an Optimal set of Radar Track-While-Scan Smoothing Equations. IRE Trans. Automatic Control, Vol. 7, July 1962, pp.27-32.
2. Friedland B: Optimum Steady-State Position and Velocity Estimation Using Noisy Sampled Position Data. IEEE Trans. Aerospace and Electronic Systems, Vol. AES-9, November 1973, pp.906-911.
3. Moore JB: Discrete-Time Fixed-Lag Smoothing Algorithms. Automatica, Vol.9, 1973, pp.163-173.
4. Bar-Shalom Y, Fortmann TE: Tracking and Data Association. Orlando, Florida, Academic Press, 1988.
5. Mahalanabis AK, Prasad S, Garg A: A Smoothing Algorithm for Improved Tracking in Clutter and Multitarget Environment. Proc. 1986 American Control Conference, Seattle, WA, 1986, pp.908-910.
6. Streit RL, Barrett RF: Frequency Line Tracking Using Hidden Markov Models. In preparation.
7. Kopec GE: Format Tracking Using Hidden Markov Models and Vector Quantisation. IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-34, August 1986, pp.709-728.

Frequency Line Tracking Using Hidden Markov Models With Phase Information

R. F. Barrett and R. L. Streit

Frequency Line Tracking Using Hidden Markov Models with Phase Information

R.F. Barrett
Maritime Systems Division, Weapons Systems Research Laboratory, DSTO, P.O. Box 1700, Salisbury, Australia 5108
R.L. Streit
Naval Underwater Systems Center, New London, Connecticut, U.S.A.

An extension to the Hidden Markov Model (HMM) frequency line tracker of Streit and Barrett (1990) is presented. In this extension, the FFT amplitudes and phases in a restricted set of states centered on the signal frequency are passed to the tracker as an input. The result is an improved tracking performance and a new ability of the tracker to follow frequency fluctuations within one FFT bin.

1. Introduction

The estimation of the frequency of isolated tones embedded in noise, and the tracking of changes in these estimated frequencies as a function of time are two related topics that have recently received considerable attention in the signal processing literature. Techniques for the solution of these problems have found applications in many diverse fields (e.g., radar, sonar, seismology, etc).

In a conventional frequency tracking problem, the incoming data are divided into blocks of contiguous time series. Fast Fourier Transforms (FFTs) are then performed on each of these blocks to obtain frequency spectra as a function of time. The resolution of the frequency spectra is restricted to one FFT cell (or the reciprocal of the data acquisition time for the FFT integral). The conventional frequency tracking problem consists of estimating the true signal frequency from the FFT spectral data.

In an earlier paper (Streit and Barrett (1990)), we have studied the formulation of frequency line tracking in terms of a Hidden Markov Model (HMM). In this approach, the frequency domain considered by the tracker is restricted to a subset, or gate, of the available FFT cells. For each time block, the cell containing the maximum power within the gate, provided that power exceeds a detection threshold, is passed to the HMM tracker. Based on *a priori* knowledge of some specified parameters describing the statistical nature of the track, the HMM tracker reconstructs the time variation of the signal frequency within the gate. A zero state is included to allow for the case when the signal terminates or the signal frequency wanders outside of the prescribed gate. Full details of the HMM frequency line tracker are contained in an earlier publication.

An extension of this basic tracker was developed by Barrett and Streit (1989) and Steele *et al* (1989). The tracker was modified so that it received as input the FFT amplitude in the cell with the maximum spectral power. This extra information was found to enhance its performance by enabling tracking and detection at lower signal-to-noise ratios (SNRs) than without amplitude (see Barrett and Streit (1989)).

The input to a frequency tracker is generally some form of frequency estimator, and in this area also, a number of new approaches have been developed. These methods offer improvements in either accuracy or resolution over conventional spectral analysis techniques. For the case of a single tone embedded in noise, the Phase Interpolation Estimator (PIE) and Generalised Phase Interpolation Estimator (GPIE) provide near optimal frequency estimates, even at low SNR. For examples, see McMahon and Barrett (1986, 1987).

In these methods, the phase information from successive FFTs is used to refine the estimate of the signal frequency so that frequency changes of less than the width of one FFT cell can be measured. Previous track history is not taken into account, and the frequency estimates for different time blocks are independent. At low SNR, the 2π ambiguity in FFT phase can occasionally result in 'outliers' far removed from the correct frequency.

The purpose of the work described in the present paper is to include FFT amplitude and phase information into the HMM frequency tracker described by Streit and Barrett (1990). The added information affects the performance in two different ways. Firstly, the intrinsic tracker quantisation error can be reduced so that frequency variations within one FFT cell can be readily tracked. Secondly, the added information enables the new tracker to out-perform the earlier tracker, even if the intrinsic tracker quantisation error is kept to one FFT resolution cell (as in the earlier tracker). The problem of outliers that occurred with the PIE algorithm is avoided because these extreme frequency fluctuations are suppressed by the Markov chain process model of the HMM tracker. The extended HMM tracker thus combines the high accuracy frequency estimation capability of the PIE algorithm with the accurate tracking and track initiation and termination capabilities of the HMM tracker.

2. Frequency Tracking Using HMMs

In the frequency tracking problem described by Streit and Barrett (1990), the range of frequencies, or gate, over

which the track is allowed to wander is divided into a finite number m of frequency cells. A zero state is included to allow for the possibility of the track wandering outside of the allowed frequency range, or terminating altogether.

The transitions that occur between states as time progresses are characterised by a transition probability matrix A. The elements of the A-matrix are the probabilities of transitions between the states of the Markov chain. These probabilities depend on the initiation and termination probabilities (u and v) of the track, and on the process noise, d. The elements of the A-matrix are calculated from the basic premise that the probability distribution for a frequency change in the tracked line is a Gaussian centred on zero. The width of this Gaussian distribution is controlled by the process noise d. In addition to a change from one frequency cell to another within the gate, transitions may also occur from within the gate to the zero state (track termination), or from the zero state to a state within the gate (track initiation).

The specification of the measurement probability matrix B is also necessary to define the HMM. In the basic HMM tracker of Streit and Barrett (1990), a measurement is defined as the specification of the frequency cell in which the maximum spectral power resides, provided that power exceeds a prescribed threshold. An element of the B-matrix is then the likelihood of such a detection in one of the cells of the gate, conditioned on the true signal residing in a particular state of the Markov chain. The calculation of the elements of the B-matrix is carried out under the assumption that the time series data comprise a constant frequency sinusoidal signal embedded in white Gaussian noise. The likelihoods depend on the SNR of the line being tracked and on the measurement detection threshold.

The last remaining quantity to be specified to the tracker is the initial state probability vector π. This vector specifies the occupation probabilities of the various Markov states at time zero.

There are three outputs from the HMM tracker described in Streit and Barrett (1990). The first is the Viterbi track, which is the optimal state sequence (in a maximum likelihood sense) conditioned on the measurement sequence. The Viterbi track is thus a discrete output. The second output, which is continuous, is the Mean Cell Occupancy (MCO) track. From the A and B matrices, the likelihood of the occupancy associated with each Markov state as a function of time can be calculated. The mean cell occupancy (for cells within the gate), and the associated standard deviation can then be obtained. The third output of the HMM tracker is the Gate Occupancy Probability (GOP). The GOP function is the probability, as a function of time, of a frequency with the required parameters lying within the selected gate. The complementary function to the GOP (i.e., 1 - GOP) is the occupation probability of the zero state.

3. Inclusion of FFT Phase

The performance of any tracker clearly depends on the amount of information that is available to it. In this section we describe a HMM tracker in which the concept of a 'measurement' is further extended beyond that of Streit and Barrett (1990) and Barrett and Streit (1989). For each block of time series data, the amplitude and phase of the FFT for each cell within the gate are determined. The complete specification of the FFT amplitudes and phases in all gate cells for two contiguous blocks of time series data is deemed a 'measurement'. As before, the B-matrix is interpreted as the likelihood of any particular measurement, conditioned on each of the underlying Markov states.

Following Streit and Barrett (1990), it is assumed that the data time series is of the form:

$$z(t_0 + kT_s) = A\sin[\tilde{\omega}(t_0 + kT_s) + \xi] + n_k \tag{1}$$

where t_0 is the initial time , and T_s is the sampling period. The amplitude A, phase ξ and angular frequency $\tilde{\omega}$ are assumed to remain constant over the period NT_s, which is the data acquisition time for a Fourier transform of size N. The noise n_k is taken to be zero mean and Gaussian in nature, with a variance of σ^2, so that

$$\langle n_k n_j \rangle = \delta_{kj}\sigma^2 \tag{2}$$

where δ denotes the Kronecker delta.

The discrete Fourier Transform $\chi(\omega)$ at angular frequency ω of the time series in Eq. 1 is defined in Streit and Barrett (1990) and can be expressed in the form:

$$\begin{aligned}\chi(\omega) &= Re^{j\eta} \\ &= Ce^{j\phi} + De^{j\theta}\end{aligned} \tag{3}$$

where R and η represent the amplitude and phase of $\chi(\omega)$. The amplitudes and phases of the signal and noise components are denoted by (C,ϕ) and (D,θ) respectively.

The joint Probability Density Function (PDF) of R and η has the form

$$P(R,\eta) = \frac{R}{2\pi\tilde{\sigma}^2} e^{-\frac{R^2 - 2AR\cos(\eta - \phi) + A^2}{2\tilde{\sigma}^2}} \tag{4}$$

where $\tilde{\sigma}^2 = \sigma^2/2N$.

The signal phase ϕ differs for the different time blocks. If we define ϕ_1 to be the phase corresponding to the signal for the first of two contiguous time blocks, and ϕ_2 to be the phase corresponding to the second time block, then

$$\phi_2 \approx \phi_1 + \tilde{\omega}NT_s \tag{5}$$

For real signals, such as that defined in (1), the approximation in (5) is only valid at frequencies far removed from zero. However, if the analytic signal is used instead of the real signal, eq. (5) becomes valid at all frequencies.

Eq. (5) implies that the angular frequency $\tilde{\omega}$ is related not to the phase in each cell, but to the difference in the phases in the cell containing the signal for the two contiguous time blocks. We are thus interested more in the phase change in a cell as time progresses, than in the absolute value of the phase. Our concept of a 'measurement' is thus extended to include the simultaneous specification of the quantities R_{n1}, R_{n2} and $\eta_{n2} - \eta_{n1}$ where R_{n1}, η_{n1} are the amplitudes and phases of the Fast Fourier Transforms in cell n at time $t = t_0$, and R_{n2}, η_{n2} are the corresponding quantities at time $t = t_0 + NT_s$, where $1 \le n \le m$ With this definition of a measurement,

overlapping causes successive measurements to be correlated, which is contrary to the HMM model. This contradiction is ignored here, but the overlapping can easily be removed if it results in difficulties.

The likelihood of the measurement, conditioned on an initial signal amplitude A and frequency $\tilde{\omega}$, can be shown to have the form:

$$B(\tilde{\omega}) = \prod_{n=1}^{m} P(R_{n1}, R_{n2}, \eta_{n2} - \eta_{n1} | A, \tilde{\omega}) \qquad (6)$$

where

$$P\ (R_{n1}, R_{n2}, \eta_{n2} - \eta_{n1} | A, \tilde{\omega}) = \frac{R_{n1} R_{n2}}{2\pi\bar{\sigma}^4} \times e^{-\frac{\left[R_{n1}^2 + R_{n2}^2 + 2A^2\right]}{2\bar{\sigma}^2}} I_0\left(\frac{Az}{\bar{\sigma}^2}\right) \qquad (7)$$

In eq.(7), I_0 represents the modi...d Bessel function and

$$z = \sqrt{R_{n1}^2 + R_{n2}^2 + 2R_{n1}R_{n2}\cos(\eta_{n2} - \eta_{n1} - \tilde{\omega}NT_s)} \qquad (8)$$

The implementation of the HMM tracker with phase and amplitude included differs from the earlier implementation described in Streit and Barrett (1990) in a number of important points. Firstly, because the presence of phase information enables frequencies to be estimated to a greater accuracy than one FFT cell, the states of the HMM need not coincide with the FFT cells. In the present work, the frequency range spanned by each HMM state is arbitrary. The state width and process noise d can be adjusted to reflect the increased estimation accuracy, particularly at high SNRs, allowed by the phase information.

The second difference lies in the fact that the likelihood function in eq (6) can be highly peaked as a function of $\tilde{\omega}$ if the SNR is fairly high. The calculation of likelihoods at the centre frequencies of the gate cells may therefore not be an accurate enough representation of the average likelihood over the span of the state. This effect can be countered, either by selecting the frequency span of each state to be significantly smaller than the spread in the likelihood function, or alternatively by integrating $B(\tilde{\omega})$ over the frequency span of the state. The particular selection will depend on the accuracy desired in the frequency track.

4. Discussion of Results

Results of the application of the HMM frequency line tracker to two sets of simulated data are presented in Figs. 1 and 2. In Fig. 1, the results from three different versions of the tracker are compared. These versions are those of i) Streit and Barrett (1990) in which no FFT amplitude or phase information is passed to the tracker; ii) Barrett and Streit (1989) in which the FFT amplitude from the cell containing the maximum power is used; and iii) the tracker presented here where FFT amplitudes and phases in all the cells are passed to the tracker. For convenience, these three trackers are designated HMM, HMM/A and HMM/AP respectively.

Fig. 1a shows an intensity modulated representation of the spectral power in the cells of the gate as a function of time. Frequency increases in the vertical direction, while time increases horizontally. Each cell is 1 Hz wide in the frequency direction and 1 Sec. long in the time direction. The total time window spans 100 seconds and there are 9 cells in the frequency gate. The parameters of the HMM are listed in the Figure caption. The SNR is defined as $A^2/2\sigma^2$. The signal consisted of a frequency modulated tone with the modulation having the form of a sinusoid of amplitude 2 Hz and period 50 seconds.

The measurement sequence, which is input to the basic tracker, is displayed in Fig. 1b. The effect of the noise is clear in the appearance of the track. In Figs. 1c - 1e, the Viterbi tracks arising from the HMM, HMM/A and HMM/AP trackers are displayed. The zero state is shown below the other state cells. An improvement to the quality of the output through Figs 1c - 1e is apparent, as the amount of input information presented to the tracker is increased. The improved quality manifests itself in fewer abrupt terminations and re-initiations of the track (i.e., entries into the zero state), and in fewer outliers.

The results displayed in Fig. 1 are typical. However, a quantitative comparison of the three trackers can only be made by comparing their average performance for a statistically significant set of realisations of time series data. The results of such an investigation will be published later.

In Fig. 2, the results for a set of data where the frequency modulation of the signal has been reduced to 0.6 Hz amplitude are displayed. All other details are the same as for Fig. 1. The frequency cells in the intensity modulated display of Fig. 2a are 1 Hz wide. However, for the displays 2b and 2c, the frequency scale has been expanded by the factor 3.33 so that the frequency cells are now 0.3 Hz wide.

Fig 2b displays the results of the PIE frequency estimation routine. The general trend of the modulation is apparent in the expanded display, but many outliers are observed. The frequency estimates that were outside the range spanned by the state cells are shown in the zero state.

Figs. 2c and 2d show the Viterbi tracks and MCO outputs from the HMM/AP tracker. The MCO display shows two traces that are respectively one standard deviation above and one standard deviation below the MCO track. The GOP function is displayed in Fig. 2e (ranging from values of 0 to 1) and indicates a high probability that the signal was present at all times.

The increased accuracy of the HMM/AP tracker (and the PIE) arises from the exploitation of the phase information available in successive FFTs. This information was not available to the HMM and HMM/A trackers and so the fine structure in the frequency modulation could not be detected by these trackers.

5. Conclusion

The HMM frequency line tracker presented by Streit and Barrett (1990) has been extended by the inclusion of FFT phase information into the tracker input measurement sequence. As a result, the tracker exhibits an improved performance, and is able to accurately track smaller frequency changes than was previously possible.

References

Barrett R.F. and Streit R.S. (1989)
Automatic Detection of Frequency Modulated Spectral Lines
Proc Australian Symp. on Sig. Proc. and Appls., Adelaide, pp283-287

McMahon D.R.A. and Barrett R.F. (1986)
An Efficient Method for the Estimation of the Frequency of a Single Tone in Noise from the Phases of Discrete Fourier Transforms
Signal Processing Vol. 11, pp169-177

McMahon D.R.A. and Barrett R.F. (1987)
Generalisation of the Method for the Estimation of the Frequencies of a Tones in Noise from the Phases of Discrete Fourier Transforms
Signal Processing Vol. 12, pp371-383

Steele A.K., Streit R.L. and Barrett R.F. (1989)
Nonlinear Frequency Line Tracking Algorithms
Proc Australian Symp. on Sig. Proc. and Appls., Adelaide, pp258-262

Streit R.L. and Barrett R.F. (1990)
Frequency Line Tracking Using Hidden Markov Models
IEEE Trans. ASSP V38, pp586-598

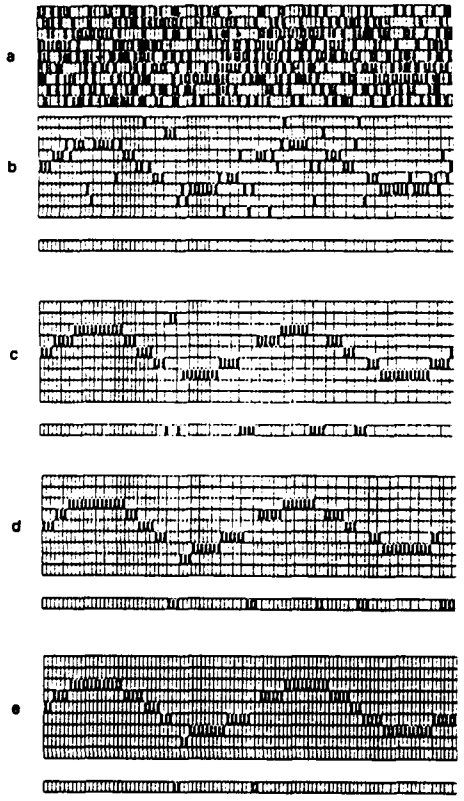


Fig.1 Frequency tracks for a sinusoidally modulated (2 Hz amplitude) tone in noise: a) intensity modulated spectrogram; b) Measurement sequence for basic HMM tracker; c) Viterbi output for basic HMM tracker; d) Viterbi output for HMM/A tracker; e) Viterbi output for HMM/AP tracker.
(HMM parameters are u = 0.5, v = 0.1, d = 0.3, Thresh. = 0.0, N = 1024, NT_s = 1 sec, SNR = 0.008 (-21 dB), Cell width = 1 Hz)

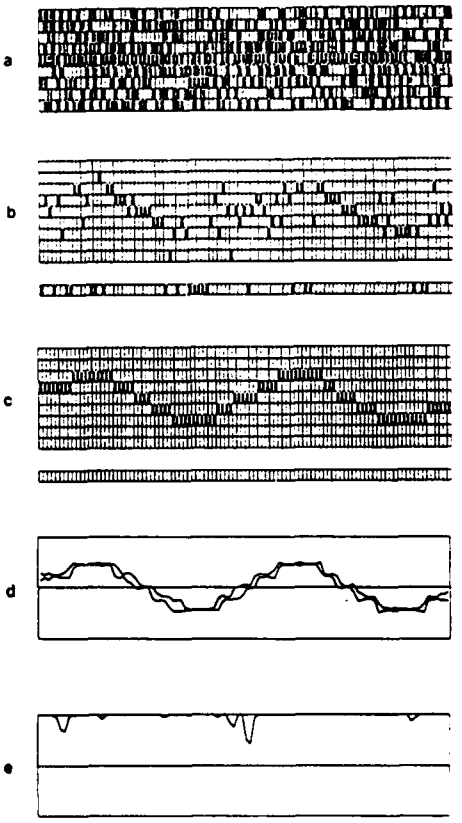


Fig. 2 Frequency tracks for a sinusoidally modulated (0.6 Hz amplitude) tone in noise: a) intensity modulated spectrogram; b) PIE frequency estimates; c) Viterbi track from HMM/AP tracker; d) MCO track from HMM/AP tracker; e) GOP function from HMM/AP tracker.
(HMM parameters as for Fig. 1, except that cell width is 0.3 Hz)

Frequency Line Detector/Tracker Using Hidden Markov Models With Amplitude Information

R. L. Streit

Abstract

Four different methods for utilizing amplitude in hidden Markov model (HMM) detector/trackers are presented. All four of the HMM detector/trackers are algorithmically identical in their basic structure. The only differences between the proposed trackers are confined to the conditional probability density functions (PDFs) of the input measurements. The fundamental HMM algorithm structure is presented in matrix form, and the necessary conditional PDFs for the four detector/trackers are derived.

Introduction

The papers [1, 2] are the first papers to discuss the inclusion of amplitude information into detector/trackers based on hidden Markov models (HMMs); however, these papers do not give a detailed description of the way in which amplitude information is utilized. Of the many ways to include amplitude information into the measurement sequence of the HMM detector/tracker, four are described here. We denote these four detector/trackers by HMM/A00, HMM/A01, HMM/A10, and HMM/A11. The notation HMM/A•• indicates the presence or absence of decision and quantization steps (defined below) in the obvious manner. HMM/A11 is identical to the detector/tracker described in the fundamental HMM detector/tracker paper [3], and HMM/A10 is identical to the best of the detector/trackers discussed in papers [1] and [2]. Familiarity with the content and notation of [3] is assumed here.

An important feature of detector/trackers based on HMMs is that they all have fundamentally identical algorithmic structures. Different input measurements change the measurement likelihood function, but do not otherwise affect the detector/tracker algorithm. Because of this feature, we describe below only the HMM forward-backward algorithm that is used to construct the probability field from which the continuous outputs of the HMM detector-tracker are derived. The discrete output is the Viterbi track, and it is computed by a dynamic programming algorithm. The minor changes required to compute the Viterbi track are analogous to those required for the continuous outputs. For further background, see [3].

The detector/tracker HMM/A11 uses a fixed frequency gate, that is, a fixed contiguous subset of size $n \geq 2$ of the full set of DFT (discrete Fourier transform) frequency cells. The DFT size is $N \geq n$ and the fixed gate cells are indexed, for convenience, 1, 2, ... , n. Amplitude information is utilized only indirectly, that is, the input measurement to HMM/A11 is the index of the DFT cell having the largest amplitude, provided this cell exceeds a specified threshold, D, and

lies within the gate. The measurement z is the output of a two step measurement process. The first step, called the decision step, chooses the largest amplitude DFT cell. The outputs of the decision step are the DFT cell index $i \geq 1$ and its amplitude r_i. The second step, called the quantization step, quantizes the amplitude r_i to one bit. The amplitude threshold determines the one bit quantization breakpoint. The output of the quantization step is the DFT cell index i if $r_i \geq D$, and the integer 0 if $r_i < D$. The measurement process of the HMM/A11 detector/tracker therefore suppresses (compresses) available amplitude data in the each step. The data compression steps affect detector/tracker performance in different ways.

The distinction between HMM/A00, HMM/A01, HMM/A10, and HMM/A11 is their different measurement processes. The HMM/A00 measurement process eliminates both the decision and quantization steps. Since HMM/A00 utilizes all the available amplitude information, its performance should be the best of the four. The HMM/A10 measurement process eliminates the quantization step, but not the decision step, of the basic HMM measurement process. The HMM/A01 measurement process eliminates the decision step but not the quantization step. As described above, the HMM/A11 measurement process uses both decision and quantization steps.

A detector/tracker HMM must have two types of states -- those corresponding to "signal absent" hypotheses and those corresponding to "signal present" hypotheses. If either type of state is not used, then the HMM cannot be described as a detector/tracker. In the frequency line tracking application utilizing fixed gates, a signal can be absent because it has faded or because it has exited either to the right or to the left of the fixed gate. The signal can be present in more than one way also, i.e., the signal can lie in any one of the DFT cells in the gate. For this discussion, we use n "signal present" states and one "signal absent" state. The "signal absent" state is indexed 0 and called the zero state. The n "signal present" states are numbered from 1 to n and correspond to signal frequency lines centered in the n DFT cells of the gate.

HMM Detector/Tracker Algorithm

Let $Z = (z_1, \ldots, z_T)$ denote a measurement sequence of length $T \geq 1$ that is input to an n+1 state HMM detector/tracker. The forward algorithm computes the vectors $\alpha_t \in \mathbb{R}^{n+1}$ by means of an algorithm that is easily stated in terms of matrix products. Let $'$ denote matrix transposition. Then the forward algorithm is defined by

$$\begin{aligned} \alpha_1 &= B(z_1)\,\pi \\ \alpha_{t+1} &= B(z_{t+1})\,A'\,\alpha_t, \qquad t = 1, \ldots, T-1, \end{aligned} \tag{1}$$

where $\pi \in \mathbb{R}^{n+1}$ is the initial state probability vector, $A = [a_{ij}] \in \mathbb{R}^{(n+1)\times(n+1)}$ is the state transition probability matrix, and, for arbitrary measurements z, the matrix $B(z) \in \mathbb{R}^{(n+1)\times(n+1)}$ is defined by

$$B(z) = \text{Diag}\,[b_0(z),\ b_1(z),\ \ldots,\ b_n(z)],$$

and where $b_i(z)$ denotes the likelihood function of the measurement z conditioned on the signal state i, i = 0, 1, ..., n. Similarly, the backward algorithm computes the vectors $\beta_t \in \mathbb{R}^{n+1}$ by means of an algorithm that is equivalent to the matrix product

$$\begin{aligned} \beta_T &= (\,1\ \ 1\ \cdots\ 1\,)' \in \mathbb{R}^{n+1} \\ \beta_t &= A\,B(z_{t+1})\,\beta_{t+1}, \qquad t = T-1, \ldots, 1. \end{aligned} \tag{2}$$

Probabilistic interpretations for the vectors α_t and β_t are given in [3]. The continuous detector/tracker outputs are derived (see [3]) from the probability field, denoted by F(t,i). The numerical value of F(t,i) is the probability the signal occupies state i at time t, conditioned on all the available measurements Z. It is defined in terms of the components $\{\alpha_t(i)\}$ and $\{\beta_t(i)\}$ of α_t and β_t by

$$F(t,i) = \alpha_t(i)\,\beta_t(i)\left[\sum_{k=0}^{n} \alpha_t(k)\,\beta_t(k)\right]^{-1}. \tag{3}$$

In equation (3), t ranges from 1 to T, and i ranges from 0 to n. An arbitrary nonzero scale factor κ_t can be applied to the diagonal matrix $B(z_t)$ without altering the probability field $F(t,i)$ because such scale factors cancel out in the definition (3). The judicious use of scale factors can result in reduced complexity in the required likelihood functions, and give greater insight into the underlying theoretical structure.

From equations (1)-(3), it is clear that the different measurement characteristics affect the detector/tracker output only via the state likelihood functions $b_i(z)$. The algorithms are otherwise blind to the measurement. The likelihood functions $b_i(z)$ for each of the four amplitude measurement processes are derived below. Two probability density functions (PDFs) of the measured amplitude r in a DFT cell are required. The PDF of r conditioned on the simple hypothesis "signal in state $i \neq 0$" is given by

$$P_1(r) = (2rN/\sigma^2)\, I_0(ArN/\sigma^2)\, \exp[-N(4r^2+A^2)/(4\sigma^2)], \qquad (4)$$

and the PDF of r conditioned on the simple hypothesis "signal in state $i = 0$" is given by

$$P_2(r) = (2rN/\sigma^2)\, \exp[-Nr^2/\sigma^2], \qquad (5)$$

where $I_0(\cdot)$ denotes the modified Bessel function and N is the size of the DFT. The derivation of these PDFs assumes zero mean white Gaussian noise of variance σ^2 and a signal frequency line of amplitude A that is centered in the DFT cell. It is necessary to integrate these functions in certain of the HMM detector/trackers. Note that the function $P_2(r)$ is easily integrable in closed form, but that $P_1(r)$ is not. Further details are given in [3].

Description of HMM/A00

This measurement process does not use either decision or quantization steps. Therefore, no quantization threshold is used, i.e., D = 0. The measurement available to the detector/tracker at any given time t takes the form $z = (r_1, r_2, \ldots, r_n)$, where r_i is the measured amplitude of the i-th DFT cell in the tracking gate. There are two cases. If the signal state is the zero state, no signal is present in any of the n DFT cells of the gate. Since the measured amplitudes r_i in each cell are independent, we have

$$b_0(z) = \kappa \prod_{j=1}^{n} P_2(r_j), \tag{6}$$

where κ is a nonzero state independent scale constant to be determined. Similarly, if the signal state is $i \geq 1$, then a frequency line lies in the center of the i-th DFT cell. Because of the center cell assumption, the amplitude measurements are independent from cell to cell, so that

$$b_i(z) = \kappa \left[\prod_{\substack{j=1 \\ j \neq i}}^{n} P_2(r_j) \right] P_1(r_i). \tag{7}$$

We now choose κ so that $b_0(z) = 1$, that is, κ is the reciprocal of the product on the right hand side of equation (6). Therefore,

$$b_i(z) = \begin{cases} 1, & i = 0 \\ \dfrac{P_1(r_i)}{P_2(r_i)}, & i \geq 1. \end{cases} \tag{8}$$

It is interesting to note that the function $b_i(z)$ is a most powerful hypothesis test on measured amplitude r_i for "signal in state $i \geq 0$" *vs* "signal in state $i = 0$".

Description of HMM/A01

This measurement process does not use a decision step, but does use a quantization step. A threshold D > 0 defines the one bit quantization breakpoint. (The case D = 0 is clearly not useful in this application.) The measurement input to the detector/tracker at any given time t takes the form of a set:

$$z = \{\text{DFT gate cells whose amplitudes equal or exceed D}\}.$$

The measurement set z identifies indirectly those gate cells whose amplitudes do not exceed D. No other information about cell amplitude values is contained in z. If the measurement set z is empty, it contains the useful information that no cell in the gate exceeds D.

There are two cases. If the signal state is the zero state, no signal is present and, because of the cell amplitude independence assumptions,

$$b_0(z) = \kappa \left[\int_D^\infty P_2(r)\, dr \right]^{\#(z)} \left[\int_0^D P_2(r)\, dr \right]^{n - \#(z)},$$

where #(z) is the number of measurements in the set z. If z is empty, then #(z) = 0. The integral from D to ∞ is the probability that the amplitude in a cell exceeds the threshold D, while the integral from 0 to D is the probability that the amplitude is less than D. Suppose the signal state is $i \geq 1$. If $i \in z$, then

$$b_i(z) = \kappa \left[\int_D^\infty P_2(r)\, dr \right]^{\#(z) - 1} \left[\int_D^\infty P_1(r)\, dr \right] \left[\int_0^D P_2(r)\, dr \right]^{n - \#(z)}.$$

Alternatively, in the "missed detection" case, $i \notin z$ and

$$b_i(z) = \kappa \left[\int_D^\infty P_2(r)\, dr \right]^{\#(z)} \left[\int_0^D P_2(r)\, dr \right]^{n-\#(z)-1} \left[\int_0^D P_1(r)\, dr \right].$$

Setting κ so that $b_0(z) = 1$ gives

$$b_i(z) = \begin{cases} 1, & \text{if } i = 0 \\ \dfrac{\int_0^D P_1(r)\,dr}{\int_0^D P_2(r)\,dr}, & \text{if } i \notin z \\ \dfrac{\int_D^\infty P_1(r)\,dr}{\int_D^\infty P_2(r)\,dr}, & \text{if } i \in z. \end{cases} \qquad (9)$$

The function $b_i(z)$ is a most powerful hypothesis test on each of two measurements types, namely

(1) ambiguous detection (that is, $i \in z$), and

(2) missed detection (that is, $i \notin z$),

for "signal in state $i \geq 0$" *vs* "signal in state $i = 0$".

Description of HMM/A10

This measurement process uses a decision step, but not a quantization step. Because there is no quantization step, the threshold D is set to 0. (N.B. The paper [1] considers the general case $D \geq 0$ and concludes, on the basis of simulation, that $D = 0$ is optimal for detection performance.) The measurement available to the detector/tracker at any given time t takes the form

$z = \{$DFT gate cell $k \geq 1$ has amplitude r_k, and all gate amplitudes $\leq r_k\}$.

The measurement z contains the DFT cell index k, its amplitude r_k, and the information that no other cell in the gate has greater amplitude. If the signal state is zero, then

$$b_0(z) = \kappa \left[\int_0^{r_k} P_2(r)\, dr \right]^{n-1} P_2(r_k).$$

If the signal state is $i \geq 1$, and $i \neq k$ (the missed detection case), then

$$b_i(z) = \kappa \left[\int_0^{r_k} P_2(r)\, dr \right]^{n-2} \left[\int_0^{r_k} P_1(r)\, dr \right] P_2(r_k).$$

If $i = k$, however, then

$$b_i(z) = \kappa \left[\int_0^{r_k} P_2(r)\, dr \right]^{n-1} P_1(r_k).$$

Setting the scale factor κ so that $b_0(z) = 1$ gives the result

$$b_i(z) = \begin{cases} 1, & \text{if } i = 0 \\ \dfrac{P_1(r_k)}{P_2(r_k)}, & \text{if } i = k \\ \dfrac{\int_0^{r_k} P_1(r)\, dr}{\int_0^{r_k} P_2(r)\, dr}, & \text{if } i \neq k. \end{cases} \qquad (10)$$

The function $b_i(z)$ is a most powerful hypothesis test on the measurements types, namely

(1) correct detection (that is, $i = k$) with amplitude r_k and

(2) missed detection (that is, $i \neq k$) with amplitude r_k,

for "signal in state $i \geq 0$" *vs* "signal in state $i = 0$".

Description of HMM/A11

This measurement process uses both a decision and a quantization step. A quantization threshold D > 0 determines the one bit quantization breakpoint. The measurement available to the detector/tracker at any given time t is one of the following:

z_0 = {no DFT gate cell has amplitude exceeding D}

z_k = {DFT gate cell k has maximum amplitude and its amplitude ≥ D},

where k = 1, 2, ... , n. The measurements z_0 and z_k contain no information about the maximum amplitude other than the fact that it does or does not exceed the threshold D. Note that the measurement z_0 contains no information on the location of the gate cell of largest amplitude. If the signal state is zero, then for measurement z_0

$$b_0(z_0) = \kappa \left[\int_0^D P_2(r)\, dr \right]^n$$

and, for measurements z_k, $k \geq 1$,

$$b_0(z_k) = \kappa \int_D^\infty P_2(r) \left[\int_0^r P_2(\rho)\, d\rho \right]^{n-1} dr \equiv \frac{\kappa - b_0(z_0)}{n} .$$

The identity follows from a normalization argument (i.e., the sum over all $k \geq 0$ of $b_0(z_k)$ equals κ) and the fact that $b_0(z_k)$ is constant for $k \geq 1$. If the signal state is $i \neq 0$, then for measurement z_0

$$b_i(z_0) = \kappa \left[\int_0^D P_1(r)\, dr \right] \left[\int_0^D P_2(r)\, dr \right]^{n-1} .$$

If the measurement is z_1, then

$$b_1(z_1) = \kappa \int_D^{\infty} P_1(r) \left[\int_0^r P_2(\rho)\, d\rho \right]^{n-1} dr.$$

The function $P_2(r)$ is easily integrated, so

$$b_1(z_1) = \kappa \int_D^{\infty} P_1(r) \left[1 - \exp(-r^2 N/\sigma^2) \right]^{n-1} dr.$$

Thus, evaluating $b_1(z_1)$ requires numerical integration of a one dimensional integral. Finally, if the measurement is z_k, $k \notin \{0, 1\}$, then

$$b_1(z_k) = \kappa \int_D^{\infty} P_2(r) \left[\int_0^r P_1(\rho)\, d\rho \right] \left[\int_0^r P_2(\rho)\, d\rho \right]^{n-2} dr.$$

Normalization arguments give the equivalent expression

$$b_1(z_k) = \frac{\kappa - b_1(z_0) - b_1(z_1)}{n-1}.$$

The expressions above, with $\kappa = 1$, were first given in [3], and are easy to utilize in practice.

It is worthwhile rewriting the above equations in a form that exhibits their theoretical character. To this end, for each positive integer n, we define the random variable $\Omega_n = \max\{R_1, \ldots, R_n\}$, where the random variables R_i are independent and identically distributed, with common PDF given by $P_2(r)$. Conditioned on the event $\Omega_m \geq D$, the PDF of Ω_n is given by

$$Q_n(r) = P_2(r) \left[\int_0^r P_2(\rho)\, d\rho \right]^{n-1} \Big/ \int_D^\infty P_2(r) \left[\int_0^r P_2(\rho)\, d\rho \right]^{n-1} dr,$$

where the denominator is the conditioning term required to make $Q_n(r)$ a valid PDF. By substituting the definition (5) of $P_2(r)$ and performing the required integrations, an explicit formula for $Q_n(r)$ can be obtained, if desired. We now choose the scale factor κ so that $b_0(z_k) = 1$ for all $k \geq 0$. Utilizing the functions $Q_n(r)$ gives the result

$$b_i(z) = \begin{cases} 1, & \text{for } i = 0 \text{ and all } z \\ \dfrac{\int_0^D P_1(r)\, dr}{\int_0^D P_2(r)\, dr}, & \text{for } i \neq 0 \text{ and } z = z_0 \\ \displaystyle\int_D^\infty \frac{P_1(r)}{P_2(r)}\, Q_n(r)\, dr, & \text{for } i \neq 0 \text{ and } z = z_i \\ \displaystyle\int_D^\infty \frac{\int_0^r P_1(\rho)\, d\rho}{\int_0^r P_2(\rho)\, d\rho}\, Q_n(r)\, dr, & \text{for } i \neq 0 \text{ and } z \neq z_i. \end{cases} \tag{11}$$

The function $b_i(z)$ is a most powerful hypothesis test on the measurements $\{z_k\}$ for the "signal in state $i \geq 0$" *vs* "signal in state $i = 0$". Note that the last two expressions in (11) are expectations of likelihood ratios.

Concluding Remarks

The four detector/trackers presented here have a common HMM algorithmic structure that is easily implemented. The only computational difference between them lies in the calculation of the state conditional PDFs of the available measurements. In all four detector/trackers, however, these PDFs can be precomputed either as a list of all possible measurement outcome probabilities, or as a likelihood function lookup table with, say, spline interpolation. With sufficient attention to the PDF calculation details, all four detector/trackers should run at approximately the same speeds.

HMM/A00 should be the best of the detector/trackers since it makes use of all the available amplitude data. HMM/A10 and HMM/A11 can be used for tracking on spectrogram displays that compress the measured amplitude data in ways that are compatable with their measurement processes. Of these two, HMM/A10 should give better performance than HMM/A11 because HMM/A10 has more amplitude information available than HMM/A11. Finally, HMM/A01 may be useful in applications in which the true signal is mismatched to the signal model used here. This possibility deserves further study.

References

1. R. F. Barrett and R. L. Streit, "Automatic Detection of Frequency Modulated Spectral Lines," Proceedings of the Australian Symposium on Signal Processing and Its Applications, Adelaide, Australia, 17-19 April 1989.

2. A. K. Steele, R. L. Streit, and R.F. Barrett, "Nonlinear Frequency Line Tracking Algorithms," Proceedings of the Australian Symposium on Signal Processing and Its Applications, Adelaide, Australia, 17-19 April 1989.

3. R. L. Streit and R. F. Barrett, "Frequency Line Tracking Using Hidden Markov Models," IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-38 (1990), 586-598.

Estimation of Signal Amplitude And Background Noise Power In Hidden Markov Model Detector/Trackers

R. L. Streit

Abstract

Frequency line detector/trackers based on hidden Markov models (HMMs) are designed for optimum detection and tracking performance at a specified design signal-to-noise ratio (SNR). In practice, their performance is observed to be robust to mismatch between the design SNR and the true SNR, especially when the true SNR exceeds the design SNR. A natural way to improve performance further is to estimate true SNR in an attempt to match the design SNR of the HMM detector/tracker to the true SNR. This memorandum derives maximum likelihood estimates of signal and background noise power for a specific HMM detector/tracker (known as HMM/AOO), using the Baum-Welch reestimation, or EM, method. The estimates are derived by exploiting the intrinsic training capabilities of general HMMs, so the approach of this memorandum is not limited to the one HMM detector/tracker presented here. Different HMM detector/trackers will generally yield different estimators for signal amplitude and noise power.

INTRODUCTION

The papers [1, 2] are the first papers to discuss the inclusion of amplitude information into detector/trackers based on hidden Markov models (HMMs), but neither paper discusses the estimation of signal and noise powers. This memorandum gives a derivation of maximum likelihood estimates of signal amplitude and background noise power for the best of the four HMM detector/trackers described in [3], namely, HMM/A00.

Maximum likelihood estimates of signal amplitude and background noise power are derived from the likelihood structure imposed on the measured data by the detector/tracker HMM/A00. The likelihood structure of HMM/A00 is highly non-Gaussian in nature because it arises from the "hidden" Markov chain and the specialized measurement likelihood functions appropriate to the frequency line tracking application. The methods used here for HMM/A00 can, in principle, be applied to derive signal and noise power estimates for the other HMM detector/trackers presented in [3]; however, *estimators for these other trackers are not presented here.*

The detector/tracker HMM/A00 uses a fixed frequency gate, that is, a fixed contiguous subset of size $n \geq 1$ of the full set of DFT (discrete Fourier transform) frequency cells. The DFT size is $N \geq n$ and cells of the fixed gate are indexed, for convenience, 1, 2, ... , n. The HMM/A00 measurement process utilizes all the available amplitude information, that is, the output of the measurement process at time t is the vector

$$z_t = (r_{1t}, r_{2t}, \ldots , r_{nt}), \qquad (1)$$

where r_{it} is the measured DFT amplitude in gate cell i at time t.

The detector/tracker HMM/A00 has two types of states -- one corresponding to the "signal absent" hypothesis and others

corresponding to "signal present" hypotheses. (If either type of state is not used, then the HMM cannot be described as a detector/tracker.) The "signal absent" hypothesis is indexed 0 and called the zero state. The "signal present" hypotheses are numbered from 1 to n and correspond to signal frequency lines centered in the n DFT cells of the gate.

BACKGROUND ON THE DETECTOR/TRACKER HMM/AOO

Let $Z = (z_1, z_2, \ldots, z_T)$ denote a measurement sequence of length $T \geq 1$ that is input to the n+1 state detector/tracker HMM/AOO, where the measurements z_t are defined by (1). The forward algorithm computes the vectors $\alpha_t \in \mathbb{R}^{n+1}$ by means of an algorithm that is easily stated in terms of matrix products. Let $'$ denote matrix transposition. Then the forward algorithm is defined by

$$\begin{aligned} \alpha_1 &= B(z_1)\,\pi \\ \alpha_{t+1} &= B(z_{t+1})\,A'\,\alpha_t, \qquad t = 1, \ldots, T-1, \end{aligned} \tag{2}$$

where $\pi \in \mathbb{R}^{n+1}$ is the initial state probability vector, $A = [a_{ij}] \in \mathbb{R}^{(n+1)\times(n+1)}$ is the state transition probability matrix, and, for an arbitrary measurement z, the matrix $B(z) \in \mathbb{R}^{(n+1)\times(n+1)}$ is defined by

$$B(z) = \text{Diag}\,[b_0(z),\ b_1(z),\ \ldots,\ b_n(z)],$$

and where $b_i(z)$ denotes the likelihood function of the measurement z conditioned on the signal state i, i = 0, 1, ..., n. Similarly, the backward algorithm computes the vectors $\beta_t \in \mathbb{R}^{n+1}$ by means of an algorithm that is equivalent to the matrix product

$$\begin{aligned} \beta_T &= (\,1\ \ 1\ \cdots\ 1\,)' \in \mathbb{R}^{n+1} \\ \beta_t &= A\,B(z_{t+1})\,\beta_{t+1}, \qquad t = T-1, \ldots, 1. \end{aligned} \tag{3}$$

Probabilistic interpretations for the vectors α_t and β_t are given in [4]. The continuous detector/tracker outputs are derived (see [4] for further details) from the probability field, denoted by $\gamma_t(i)$. The numerical value of $\gamma_t(i)$ is the probability the signal occupies state i at time t, conditioned on all the available measurement sequence Z. It is defined in terms of the components $\{\alpha_t(i)\}$ and $\{\beta_t(i)\}$ of the vectors α_t and β_t by

$$\gamma_t(i) = \alpha_t(i)\,\beta_t(i) \left[\sum_{k=0}^{n} \alpha_t(k)\,\beta_t(k) \right]^{-1}. \qquad (4)$$

In equation (4), t ranges from 1 to T, and i ranges from 0 to n. Although it is not immediately clear, it can be shown that

$$\sum_{i=0}^{n} \gamma_t(i) = 1$$

for each time t. The name "probability field" for $\gamma_t(i)$ was first used in [7], where its image enhancement property was first described.

A statistic closely related to the probability field is the "gate occupancy probability", denoted G_{op}. The G_{op} is defined by

$$G_{op} = \frac{1}{T} \sum_{t=1}^{T} [1 - \gamma_t(0)] = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{n} \gamma_t(i). \qquad (5)$$

The zero state is excluded from the sums (5), so the statistic G_{op} is conditioned on signal presence somewhere within the gate. An important interpretation of the G_{op} is that it represents the fraction of time for which a signal is present within the gate. The G_{op} is potentially useful as a detection statistic.

From equations (2)-(4), it is clear that measurements affect the detector/tracker output only via the state likelihood functions $b_i(z)$. HMM/AOO is otherwise blind to the measurement. The likelihood functions $b_i(z)$ are derived in terms of the following two probability density functions (PDFs) of the measured amplitude r in a DFT cell. Define the noise parameter $\rho = 1/\sigma^2$, where σ^2 is the power (variance) of the zero mean white Gaussian background noise. The PDF of r conditioned on the hypothesis "signal in state $i \neq 0$" is the Rician PDF defined by

$$P_1(r|A,\rho) = (2rN\rho)\, I_0(ArN\rho)\, \exp[-N\rho(r^2 + A^2/4)], \qquad (6)$$

where $I_\nu(\cdot)$ denotes the modified Bessel function of order ν, and N is the size of the DFT. The derivation of this PDF assumes the signal frequency line of amplitude A is centered in the DFT cell. The PDF of r conditioned on the hypothesis "signal in state i = 0" is the Rayleigh PDF defined by

$$P_2(r|\rho) = (2rN\rho)\, \exp[-Nr^2\rho]. \qquad (7)$$

Note that $P_2(r|\rho) \equiv P_1(r|0,\rho)$. Plots of $P_1(r|A,\rho)$ for several values of A for fixed noise power $N\rho = 1$ are given in Figure 1.

As is shown in [3], for the measurement z_t, the state conditional likelihood functions take the form

$$b_i(z_t) \equiv b_i(z_t|A,\rho) = \begin{cases} \kappa_t(\rho) \\ \dfrac{P_1(r_{it}|A,\rho)}{P_2(r_{it}|A,\rho)}\ \kappa_t(\rho), \end{cases} \qquad (8)$$

where the "scaling" function $\kappa_t(\rho)$ is given by

$$\kappa_t(\rho) = \prod_{k=1}^{n} P_2(r_{kt}|A,\rho). \qquad (9)$$

The expressions (8) and (9) can be simplified if desired by substituting the equations (6) and (7).

Note that the symbol A has been used to denote both the Markov chain transition probability matrix and also the signal amplitude. Both uses of the symbol A are somewhat conventional, and it is inconvenient to change either. This abuse of notation should cause no confusion.

DERIVATION OF MAXIMUM LIKELIHOOD ESTIMATION EQUATIONS

Estimates of signal amplitude A and background noise power σ^2 are derived using the EM (Expectation-Maximization) method [5]. When the EM method is applied to HMM's, the EM method is more commonly known as the Baum-Welch reestimation algorithm [6]. The derivation in this Appendix is a variant of Baum-Welch that accomodates continuously variable parameters and the specialized likelihood structure of the frequency line estimation problem. The discussion here assumes familiarity with the fundamental HMM detector/tracker concepts as described in [4] and with the definition of HMM/AOO as described in [3]. To the extent possible, the notation of [4] is used here.

For times t = 1, 2, ... , T, the measurement $z_t \in Z$ is given by (1). The "missing data" in the sense of the EM method is the state sequence I of the Markov chain. Let

$$I = \{i(1), i(2), \ldots, i(T)\} \in \mathcal{I},$$

where the index i(t) denotes the true signal state at time t, and $\mathcal{I}$ denotes the set of all possible state sequences. (N.B. We use the notation i(t) instead of the more common i_t to avoid the use of subscripted subscripts in the sequel.) The n+1 Markov chain states are numbered consecutively from 0 to n, so $0 \le i(t) \le n$ for all time t. Let $Z' = Z \cup I$. The PDF of Z', conditioned on the detector/tracker HMM/AOO, is parameterized by signal amplitude A and white Gaussian background noise parameter $\rho = 1/\sigma^2$ and is given by

$$\mathcal{F}(Z'|A,\rho) = \pi_{i(1)}\, b_{i(1)}(z_1|A,\rho) \prod_{t=2}^{T} a_{i(t-1),i(t)}\, b_{i(t)}(z_t|A,\rho). \quad (10)$$

The transition probability matrix $A = [a_{ij}] \in \mathbb{R}^{(n+1)\times(n+1)}$ and the initial state probability vector $\pi = [\pi_i] \in \mathbb{R}^{n+1}$ define the Markov

chain, and are specified *a priori* by the detector/tracker HMM/AOO, while the likelihood function $b_k(z_t|A,\rho)$ is given explicitly by equations (8)-(9) above. As required by the EM method, PDF induced on the set $\mathcal{I}$ of all state sequences is derived from Bayes Theorem and is given by

$$\mathcal{K}(I|Z,A,\rho) = \frac{\mathcal{F}(Z'|A,\rho)}{\mathcal{L}(Z|A,\rho)}, \tag{11}$$

where, from the HMM likelihood structure,

$$\mathcal{L}(Z|A,\rho) = \sum_{I\in\mathcal{I}} \mathcal{F}(Z'|A,\rho). \tag{12}$$

For future use, note that

$$\sum_{I\in\mathcal{I}} \mathcal{K}(I|Z,A,\rho) = 1 \tag{13}$$

$$\sum_{I\in\mathcal{I}\setminus i(t)} \mathcal{K}(I|Z,A,\rho) = \gamma_t(i(t)), \tag{14}$$

where $\gamma_t \equiv \gamma_t(A,\rho) \in \mathbb{R}^{n+1}$ is the likelihood vector computed from equation (4). In equation (14), the notation $I\in\mathcal{I}\setminus i(t)$ means that the sum is over all state indices I except i(t). The proof of (13) is obvious, while the proof of (14) requires examining the probabilistic interpretations of the forward and backward likelihood vectors, denoted by $\alpha_t \in \mathbb{R}^{n+1}$ and $\beta_t \in \mathbb{R}^{n+1}$, respectively, and computed from equations (2)-(3). The dependence of α_t and β_t on the parameters A and ρ is implicit in the notation. The result (14) is essentially a corollary of Baum's Theorem stating that, for all t,

$$\mathcal{L}(Z|A,\rho) = \sum_{i(t)=0}^{n} \alpha_t(i(t))\,\beta_t(i(t)). \tag{15}$$

Further details of the proof of (14) are omitted.

The expectation step of the EM method is defined by

$$Q(A,\rho|A',\rho') \equiv E[\ \log \mathcal{F}(Z'|A,\rho)\ |\ Z,A',\rho'\ ]$$

$$\equiv \sum_{I\in\mathcal{I}} \mathcal{K}(I|Z,A',\rho')\ \log \mathcal{F}(Z'|Z,A,\rho). \qquad (16)$$

In (16), the parameters A' and ρ' are assumed given, and the goal is to maximize Q by choice of of A and ρ. Taking the logarithm of $\mathcal{F}(Z'|A,\rho)$ in (10) and substituting into (16) gives

$$Q(A,\rho|A',\rho') = c + \sum_{I\in\mathcal{I}} \sum_{t=1}^{T} \mathcal{K}(I|Z,A',\rho')\ \log b_{i(t)}(z_t|A,\rho), \quad (17)$$

where c denotes a constant independent of the parameters A and ρ. Isolating the sum over i(t) in equation (16), and using the result (13), gives

$$Q(A,\rho|A',\rho') = c + \sum_{t=1}^{T} \sum_{i(t)=0}^{n} \log b_{i(t)}(z_t|A,\rho) \sum_{I\in\mathcal{I}\setminus i(t)} \mathcal{K}(I|Z,A',\rho')$$

$$= c + \sum_{t=1}^{T} \sum_{i=0}^{n} \gamma'_t(i)\ \log b_i(z_t|A,\rho), \qquad (18)$$

where $\gamma'_t \equiv \gamma_t(A',\rho') \in R^{n+1}$ is the state occupancy likelihood vector defined by equation (4) for the given parameters A' and ρ'. Substituting expressions (8)-(9) into equation (18) gives the function $Q(A,\rho|A',\rho')$ as an explicit function of the parameters A and ρ.

A stationary point of $Q(A,\rho|A',\rho')$ with respect to independent variables A and ρ satisfies the equations

$$\frac{\partial Q}{\partial A} = \sum_{t=1}^{T} \sum_{i=0}^{n} \gamma'_t(i)\ \frac{\partial}{\partial A}\ \log b_i(z_t|A,\rho) = 0 \qquad (19)$$

$$\frac{\partial Q}{\partial \rho} = \sum_{t=1}^{T} \sum_{i=0}^{n} \gamma'_t(i)\ \frac{\partial}{\partial \rho}\ \log b_i(z_t|A,\rho) = 0. \qquad (20)$$

The required partial derivatives in (19)-(20) can be computed from (8)-(9). Using the fact that $I_0'(\cdot) = I_1(\cdot)$, the required partial derivatives are

$$\frac{\partial}{\partial A} \log \kappa_t(\rho) = 0,$$

$$\frac{\partial}{\partial \rho} \log \kappa_t(\rho) = \frac{n}{\rho}\left[1 - \rho N \overline{r_t^2} \right],$$

$$\frac{\partial}{\partial A} \log \left[\frac{P_1(z_t|A,\rho)}{P_2(z_t|A,\rho)} \kappa_t(\rho) \right] = \frac{N\rho}{2}\left[2 r_{it} \frac{I_1(Ar_{it}N\rho)}{I_0(Ar_{it}N\rho)} - A \right],$$

$$\frac{\partial}{\partial \rho} \log \left[\frac{P_1(z_t|A,\rho)}{P_2(z_t|A,\rho)} \kappa_t(\rho) \right] = \frac{1}{\rho}\left[Ar_{it}N\rho \frac{I_1(Ar_{it}N\rho)}{I_0(Ar_{it}N\rho)} - \frac{NA^2\rho}{4} + n - nN\rho \overline{r_t^2} \right],$$

where

$$\overline{r_t^2} = \frac{1}{n} \sum_{i=1}^{n} r_{it}^2 .$$

For later use, we define

$$\overline{r^2} = \frac{1}{T} \sum_{t=1}^{T} \overline{r_t^2} = \frac{1}{nT} \sum_{t=1}^{T} \sum_{n=1}^{n} r_{it}^2. \qquad (21)$$

Substituting these partial derivatives into the first necessary condition (19) and simplifying gives the nonlinear equation

$$A = 2 \sum_{t=1}^{T} \sum_{i=1}^{n} \psi_{it}' \, r_{it} \frac{I_1(Ar_{it}N\rho)}{I_0(Ar_{it}N\rho)}, \qquad (22)$$

where the coefficients $\psi'_{it} \equiv \psi'_{it}(A',\rho')$ are defined by

$$\psi'_{it} = \frac{\gamma_t(i|A',\rho')}{T\, G'_{op}} . \qquad (23)$$

In equation (22), $G'_{op} \equiv G_{op}(A',\rho')$ denotes the gate occupancy probability defined by equation (5) for A' and ρ'. The definitions (23) and (5) imply that $\psi'_{it} \geq 0$ and that

$$\sum_{t=1}^{T} \sum_{i=1}^{n} \psi'_{it} = 1. \qquad (24)$$

The coefficients $\{\psi'_{it}\}$ are independent of the unknown parameters A and ρ. Equation (22) is one of the two coupled estimation equations. The other equation is derived by substituting the necessary partial derivatives into (20) and simplifying terms. This gives

$$\frac{\sigma^2}{N} \equiv \frac{1}{N\rho} = \overline{r^2} - \frac{G_{op}(A',\rho')}{n} \frac{A^2}{4} . \qquad (25)$$

Equation (25) is the second estimation equation. Substituting (25) into (22) gives a single nonlinear equation in the signal amplitude A, namely,

$$A = 2 \sum_{t=1}^{T} \sum_{i=1}^{n} \psi'_{it}\, r_{it} \frac{I_1(Ar_{it} / (\overline{r^2} - G_{op}A^2/(4n)))}{I_0(Ar_{it} / (\overline{r^2} - G_{op}A^2/(4n)))} . \qquad (26)$$

The weights $\{\psi'_{it}\}$ are not updated while solving equation (26) numerically for the signal amplitude A; therefore, equation (26) can be solved by any suitable one dimensional iteration method such as bisection, Newton-Raphson, etc. Care should be exercised to account for the possibility of multiple solutions of (26). After A is found, the noise parameter ρ is computed from equation (25).

The EM method requires that the global maximum of Q be found. This aspect of the derivation is necessary if the EM algorithm is to be proved mathematically convergent. The proof is omitted.

The maximum likelihood estimates are computed by an "inner-outer" iteration. The EM algorithm is the outer iteration, while the inner iteration is, say, a bisection or gradient descent method for solving the necessary conditions (22) and (25). The EM algorithm is explicitly stated as follows:

EM Algorithm for Maximum Likelihood Estimates A_{ML} and ρ_{ML}:

1. Initialize: $A(0) > 0$, $\rho(0) > 0$, $k = 0$.
2. Set $k = k + 1$.
3. Let $A'=A(k-1)$ and $\rho'=\rho(k-1)$. Then:
   a. Using equations (2)-(4), compute the vectors $\{\alpha_t, \beta_t, \gamma_t\}_1^T$.
   b. Solve equations (22) and (25) for $A(k)$ and $\rho(k)$.
4. Test for convergence: Done?
   NO: Loop to step 2.
   YES: Set $A_{ML} = A(k)$ and $\rho_{ML} = \rho(k)$.

THE SPECIAL CASE OF ONE DFT CELL IN THE GATE

It is interesting to examine the estimation equations in the special case where no tracking is required. If the Markov chain has one state corresponding to a DFT cell and no zero state, the signal must always be present in the only state of the Markov chain. Thus, $\gamma_t(1) \equiv 1$ for all time t and for all choices of signal amplitude A and noise parameter ρ. Similarly, $G_{op} \equiv 1$. The EM algorithm is not iterative because there is no "missing data" in this special case. (N.B. The EM algorithm is equivalent to maximum likelihood estimation when there is no "missing data.") Substituting $\gamma_t \equiv G_{op} \equiv 1$ and $\psi'_{it} \equiv 1/T$ into the estimation equations (22) and (25) gives

$$A = \frac{2}{T} \sum_{t=1}^{T} r_t \frac{I_1(Ar_tN\rho)}{I_0(Ar_tN\rho)} \tag{27}$$

$$\frac{\sigma^2}{N} \equiv \frac{1}{N\rho} = \overline{r^2} - \frac{A^2}{4}, \tag{28}$$

where

$$\overline{r^2} = \frac{1}{T} \sum_{t=1}^{T} r_t^2, \tag{29}$$

and where $r_t = r_{1t}$ is the amplitude of the only DFT cell in the gate (see equation (1)). Solving for the noise parameter ρ in (28) and substituting into (27) gives a single nonlinear equation to be solved numerically for the signal amplitude A. Explicitly, this equation is

$$A = \frac{2}{T} \sum_{t=1}^{T} r_t \frac{I_1(Ar_t/(\overline{r^2} - A^2/4))}{I_0(Ar_t/(\overline{r^2} - A^2/4))} . \tag{30}$$

Equation (30) is a special case of equation (26), and the remarks made

after (26) concerning numerical solution apply to (30) also. The noise parameter ρ is computed from (28) after A has been computed from (30).

The estimation equations (27)-(28) can be derived directly by maximum likelihood methods without using the HMM/AOO tracking structure. The measurement sequence $Z = (r_1, r_2, \ldots, r_T)$ is a sequence of independent realizations of a random variable whose PDF is the Rician $P_1(r|A,\rho)$ given by (6). Therefore, the posterior likelihood function of Z is

$$\mathcal{L}(Z|A,\rho) = \prod_{t=1}^{T} P_1(r_t|A,\rho).$$

Omitting scale factors independent of A and ρ, the posterior likelihood function of Z is written more simply as

$$\mathcal{L}(Z|A,\rho) = \rho^T \exp(-N\rho TA^2/4) \exp\left[-N\rho T\, \overline{r^2}\right] \prod_{t=1}^{T} I_0(AN\rho r_t). \quad (31)$$

Differentiating $\mathcal{L}$ with respect to A and ρ, setting the result to zero, and simplifying gives the necessary conditions (27)-(28) for maximum likelihood estimates of A and ρ.

Equations (27) and (28) are natural to the frequency line SNR estimation problem. Related results for optimal <u>detection</u> problems have been discussed elsewhere. Helstrom [9, Chapter VII, Section 1(b)] examines the optimum likelihood-ratio receiver for the sequential frequency line detection problem, and gives several references to earlier work.

CONCLUDING REMARKS

The estimation equation (25) for noise power in a bin, σ^2/N, is interpreted as the sample mean of the square of all the measured amplitudes in Z, corrected by a term representing the signal power smeared over the entire gate and over the full time history T. This interpretation of the correction term is reasonable because the division by n smears the signal over the DFT cells in the gate, and because the multiplication by G_{op} smears the signal over time. Since the G_{op} estimates the fraction of the time interval T during which the signal occupies the gate, the estimation equation (22) for signal amplitude A is interpreted as an estimate of signal amplitude *when signal is present in the gate*. Without the G_{op} factor, the estimate of A would be biased low by signal absence.

The estimation equation (22) for A is a weighted average of the measured amplitudes $\{r_{it}\}$ over the full measurement sequence Z. The weight applied to an amplitude r_{it} is interesting because it is a product of a factor representing the "global" properties of the detector/tracker HMM/A00 and a factor representing the "local" statistical properties of the measured amplitude in a DFT cell. The global properties are those associated with the probability field $\{\gamma_t(i)\}$, and must be computed by the recursions (2)-(4). The local properties are those associated with the ratio of Bessel functions (discussed in the Appendix) and are computed easily from knowledge of the measured amplitude in each DFT cell.

An alternative to solving the estimation equations (22) and (25) is to use them simply as estimators for A and σ^2/N. If successful, this would obviate the need to solve the equations numerically. Another alternative is to estimate the noise power using data in DFT cells that presumably contain no signal. If this is done, the estimation equation (25) can be eliminated, although the equation (22) must still be solved. In any event, further study of these equations is merited.

APPENDIX

The function I(x), defined by the Bessel function ratio

$$I(x) \equiv \frac{I_1(x)}{I_0(x)}, \tag{A.1}$$

is of independent interest. I(x) is an odd function because $I_1(x)$ is odd and $I_0(x)$ is even. The function I(x) is plotted in Figure 2 for $x \geq 0$. Using the identities $I_0'(x) = I_1(x)$ and $I_1'(x) = I_0(x) - I_1(x)/x$, the derivative of I(x) is given explicitly by

$$I'(x) = 1 - \frac{1}{x} I(x) - I^2(x). \tag{A.2}$$

I'(x) is plotted in Figure 3. The apparent singularity of I'(x) at x = 0 is removable since $I(x) = x/2$ asymptotically as $x \rightarrow 0$; hence, $I'(0) = 1/2$. From Figures 2 and 3, it is clear that, for $x \geq 0$, I(x) is a cumulative distribution function (CDF) and that I'(x) is a PDF. A formal proof of this fact (due to A. H. Nuttall) is given in this Appendix. From the identity [8, equation (9.6.19)]

$$I_0(x) = \frac{1}{\pi} \int_0^{\pi} \exp(x \cos \theta) \, d\theta, \tag{A.3}$$

it follows by differentiation that

$$I_0'(x) = I_1(x) = \frac{1}{\pi} \int_0^{\pi} \cos \theta \, \exp(x \cos \theta) \, d\theta \tag{A.4}$$

$$I_0''(x) = I_1'(x) = \frac{1}{\pi} \int_0^{\pi} \cos^2 \theta \, \exp(x \cos \theta) \, d\theta. \tag{A.5}$$

Differentiating I(x) in (A.1) gives $I'(x) = N(x)/I_0^2(x)$, where the function $N(x) = I_0(x) I_1'(x) - I_0'(x) I_1(x)$. Substituting $I_0'(x) = I_1(x)$, and the identities (A.3) - (A.5) into the expression for N(x) gives the result

$$\pi^2 N(x) = \left[\int_0^{\pi} \exp(x \cos\theta) \, d\theta \right] \left[\int_0^{\pi} \cos^2 x \ \exp(x \cos\theta) \, d\theta \right] - \left[\int_0^{\pi} \cos\theta \ \exp(x \cos\theta) \, d\theta \right]^2 . \qquad (A.6)$$

Fix $x \geq 0$. Substituting into (A.6) the functions

$$a(\theta) \equiv \exp[(x \cos\theta)/2]$$
$$b(\theta) \equiv \cos\theta \ \exp[(x \cos\theta)/2]$$

and applying the Cauchy-Schwartz inequality shows that $N(x) \geq 0$. Equality holds in the Cauchy-Schwartz inequality if and only if, for some constant c, $a(\theta) \equiv c\, b(\theta)$ for all θ in the interval $[0, \pi]$. Since $a(\theta) \not\equiv c\, b(\theta)$, $N(x)$ cannot equal 0. It follows that $I'(x) = N(x)/I_0^2(x) > 0$. Therefore, $I(x)$ is strictly increasing for $x \geq 0$. From the asymptotic result, valid for fixed ν, [8, equation (9.7.1)]

$$I_\nu(x) = \frac{e^x}{\sqrt{2\pi x}} \left[1 - \frac{4\nu^2 - 1}{8x} + O(x^{-2}) \right], \quad x \to \infty, \qquad (A.7)$$

it follows that $I(x) \to 1$ as $x \to \infty$. Therefore, $I(x)$ is a CDF.

The mean value of $I'(x)$ does not exist because, like the Cauchy density, $I'(x)$ has such a heavy tail that its first moment is infinite. The heavy tail is evident from Figure 2. A proof of the unboundedness of the mean value follows from the asymptotic formula (A.7).

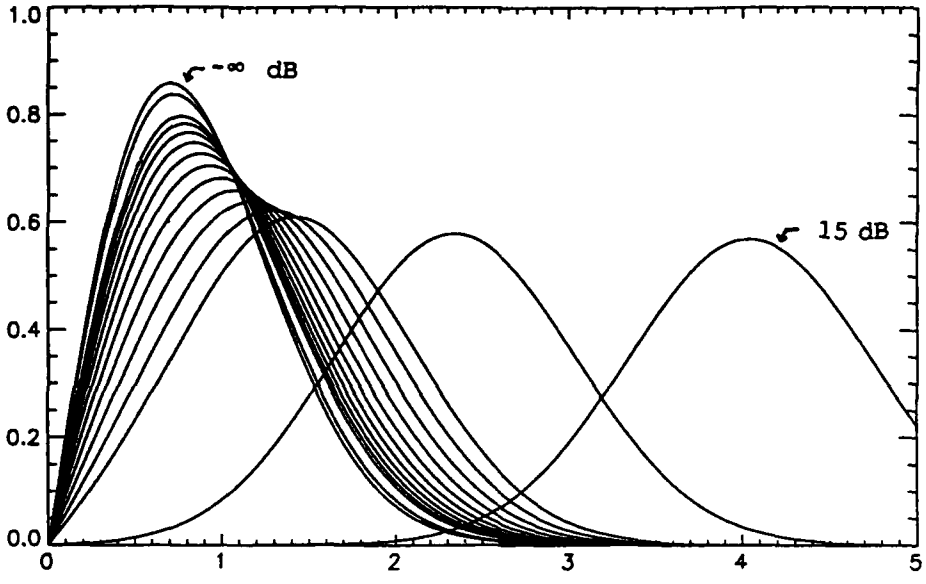


Figure 1. Plots of $P_1(r|A,\rho)$ for $N\rho=1$ and values of A corresponding to the stated values of signal-to-noise ratio:

Signal power =	-∞	dB,	A =	0
Signal power =	-10	dB,	A =	.447214
Signal power =	-5	dB,	A =	.795271
Signal power =	-4	dB,	A =	.892308
Signal power =	-3	dB,	A =	1.00119
Signal power =	-2	dB,	A =	1.12335
Signal power =	-1	dB,	A =	1.26042
Signal power =	0	dB,	A =	1.41421
Signal power =	1	dB,	A =	1.58677
Signal power =	2	dB,	A =	1.78039
Signal power =	3	dB,	A =	1.99763
Signal power =	4	dB,	A =	2.24138
Signal power =	5	dB,	A =	2.51487
Signal power =	10	dB,	A =	4.47214
Signal power =	15	dB,	A =	7.95271

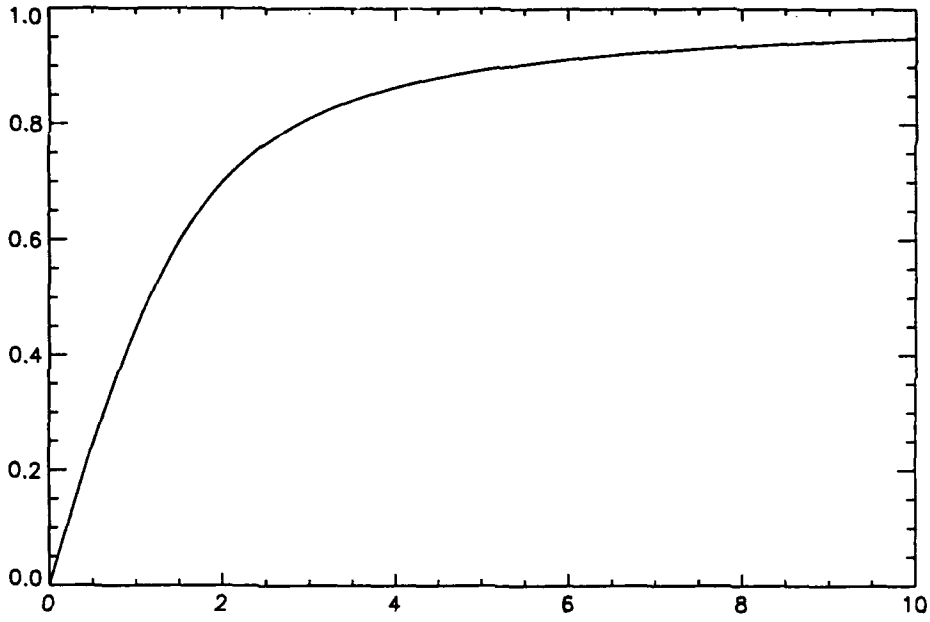


Figure 2. Plot of the CDF I(x), defined by equation (A.1).

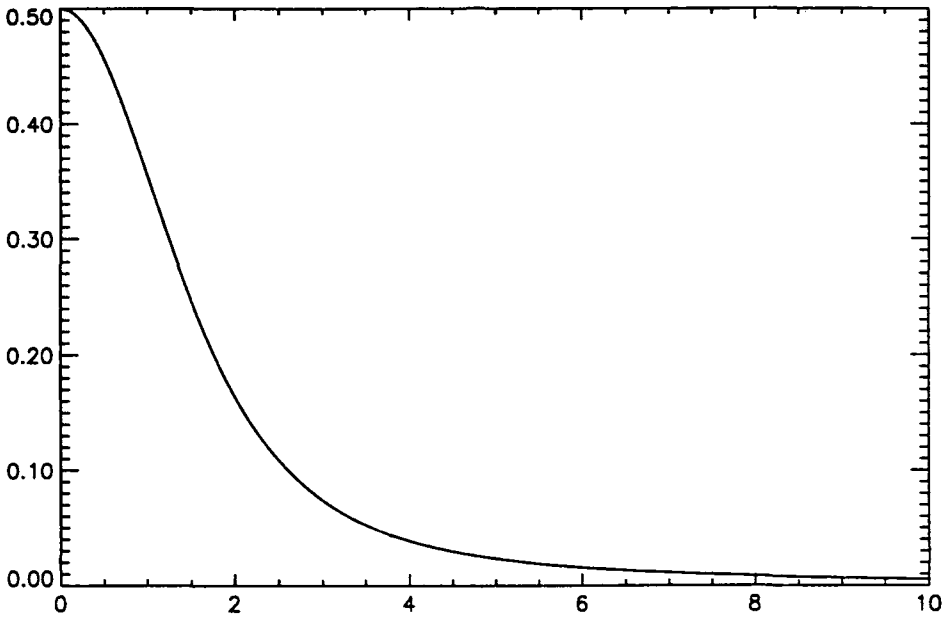


Figure 3. Plot of the PDF I'(x), defined by equation (A.2).

REFERENCES

1. R. F. Barrett and R. L. Streit, "Automatic Detection of Frequency Modulated Spectral Lines," Proceedings of the Australian Symposium on Signal Processing and Its Applications, Adelaide, Australia, 17-19 April 1989.

2. A. K. Steele, R. L. Streit, and R.F. Barrett, "Nonlinear Frequency Line Tracking Algorithms," Proceedings of the Australian Symposium on Signal Processing and Its Applications, Adelaide, Australia, 17-19 April 1989.

3. R. L. Streit, "Frequency Line Detector/Tracker Using Hidden Markov Models With Amplitude Information," NUSC Technical Memorandum No. 911143, 20 June 1991.

4. R. L. Streit and R. F. Barrett, "Frequency Line Tracking Using Hidden Markov Models," IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-38 (1990), 586-598.

5. A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," Journal of the Royal Statistical Society, Series B, 39(1977), 1-38.

6. L. E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Chains," in Inequalities III, O. Shisha (Editor), Academic Press, New York, 1972, 1-8.

7. J. L. Muñoz and R. L. Streit, "Connection Machine Implementation of Hidden Markov Models for Frequency Line Tracking," in Very Large Scale Computation in the 21st Century (J. P. Mesirov, Editor), Society for Industrial and Applied Mathematics, Philadelphia, 1991, 204-217.

8. Handbook of Mathematical Functions, M. Abramowitz and I. R. Stegun (Editors), AMS 55, National Bureau of Standards, 1964 (Tenth printing 1972).

9. C. W. Helstrom, Statistical Theory of Signal Detection, Second Edition, New York, Pergamon Press, 1968.

Automatic Detection Of Frequency Modulated Spectral Lines

R. F. Barrett and R. L. Streit

Automatic Detection of Frequency Modulated Spectral Lines

R.F. BARRETT
Maritime Systems Div., Weapons Systems Research Laboratory, Salisbury, S.A.

R.L. STREIT
Naval Underwater Systems Center, New London, Connecticut, U.S.A.

SUMMARY Three methods for the detection of narrowband signals of unstable frequency embedded in a white Gaussian noise background are investigated. The first method, known as the Maximum-Power-Track detector, divides the frequency domain into gates containing a fixed number of FFT cells. The maximum spectral power within the gate is integrated over time and used for detection purposes. The MPT detector is fast and easy to implement. The other detectors involve an extension of the Hidden Markov Model (HMM) frequency tracker developed earlier by Streit and Barrett (1988) to allow for detection. The HMM trackers have been extended to allow for the inclusion of amplitude information in the input measurement sequence. It is found that the MPT detector is a simple and effective detector, and that the extended HMM detectors represent a significant improvement, albeit at the cost of increased computational complexity.

1. INTRODUCTION

The detection of stable frequency lines embedded in white Gaussian noise is a well-studied problem in detection theory. In this case, the optimal detector is the conventional integrated power detector. In the usual implementation, the sampled time series is divided into non-overlapped data blocks, and Fast Fourier Transforms (FFTs) of each block are computed. The size of the blocks is chosen to give the desired frequency resolution for the problem under study. It is assumed that the signal frequency is stable enough so that the signal remains entirely within the same FFT frequency cell from data block to data block. The spectral power within each FFT cell is then summed over all the data blocks, and a detection is registered whenever the integrated power in one cell exceeds a predetermined detection threshold. This threshold is set by considering the case when noise only is present. The Probability Distribution Function (PDF) of the noise power is assumed to be Gaussian. When the noise is not white Gaussian, it is necessary to first "pre-whiten" by filtering the noise background so that the filtered noise has the desired statistical characteristics (e.g., unit variance, white).

In the case when the frequency of the signal to be detected is unstable (i.e., the signal frequency wanders over several FFT frequency cells), the detection performance of the conventional integrated power detector becomes degraded. Instead of the spectral power being concentrated in one frequency cell from data block to data block, it becomes smeared over several cells in the frequency "gate". The integrated signal power in individual cells within the gate may then not exceed the detection threshold, and a missed detection will ensue.

In this paper, we compare three methods for the detection of unstable frequencies embedded in white Gaussian noise. The first method, discussed in Section 2, is known as the Maximum-Power-Track (MPT) detector, and represents a simple but effective extension of the conventional detector. The other two methods are based on the Hidden Markov Model (HMM). They are extensions of an earlier work on the use of the HMM for frequency line tracking presented by Streit and Barrett (1988), and are discussed in Section 3. A comparison of the results from the three approaches is presented in Section 4.

2. THE MAXIMUM-POWER-TRACK DETECTOR

The MPT detector represents a straightforward attempt to enhance the poor performance of the conventional detector for fluctuating signal frequencies. A gate of frequency cells is defined, with the number M of cells in the gate chosen large enough to encompass the extremes of the frequency meanders. For each of the T blocks of data, the cell containing the maximum spectral power is selected from all cells within the gate. A measurement threshold is set for the spectral power. If the power in the selected cell exceeds this measurement threshold, the cell's power is added to a sum that accumulates the total power in the selected cells over the window of T data blocks. The accumulated total power is designated the "gate power". A detection is registered whenever the gate power exceeds a detection threshold. This threshold is set by estimating the gate power in gates containing only noise, in the same manner as for a conventional detector. The conventional detector represents a special case of the MPT detector with $M = 1$.

The simplicity of the MPT detector enables a statistical analysis to be carried out of its performance. In the analysis, we assume that at the time the frequency is computed for each data block L, a signal is present with amplitude A in the centre of the m-th FFT frequency cell. The background noise is assumed to be white, zero-mean Gaussian, with a broadband noise power of σ^2. The signal frequency lies in different FFT cells at different times (i.e., m depends on L). The FFT data blocks are of size N.

Under these assumptions, the PDF F(p) of the spectral power contribution p of each FFT data block to the total gate power is given by:

$$F(p) = \delta(p)\left[\int_0^{D^2} P_2(e)\,de\right]^{n-1}\int_0^{D^2} P_1(e)\,de \qquad (1a)$$

$$\text{for } p \le D^2 ,$$

$$= P_1(p)\left[\int_0^{p} P_2(e)\,de\right]^{n-1}$$

$$+ (n-1)\,P_2(p)\left[\int_0^{p} P_1(e)\,de\right]\left[\int_0^{p} P_2(e)\,de\right]^{n-2} \qquad (1b)$$

$$\text{for } p > D^2 ,$$

where D^2 is the measurement power threshold, and $\delta(x)$ is Dirac's delta function.

In Equation 1, $P_1(e)$ is the PDF for the spectral power in the cell containing the signal, and $P_2(e)$ is the corresponding PDF in cells containing no signal. The PDFs, $P_1(e)$ and $P_2(e)$, are given by:

$$P_1(e) = \left[\frac{N}{\sigma^2}\right] I_0\left[\frac{AN\sqrt{e}}{\sigma^2}\right] \exp\left[\frac{-N(4e + A^2)}{4\sigma^2}\right] \qquad (2a)$$

and

$$P_2(e) = \left[\frac{N}{\sigma^2}\right] \exp\left[\frac{-Ne}{\sigma^2}\right] \qquad (2b)$$

where $I_0(x)$ is the zeroth order Bessel function.

The gate power is obtained by the summation of T independent realisations of the stochastic variable p. Under the assumption that T is large, the mean and variance of the gate power are readily calculated from Equations 1 and 2 by making use of the Central Limit Theorem. The variance of the MPT detector can then be studied as a function of the measurement power threshold D^2 by the numerical evaluation of Equations 1 and 2. From such a study, it was found that the optimal value of D for this detector is zero.

In Section 4, the Receiver Operating Characteristics (ROC) and minimum total error versus SNR curves are presented for the MPT detector. Section 4 also gives detection performance comparisons with the HMM detectors described next in Section 3.

3. DETECTION BASED ON HMMs

The HMM trackers presented by Streit and Barrett (1988) and Barrett, et.al., (1988) for tracking time-varying frequency lines are post-measurement device trackers. The FFTs of a sequence of blocked non-overlapped sampled time series data are passed through a similar gated threshold detector to that of the MPT detector. The detector output is the measured frequency of the signal, i.e., the midpoint of the FFT cell within the gate having the largest amplitude is the measurement, provided this amplitude exceeds the measurement threshold. If the threshold is not exceeded, the measurement is defined to be the "zero state". Thus, at every time step a measurement is made. Because of the zero state, the HMM trackers have the intrinsic capability of automatic track initiation and termination.

The HMM trackers presented by Streit and Barrett (1988) and Barrett, et.al, (1988) do not use phase or amplitude information, yet they are effective trackers. We have included amplitude information in the HMM tracker used in this paper. This extension is straightforward and does not materially alter the probabilistic methods used in the HMM tracker. However, the addition of amplitude information results in better detection performance at lower SNRs than the original HMM tracker described by Streit and Barrett (1988) and Barrett, et.al. (1988). The HMM tracker with amplitude information included is denoted by the acronym HMM/A, and its tracking performance is discussed by Steele, et.al., (1989).

The HMM trackers (either with or without amplitude information) can reconstruct the signal frequency track in several different ways by utilizing the different features of a general probabilistic structure that is defined from the mathematical structure of HMMs. The two features of interest in this paper are the posterior likelihood function that gives the likelihood of the entire measurement sequence, and the gate occupancy probability (GOP) function that gives the likelihood of the signal not occupying the zero state (i.e., signal presence at each time step). Complete details are given by Streit and Barrett (1988).

The first HMM/A detector we discuss is the log-likelihood ratio (LLR/A) detector. The likelihood functions used are the ones that the HMM/A tracker defines under the "signal present" and "signal absent" hypotheses. The "signal present" hypothesis is true if the measurement sequence is a realisation of the synthetic signal source characterised by the detector's HMM, while the "signal absent" hypothesis is true if the sequence is a realisation of a synthetic noise source characterised by the zero state of the detector's HMM. The parameters defining the detector's HMM are the assumed SNR, the process noise standard deviation d , the track initiation probability u and the track termination probability v. The LLR/A detector is optimal in the Neyman-Pearson sense; i.e., for a given probability of false alarm, the probability of signal detection is a maximum. We stress that the LLR/A detector is optimal if and only if the posterior likelihood function defined by the HMM/A tracker is exactly matched to the signal. Mismatch necessarily causes the LLR/A detector to be a sub-optimal detector.

The second detector is the GOP/A detector, so named because it integrates the GOP/A function that the HMM/A tracker uses to decide on track initiation and termination at each time step. When the HMM/A tracker initiates a track automatically, it is effectively registering a detection. The GOP/A detector is therefore employing the intrinsic signal detector of the HMM/A tracker. Because this detector is an integrator, it can be interpreted as an estimate of the fraction of the total measurement time interval for which a signal with certain specified values of the HMM parameters (SNR, d, u and v) is present. Analysis of the detection performance is made difficult by the fact that the GOP/A function values are highly correlated from time sample to time sample because the HMM/A tracker is a fixed interval tracker. The GOP/A detector is not equivalent to the LLR/A detector, and no optimality criterion is known for it.

Analytic expressions for the ROC curves of the LLR/A and GOP/A detectors are unknown. Preliminary theoretical results suggest, however, that the underlying conditional PDFs for the relevant

statistics, under the hypotheses of "signal present" and "signal absent", are asymptotically log-normal for both detectors. Further support is given by the observation that the ROC curves obtained below by simulation are straight lines on normal probability paper.

It will be seen in Section 4 that the GOP/A detector is superior to the LLR/A detector. Since the LLR/A uses an optimal detection statistic, the question naturally arises as to how the LLR/A detector can be inferior to the GOP/A detector. The answer is that the detector HMM, used in the LLR/A to characterise the "signal present" hypothesis, is not matched to the HMM used to simulate the input measurement sequence. The LLR/A detector is therefore sub-optimal for the simulated data.

4. RESULTS

Of the different detectors discussed above, only the performance of the MPT detector lends itself to theoretical analysis. Consequently, the detectors are compared in this paper by using simulation. The comparisons are made on the basis of ROC curves at different SNRs (Fig. 1) and plots of the minimum total error rate as a function of SNR (Fig. 2). For any SNR, as the signal detection threshold is raised or lowered, the relative frequency of occurrence of false alarms and false dismissals is varied. There exists an optimal detection threshold for which the combined sum of these two sources of error is minimised. Fig. 2 displays this minimum error rate as a function of SNR.

Simulated measurement sequences were generated by using the HMM/A tracker as a synthetic source. A simulated measurement sequence is a sequence of measurements at the output of the threshold measurement device when a signal of specified characteristics is input to it. The device measurement threshold is preselected and fixed. Thus, to simulate the "signal present" case, it is necessary to specify only the track SNR and the track process noise. The track is started in the middle of the gate and is prevented from terminating by setting the track termination probability, v, to zero. (Note that this model of "signal present" differs from that assumed by the HMM detector only in the underlying track initiation and termination probabilities.) The "signal absent" hypothesis is simulated by starting the track in the zero state, and setting SNR = 0, u = 0 and v = 1. Thus, the simulated "signal absent" measurement sequence consists of a sequence of independent measurements emitted from the zero state of the HMM.

Input data to the each of the detectors comprised a set of 10000 simulated measurement sequences, each of length 100 time steps, for both the "signal present" and "signal absent" cases. The input data to all detectors were identical. The HMM/A parameters u, v and d are unchanged by the addition of amplitude information to the original HMM tracker, and have been discussed in earlier publications by Streit and Barrett, (1988) and Barrett, et.al., (1988). The process noise standard deviation d for the simulated "signal present" measurement sequence was set to 0.333 (in FFT resolution cell widths). The parameters for the GOP detectors were: track initiation probability u = 0.00029, track termination probability v = 0.000035; process noise standard deviation d = 0.333; measurement threshold D = 0.

Fig. 1 shows the ROC curves for the MPT detector (solid lines) and the best of the HMM detectors, the GOP/A detector (dashed line). Fig.2 shows the minimum error as a function of SNR for the MPT detector and all of the HMM detectors investigated here (i.e., the GOP and LLR detectors, both with and without amplitude information).

The first conclusion to be drawn from Fig. 2 is that the MPT detector is a very good detector, and it is not until amplitude information is included that the two HMM detectors surpass the performance of the MPT detector. The MPT detector represents a substantial improvement over the conventional detector, and regularly produced correct detections with simulated data when no peaks were observable in the conventional power spectrum.

The optimal measurement threshold for the MPT detector is zero. For the HMM/A detectors, as noted in Section 2, the optimal measurement threshold (as determined from simulation studies) is sufficiently small that missed measurements occur only infrequently. However, for HMM detectors with no amplitude information, the optimal measurement threshold is significantly different from zero. The comparisons in Fig. 2, where all detectors have a zero measurement threshold, is therefore somewhat unfair towards the last two detectors. However, optimisation of the measurement threshold does not improve the performance of these two detectors to the point where they equalled the MPT detector in performance. This is remarkable considering the relative complexities of the three detectors, and indicates the importance of amplitude information to the detection process.

The inclusion of amplitude information in the HMM detectors results in approximately a 1.6 dB improvement in performance at an SNR of 0.002 (i.e.,-27 dB). In other words, the SNR would have to be increased by 1.6 dB before the performance of the HMM detectors without amplitude information would equal that of the HMM/A detectors. Here, we are defining SNR to be the ratio of signal power to broadband noise power. The best of the detectors was the GOP/A detector, which was marginally superior to the LLR/A detector, and 0.8 dB superior to the MPT detector. From the ROC curves in Fig. 1, it can be seen that for an SNR of 0.002 and a false alarm probability of 0.2%, the probability of detection for the MPT detector is 70% but is almost 90% for the GOP/A detector. This improved performance is achieved at the expense of a considerably increased numerical complexity.

5. CONCLUSIONS AND CONCLUDING REMARKS

The MPT detector is recommended in applications requiring simplicity, ease of implementation, and robust detection performance on unstable, or time-varying, frequency lines. The MPT detector is very simple to implement, yet it gives better detection performance than the conventional detector for unstable lines. Measurement outliers do not significantly affect the MPT detector because the track estimate implicit in the MPT detector is the detection sequence itself. Missed detections do not occur because numerical evaluation of Equations 1 and 2 shows that the optimal measurement threshold is zero.

To achieve better detection performance than that given by the MPT detector, it seems necessary to track the frequency line accurately and to include amplitude information in the tracker. The best HMM detectors studied in this paper utilize amplitude information and significantly out-perform the MPT detector. However, the HMM/A detectors require

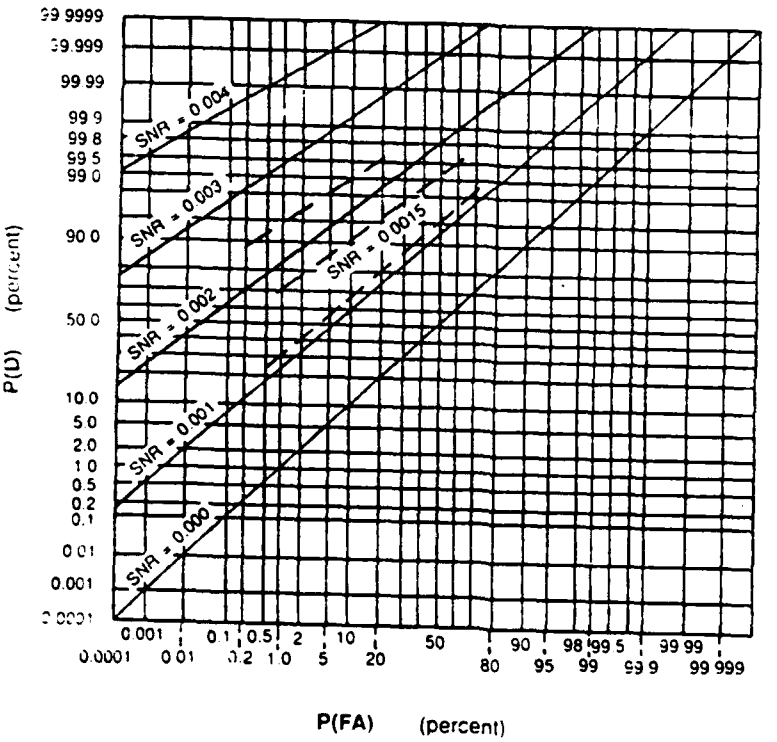


FIG. 1 Receiver Operating Characteristics (ROC) curves for the MPT (solid lines) and GOP/A (dashed lines) detectors.

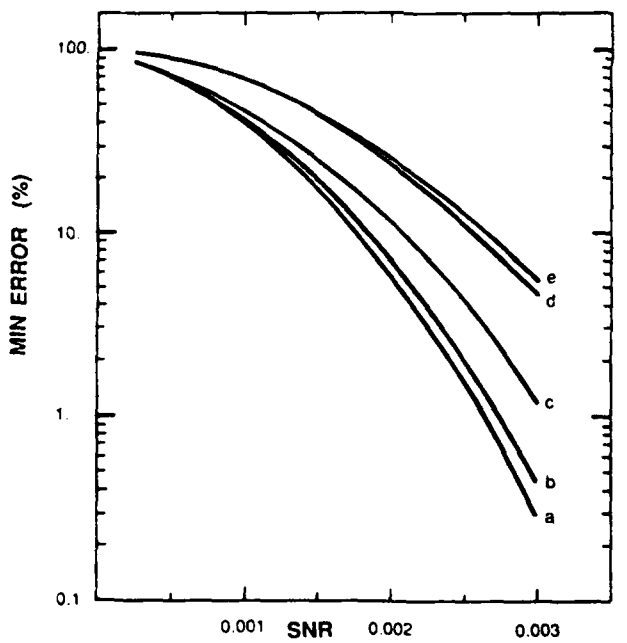


FIG. 2 The minimum error as a function of SNR for the detectors: a) GOP/A ; b) LLR/A ; c) MPT ; d) GOP ; e) LLR

significantly more computational effort and are not as easy to implement as the MPT detector. The HMM/A detectors are therefore recommended in applications requiring the best possible detection performance. Two track estimates are implicit in the HMM/A tracker, although neither track estimate is utilized explicitly by the detection algorithm. Measurement outliers do not seriously degrade the performance of HMM/A detectors because HMM/A trackers discriminate well against outliers. Missed measurements are not a problem for HMM/A detectors because simulation shows that the best measurement threshold to use is so small that missed measurements occur infrequently.

The computational burden required by HMM/A detectors adversely affects total system performance. It is possible, however, to greatly reduce these effects in some applications. For example, examination of the mathematical structure of HMM/A detection algorithms reveals that they can be formulated for several, say K, gates simultaneously, and this formulation is equivalent to a sequence of K-by-M matrix multiplications, if each gate has width M. This formulation can be exploited in either software or hardware, or both. In particular, if K = M, systolic arrays designed especially for K-by-K matrix multiplication can efficiently implement K HMM detectors in parallel. Inexpensive, commercially available, PC-level systolic arrays will probably become available in the not too distant future (see Marwood and Clarke, (1988)).

REFERENCES

Barrett, R.F., Steele, A.K., and Streit, R.L. (1988) Frequency line tracking algorithms. Proceedings of the NATO Advanced Study Institute on Underwater Acoustic Data Processing, Kingston, Ontario, Canada, 18-29 July 1988.

Marwood, W. and Clarke, A.P. (1988) A coprocessor with supercomputer capabilities for personal computers. Proceedings of the ICCD-IEEE International Conference on Computer Design: VLSI Computers and Processors, IEEE Computer Society Press, New York, pp. 468-471.

Steele, A.K., Streit, R.L., and Barrett, R.F. (1989) Nonlinear frequency line tracking algorithms, this conference.

Streit, R.L. and Barrett, R.F. (1988) Frequency line tracking using hidden Markov models. IEEE Trans. on Acoust., Sp., and Sig. Proc., submitted.

The Moments Of Matched And Mismatched Hidden Markov Models

R. L. Streit

The Moments of Matched and Mismatched Hidden Markov Models

ROY L. STREIT, SENIOR MEMBER, IEEE

Abstract—**An algorithm for computing the moments of matched and mismatched hidden Markov models from their defining parameters is presented. The algorithm is of general interest because it is an extension of the usual forward-backward linear recursion. The algorithm computes the joint moments of the posterior likelihood functions (i.e., the scores) by a multilinear recursion involving the joint moments of the random variables associated with the hidden states of the Markov chain. Examples comparing the first two theoretical moments to simulation results are presented. They are of independent interest because they indicate that the distribution of the posterior likelihood function scores for matched and mismatched models are asymptotically log-normal in important special cases and, therefore, are characterized asymptotically by the first two moments alone. One example discusses the effect of a noisy discrete communication channel on a suboptimal classification method based on the distributions of scores rather than on maximum likelihood classification.**

I. Introduction

HIDDEN Markov models (HMM's) are statistical models that are developed in diverse applications to characterize different classes of nonstationary time series or signals. Subsequently, HMM's are utilized for the automatic classification of an unknown signal into one of these signal classes. In speech applications, they are used to characterize the time variation of the short-term spectra of spoken words. An example is the speaker-independent isolated word recognition (SIIWR) problem where HMM's characterize the words (or parts of words) in a finite size vocabulary. Different words are characterized by different HMM's [1]. In target tracking applications, HMM's are used to characterize the time variation of a target track measurement sequence. A specific example is the narrow-band frequency line tracking problem where HMM's characterize possible target frequency shifts as well as noise in the measurement sequence for finite signal-to-noise ratio (SNR). Different HMM's characterize different target track dynamics and different SNR's [8]. A brief description of the mathematical structure of HMM's is given at the beginning of Section II.

An application-specific preprocessor is critical to the successful use of HMM's in the application. This preprocessor maps (or transforms) an arbitrary input signal $s(t)$, $t \geq 0$ into a discrete observation sequence $\{O(t), t = 1, 2, \cdots\}$. Reference [1, pp. 1077–1078] gives a description of one such preprocessor for the SIIWR problem, and [8] describes one suitable for frequency line tracking. Throughout this paper, it is assumed that a satisfactory preprocessor is available, but no assumptions are made about its specific nature. The output of the preprocessor constitutes the observation sequence. In practice, this sequence is truncated to have finite length T where T is selected according to the application needs. The truncated sequence is denoted by $O_T = \{O(t), t = 1, 2, \cdots, T\}$.

The act of computing specific numerical values for the various parameters of an HMM is called "training." Training takes place on the outputs of the preprocessor when it is given multiple realizations of a specific signal class. If the Baum–Welch reestimation algorithm is used for training, then training is equivalent to solving a mathematical optimization problem to determine maximum likelihood estimates of the HMM parameters [2]. In this paper, it is assumed that the training phase is completed and that the HMM's developed are adequate models for each of the signal classes of interest (e.g., the vocabulary words in the SIIWR problem or the target/SNR characteristics in the tracking application). We denote by HMM(i) the HMM parameter set defining the ith signal class. An important consequence of these training assumptions is that HMM(i) can be used as a synthetic signal source, that is, HMM(i) can be used to simulate the output of the preprocessor when the ith signal is input to it. We use the notation $O_T \in \text{HMM}(i)$ to mean that the observation sequence O_T is a realization of a random vector whose statistical distribution is defined implicitly by HMM(i).

HMM's are used for classification of an unknown observation sequence O_T by exploiting a probability measure or posterior likelihood function, as depicted in Fig. 1. The posterior likelihood function is defined on the set of all truncated sequences $\{O_T\}$ by utilizing the mathematical structure of HMM's. Thus, the likelihood of a given O_T depends critically on the numerical values of the parameters defining the underlying HMM. The ith HMM recognizer computes the posterior likelihood function $f_i(O_T)$. If HMM(i) is a finite symbol HMM (see Section II below), then $f_i(O_T)$ is equivalent to a probability, that is,

$$f_i(O_T) = \Pr\left[O_T \middle| \text{HMM}(i)\right], \quad i = 1, \cdots, p. \quad (1)$$

The maximum of the computed likelihoods identifies or classifies the original signal $s(t)$ that was input to the pre-

Manuscript received July 11, 1987; revised January 3, 1989.

The author is with the Naval Underwater Systems Center, New London, CT 06320.

IEEE Log Number 8934008.

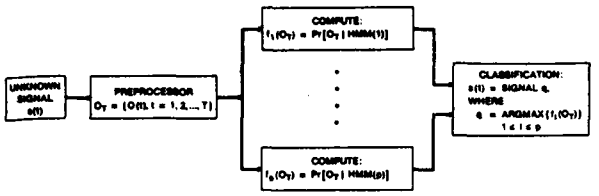


Fig. 1. Classification of unknown signal $s(t)$ as one of p signals for which trained HMM's are available.

processor. It is well known that this classifier is optimum in the Neyman-Pearson sense; that is, for a specified probability of incorrect classification, the probability of correct classification is a maximum [3].

The fundamental problem studied in this paper is the determination of the probability density function (pdf) of the test statistic $f_i(O_T)$ when $O_T \in \text{HMM}(j)$. In other words, if O_T is a random vector generated by HMM(j), what is the pdf of the numerical values of the ith posterior likelihood function $f_i(O_T)$? Note that the HMM's are matched if $i = j$ and mismatched if $i \neq j$. This paper presents an algorithm for computing explicitly the moments of the desired pdf up to any required order directly from the underlying parameters of the HMM's involved, and presents examples that compare the first two theoretical moments to simulation results. The algorithm is of general interest because it is an extension of the usual forward-backward linear recursion [2] for HMM's. It computes the joint moments of the likelihood functions $f_i(O_T)$ by a multilinear recursion involving the joint moments of the random (observation) variables uniquely associated with the hidden states of the HMM's. The examples are of independent interest as well. First, they indicate that the desired pdf is asymptotically log-normal in important special cases and, therefore, is completely characterized (asymptotically) by the first two moments alone. It is not obvious how the central limit theorem can be used to account for this result. Second, the examples show that a suboptimal classification method using preset detection thresholds for the likelihood functions $f_i(O_T)$ may be useful in certain instances. This point is discussed at the end of this section.

The distribution we seek is defined via its cumulative distribution function (cdf), denoted by $F_{ij}(x)$. It is intuitively appealing to attempt to define $F_{ij}(x)$ by setting

$$F_{ij}(x) = \Pr\left[f_i(O_T) < x \quad \text{and} \quad O_T \in \text{HMM}(j)\right]$$

where x is any real number; however, such a definition is ambiguous because the meaning of the probability measure $\Pr[\cdot]$ is unclear. Instead, for finite symbol HMM's, we define

$$F_{ij}(x) = \sum_{O_T} H(x - f_i(O_T)) f_j(O_T) \tag{2}$$

where the function $H(\cdot)$ is defined by

$$H(x) = 1 \quad \text{if } x \geq 0$$
$$H(x) = 0 \quad \text{if } x < 0.$$

From (2), it is clear that $F_{ij}(x)$ is a cdf because it is a nonnegative increasing right-continuous function, and the limit of $F_{ij}(x)$ is 0 as x goes to 0^- and 1 as x goes to $+\infty$. For continuous symbol HMM's, the summation over O_T in (2) must be replaced by integration over O_T. Algorithms that calculate F_{ij} directly from the HMM parameters are not known. For later reference, note that, in general, $F_{ij}(x) \neq F_{ji}(x)$.

The moments of $dF_{ij}(x)$ are defined by the Riemann-Stieltjes integral

$$M_{ij}(k, T) = \int_{-\infty}^{\infty} x^k dF_{ij}(x), \qquad k = 0, 1, 2, \cdots. \tag{3}$$

If $F_{ij}(x)$ is differentiable with derivative $F'_{ij}(x)$, then the moments can be written equivalently as the Riemann integral

$$M_{ij}(k, T) = \int_{-\infty}^{\infty} x^k F'_{ij}(x)\, dx.$$

The moments depend on the length T of the observation sequence because $F_{ij}(x)$ depends on T, as seen from (2). They uniquely determine $dF_{ij}(x)$ when they are all finite and the characteristic function of $dF_{ij}(x)$ has a finite radius of convergence [4]. For finite symbol HMM's, it is clear from (1) and (2) that $dF_{ij}(x) = 0$ for $x < 0$ and $x > 1$. Thus,

$$M_{ij}(k, T) = \int_0^1 x^k dF_{ij}(x) \leq 1$$

so that all the moments are finite. The series

$$\phi_{ij}(w) = \sum_{r=0}^{\infty} M_{ij}(r, T)(iw)^r / r!$$

for the characteristic function of $dF_{ij}(x)$ is absolutely convergent with an infinite radius of convergence because, for fixed $w_0 \neq 0$, each summand is bounded above in magnitude by $|w_0|^r / r!$, and thus the radius of convergence must be at least as large as $|w_0|$. Consequently, for finite symbol HMM's, the moments of $dF_{ij}(x)$ uniquely determine $dF_{ij}(x)$. A similar argument holds for continuous symbol HMM's, provided the likelihood functions $f_i(O_T)$ are bounded on the set of all sequences $\{O_T\}$. In this paper, we assume that the likelihood functions are bounded because such an assumption is not particularly restrictive for applications.

Receiver-operator characteristic (ROC) curves [3] are commonly used in the radar and sonar communities to provide quantitative assessments of the correct and incorrect classification rates for classification schemes based on likelihood functions. ROC curves can be used for the same purpose here. To develop a ROC curve for a given classification-related test statistic, say q, under two hypotheses H_i and H_j, the conditional pdf's (using the notation in [3])

$$p_{q|H_i}(Q|H_i) \quad \text{and} \quad p_{q|H_j}(Q|H_j)$$

that define the test statistic q under the different hypotheses H_i and H_j, respectively, must be known. For each real number u, $-\infty < u < +\infty$, we define the probability

$$P_F(u) = \int_u^\infty p_{q|H_i}(Q|H_i)\,dQ$$

and the probability

$$P_D(u) = \int_u^\infty p_{q|H_j}(Q|H_j)\,dQ.$$

The ROC curve for q is simply the locus of points $(P_F(u), P_D(u))$ parameterized by u. The parameter u is usually treated as a decision threshold in applications. Suppose the decision threshold u_{thresh} is selected. Then if $q \geq u_{\text{thresh}}$, the classifier decides H_j. The probability of this decision being correct is P_D, and the probability that it is incorrect is P_F. P_F and $1 - P_D$ are usually referred to as the false alarm and false dismissal probabilities, respectively. Analogous remarks pertain if $q < u_{\text{thresh}}$. Note that the ROC curve for $u = -\infty$ goes through the point $(1, 1)$ and for $u = +\infty$ it goes through the point $(0, 0)$.

The ROC curve of the optimum classifier depicted in Fig. 1, under the hypotheses $O_T \epsilon$ HMM(i) and $O_T \epsilon$ HMM(j), is determined for the likelihood ratio test statistic

$$q_{\text{opt}} = f_j(O_T)/f_i(O_T).$$

The required conditional pdf for q_{opt} is defined by the cdf

$$L_{ij}(x) = \sum_{O_T} H\big(x - \{f_j(O_T)/f_i(O_T)\}\big) f_j(O_T).$$

No recursion for $L_{ij}(x)$ is known, so the only way to evaluate it is by doing the summation; however, this is impractical because the number of terms in the summation grows exponentially in T. Simulation is probably the best method for estimating the ROC curves for the optimal test statistic q_{opt}. In any event, a decision threshold u_{ij} must be set to enable classification to proceed. The "natural" threshold to set is $u_{ij} = 1$ for all i and j, for then the maximum likelihood determines the classification, the classification decision is unique (except for ties) and the classifier depicted in Fig. 1 is obtained. However, in general, it is not necessary to make the natural choice. The best choice depends on the false alarm and false dismissal requirements for each pair of hypotheses $O_T \epsilon$ HMM(i) and $O_T \epsilon$ HMM(j) in the application.

The ROC curve of the suboptimal classifier, under the hypotheses HMM(i) and HMM(j), is determined for the test statistic

$$q_{\text{subopt}} = f_j(O_T).$$

The required conditional pdf's for q_{subopt} are given by $dF_{ji}(x)$ and $dF_{jj}(x)$, respectively. As shown in Section II, the moments of $dF_{ji}(x)$ and $dF_{jj}(x)$ can be computed to any desired order; hence, the ROC curve for q_{subopt} can, in principle, be approximated to any required accuracy without resorting to simulation. A natural choice of decision threshold u_{ij} for each pair of hypotheses $O_T \epsilon$ HMM(i) and $O_T \epsilon$ HMM(j) is not available. Instead, the thresholds must be set by direct examination of the ROC curves.

The test statistic q_{subopt} is identical to q_{opt} in one important special case. If HMM(i) is such that $f_i(O_T)$ in the denominator of q_{opt} is a constant function of O_T, then q_{subopt} can be scaled so that $q_{\text{subopt}} = q_{\text{opt}}$. A situation that might require such an HMM(i) is one in which white noise is being modeled, for then one might anticipate that all observation sequences at the output of the preprocessor are equally likely. The classification statistic is more appropriately referred to as a "detection" statistic in this instance. Thus, a ROC curve for the optimum detection statistic can be developed from the moments computed by the algorithm given in Section II.

The use of q_{subopt} in preference to q_{opt} is appropriate only if the associated conditional pdf's for the ROC curves are "well separated" from each other, and if the application places great emphasis on control of the false alarm or false dismissal probabilities. In this situation, both q_{opt} and q_{subopt} are very likely to perform well; however, estimated ROC curves for q_{opt} would have to be obtained from very large simulations, especially if very small false alarm probabilities or false dismissal probabilities are required in the application. On the other hand, ROC curves for q_{subopt} can be obtained reliably without simulation. In any event, classification performance using q_{subopt} should bound the classification performance using q_{opt}.

II. The Moment Algorithm

Every HMM is comprised of two basic parts: a Markov chain and a set of random variables. The Markov chain has a finite number of states, and each state is uniquely associated with one of the random variables. The state sequence generated by the chain is not observable, i.e., the Markov chain is "hidden." At each time $t = 0, 1, 2, \cdots$, the Markov chain is assumed to be in some state; it transitions to another state at time $t + 1$ according to its transition probability matrix. At each time t, one observation is generated by the random variable associated with the state of the Markov chain at time t. The observations are referred to as symbols. If the random variables assume only a finite set values, the HMM is referred to as a finite symbol HMM. If the random variables assume a continuum of values, the HMM is called a continuous symbol HMM. The full parameter set defining an HMM is comprised of the initial state probability density function of the Markov chain at time $t = 0$, the Markov chain state transition probability matrix, and the pdf's of each of the random observation variables.

The reader is referred to [2] for further discussion of HMM's and the basic algorithms related to them. Of particular importance is the forward-backward algorithm that is used extensively in this section. It is not necessary to read the remainder of this section to understand the examples presented in Section III.

The algorithm is presented separately for finite and continuous symbol HMM's in Section II-A and II-B, respectively. Since the presentation uses only the forward part of the forward–backward algorithm, the algorithm may be named the forward moment algorithm. Section II-C contains a statement of the backward moment algorithm and an identity that is analogous to the Baum identity of the usual forward–backward algorithm.

A. Finite Symbol HMM's

Let HMM(ν) be a hidden Markov chain with $n(\nu)$ states, $\nu = 1, \cdots$. Subscripted indexes will always be written as functions of their subscripts (for instance, $n(\nu)$ is used instead of n_ν) to avoid the later use of subscripted subscripts. Let the state transition probability matrix of HMM(ν) be denoted as $A^\nu = [a^\nu_{i(\nu),j(\nu)}]$ for $i(\nu), j(\nu) = 1, \cdots, n(\nu)$. Let the initial state probability vector of HMM(ν) be denoted as $\pi^\nu = [\pi^\nu_{i(\nu)}]$ for $i(\nu) = 1, \cdots, n(\nu)$.

We first restrict attention to finite symbol HMM's, that is, we suppose that every observation sequence $O_T = \{O(t), t = 1, \cdots, T\}$ is such that

$$O(t) \in V = \{V_1, \cdots, V_m\}$$

where V is the set of all possible output symbols of the preprocessor. The true nature of the symbols in V is of no importance here. HMM's assume that $O(t)$ is a random variable whose probability density function depends on the current state of the Markov chain. Let the discrete probability density function for HMM(ν) when it is in state $i(\nu)$ be denoted as $B^\nu_{i(\nu)}$ for $i(\nu) = 1, \cdots, n(\nu)$. Thus, each $B^\nu_{i(\nu)}$ is a row vector of length m. Stacking these row vectors gives the $n(\nu) \times m$ symbol probability matrix

$$B^\nu = [b^\nu_{i(\nu),j(\nu)}] = \begin{bmatrix} B^\nu_1 \\ B^\nu_2 \\ \vdots \\ B^\nu_{n(\nu)} \end{bmatrix}.$$

Note that

$$b^\nu_{i(\nu)}(V_{j(\nu)}) = b^\nu_{i(\nu),j(\nu)}$$

where we define

$$b^\nu_{i(\nu)}(O(t)) = \Pr\left[O(t) \middle| \text{HMM}(\nu) \text{ and state} = i(\nu)\right].$$

The assumption that the training phase is completed means that the parameters HMM(ν) = (π^ν, A^ν, B^ν) are known.

For finite symbol HMM's, $F_{ij}(x)$ has a finite number of jump discontinuities. Let X_{ij} denote the set of all values of x for which $F_{ij}(x)$ is discontinuous. Definition (2) implies that the discontinuities of $F_{ij}(x)$ occur precisely at the different possible values of $f_i(O_T)$. Define the subset $S_i(x)$ of the set of all observation sequences $\{O_T\}$ by

$$S_i(x) = \{O_T : f_i(O_T) = x\}.$$

The sets $S_i(x)$ and $S_i(y)$ are disjoint if $x \neq y$. Also, the union of $S_i(x)$ over all x in X_{ij} is the set $\{O_T\}$ of all observation sequences. Now, from definition (2), it follows that

$$dF_{ij}(x) = F_{ij}(x+) - F_{ij}(x-) = \sum_{O_T \in S_i(x)} f_j(O_T). \tag{4}$$

Substituting (4) into (3) gives

$$\begin{aligned} M_{ij}(k, T) &= \sum_{x \in X_{ij}} x^k \sum_{O_T \in S_i(x)} f_j(O_T) \\ &= \sum_{x \in X_{ij}} \sum_{O_T \in S_i(x)} \{f_i(O_T)\}^k f_j(O_T) \\ &= \sum_{O_T} \Pr\left[O_T \middle| \text{HMM}(i)\right]^k \Pr\left[O_T \middle| \text{HMM}(j)\right] \end{aligned} \tag{5}$$

where, in the last equation, we have used (1). It is clear from (5) that $M_{ij}(k, T) \neq M_{ji}(k, T)$, in general, for $k > 1$. For $k = 1$, however, we have $M_{ij}(1, T) = M_{ji}(1, T)$ for all i, j, and T.

The expression in (5) is computable directly from the parameters of HMM(i) and HMM(j); however, such a calculation is not practical except for small T because the computational effort increases exponentially in T. To see this, note that the forward–backward algorithm [2] calculates $\Pr[O_T|\text{HMM}(\nu)]$ using $n^2(\nu)T$ multiplications. Thus, each summand in (5) requires $[n(i)n(j)]^2T^2$ multiplications. There are m^T different possible observation sequences $O_T = \{O(t), t = 1, \cdots, T\}$ because each $O(t)$ can be any one of the m output symbols in V. Thus, direct calculation of (5) requires a total of $[n(i)n(j)]^2T^2m^T$ multiplications.

We now derive a recursion for (5) that requires computational effort that grows only linearly with T. The recursion is derived for a more general expression that contains (5) as a special case. For $k = 1, 2, \cdots$, define

$$R(k, T) = \sum_{O_T} \prod_{\nu=1}^{k} \Pr\left[O_T \middle| \text{HMM}(\nu)\right]. \tag{6}$$

The application of (6) to compute moments is straightforward; for example, $R(k+1, T)$ equals $M_{21}(k, T)$ when HMM(2) = $\cdots$ = HMM($k+1$). Note that $R(k, T)$ can be interpreted as a joint moment of HMM's, that is, as a joint moment of the likelihood functions $f_i(O_T)$ of the HMM's.

The derivation of the recursion for $R(k, T)$ proceeds as follows. The forward recursion portion of the forward–backward algorithm gives the expression

$$\Pr\left[O_T \middle| \text{HMM}(\nu)\right] = \sum_{j(\nu)=1}^{n(\nu)} \alpha^\nu_T(j(\nu)) \tag{7}$$

where, for $2 \leq t \leq T$,

$$\alpha^\nu_t(j(\nu)) = \left[\sum_{i(\nu)=1}^{n(\nu)} \alpha^\nu_{t-1}(i(\nu))\, a^\nu_{i(\nu),j(\nu)}\right] b^\nu_{j(\nu)}(O(t)) \tag{8}$$

and

$$\alpha_1^\nu(j(\nu)) = \pi^\nu_{j(\nu)}\, b^\nu_{j(\nu)}(O(1)). \tag{9}$$

Substitute (7) into (6) to obtain

$$R(k, T) = \sum_{\substack{j(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \sum_{O_T} \prod_{\nu=1}^{k} \alpha_T^\nu(j(\nu))$$

$$= \sum_{\substack{j(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \mu_T(j(1), \cdots, j(k)) \tag{10}$$

where we define, for $t = 1, \cdots, T$,

$$\mu_t(j(1), \cdots, j(k)) = \sum_{O_t} \prod_{\nu=1}^{k} \alpha_t^\nu(j(\nu)). \tag{11}$$

One interpretation of μ_T is that it equals $R(k, T)$ given that HMM(ν) must end in state $j(\nu)$, $\nu = 1, \cdots, k$. We seek a recursion for μ_T. First suppose that $2 \le t \le T$. Then, substituting (8) into (11) gives

$$\mu_t(j(1), \cdots, j(k))$$

$$= \sum_{O_t} \prod_{\nu=1}^{k} \left\{ \sum_{i(\nu)=1}^{n(\nu)} \alpha_{t-1}^\nu(i(\nu))\, a^\nu_{i(\nu),j(\nu)} b^\nu_{j(\nu)}(O(t)) \right\}$$

$$= \sum_{\substack{i(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \sum_{O_t} \left\{ \left[\prod_{\nu=1}^{k} \alpha_{t-1}^\nu(i(\nu)) \right] \left[\prod_{\nu=1}^{k} a^\nu_{i(\nu),j(\nu)} \right] \cdot \left[\prod_{\nu=1}^{k} b^\nu_{j(\nu)}(O(t)) \right] \right\}$$

$$= \sum_{\substack{i(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \left[\prod_{\nu=1}^{k} a^\nu_{i(\nu),j(\nu)} \right] \left\{ \sum_{O_t} \left[\prod_{\nu=1}^{k} \alpha_{t-1}^\nu(i(\nu)) \right] \cdot \left[\prod_{\nu=1}^{k} b^\nu_{j(\nu)}(O(t)) \right] \right\}.$$

Because $\alpha^\nu_{t-1}(i(\nu))$ does not depend on the last symbol $O(t)$ in the observation sequence $O_t = \{O(1), \cdots, O(t)\}$, we have

$$\mu_t(j(1), \cdots, j(k))$$

$$= \sum_{\substack{i(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \left[\prod_{\nu=1}^{k} a^\nu_{i(\nu),j(\nu)} \right] \left\{ \sum_{O_{t-1}} \left[\prod_{\nu=1}^{k} \alpha_{t-1}^\nu(i(\nu)) \right] \cdot \sum_{O(t)} \left[\prod_{\nu=1}^{k} b^\nu_{j(\nu)}(O(t)) \right] \right\}.$$

Because the sum over $O(t)$ is independent of the observation sequence $O_{t-1} = \{O(1), \cdots, O(t-1)\}$, as well as the indexes $i(\nu)$, and because of (11), we have

$$\mu_t(j(1), \cdots, j(k)) = \Gamma(j(1), \cdots, j(k)) \sum_{\substack{i(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \left[\prod_{\nu=1}^{k} a^\nu_{i(\nu),j(\nu)} \right] \mu_{t-1}(i(1), \cdots, i(k)) \tag{12}$$

where

$$\Gamma(j(1), \cdots, j(k)) = \sum_{O(t)} \prod_{\nu=1}^{k} b^\nu_{j(\nu)}(O(t)) = \sum_{s=1}^{m} \prod_{\nu=1}^{k} b^\nu_{j(\nu)}(V_s). \tag{13}$$

Note that Γ is the joint moment of the random observation variables uniquely associated with the state $j(\nu)$ of HMM(ν).

Equation (12) is the desired recursion for $2 \le t \le T$. For $t = 1$, substituting (9) into (11) gives

$$\mu_1(j(1), \cdots, j(k)) = \sum_{O(1)} \prod_{\nu=1}^{k} \alpha_1^\nu(j(\nu)) = \left(\prod_{\nu=1}^{k} \pi^\nu_{j(\nu)} \right) \sum_{O(1)} \prod_{\nu=1}^{k} b^\nu_{j(\nu)}(O(1)) = \Gamma(j(1), \cdots, j(k)) \prod_{\nu=1}^{k} \pi^\nu_{j(\nu)}. \tag{14}$$

Let $k = 1$. From the definition, it is clear that $R(1, T) = 1$ for all T, regardless of HMM(1) because the sum in (6) is over all O_T. To check independently the recursion (12)–(13), note that, from (13),

$$\Gamma(j(1)) = \sum_{s=1}^{m} b^1_{j(1)}(V_s) = 1, \qquad 1 \le t \le T.$$

From (14), we have

$$\mu_1(j(1)) = \pi^1_{j(1)}.$$

Hence, from (10), we obtain

$$R(1, 1) = \sum_{\substack{j(1) \\ j(1)=1}}^{n(1)} \pi^1_{j(1)} = 1.$$

The recursion is verified for $T = 1$. For $T = 2$, from (12), we have

$$\mu_2(j(1)) = \sum_{i(1)=1}^{n(1)} \mu_1(i(1))\, a^1_{i(1),j(1)} = \sum_{i(1)=1}^{n(1)} \pi^1_{i(1)} a^1_{i(1),j(1)}$$

so that, from (10),

$$R(1,2) = \sum_{j(1)=1}^{n(1)} \left\{ \sum_{i(1)=1}^{n(1)} \pi^1_{i(1)} a^1_{i(1),j(1)} \right\}$$
$$= \sum_{i(1)=1}^{n(1)} \left\{ \pi^1_{i(1)} \sum_{j(1)=1}^{n(1)} a^1_{i(1),j(1)} \right\}$$
$$= 1$$

and the recursion is verified for $T = 2$.

The first nontrivial special case is $k = 2$. In this case, $R(2, T)$ is identically the first moment $M_{12}(1, T)$. From (12), we have for $2 \le t \le T$

$$\mu_t(j(1), j(2)) = \Gamma(j(1), j(2)) \cdot \sum_{i(1)=1}^{n(1)} \sum_{i(2)=1}^{n(2)} \mu_{t-1}(i(1), i(2)) a^1_{i(1),j(1)} a^2_{i(2),j(2)}$$

and, from (14),

$$\mu_1(j(1), j(2)) = \Gamma(j(1), j(2)) \pi^1_{j(1)} \pi^2_{j(2)}$$

where, from (13),

$$\Gamma(j(1), j(2)) = \sum_{s=1}^{m} b^1_{j(1)}(V_s)\, b^2_{j(2)}(V_s).$$

From (10), then, we have

$$R(2, T) = \sum_{j(1)=1}^{n(1)} \sum_{j(2)=1}^{n(2)} \mu_T(j(1), j(2)).$$

Computation of $R(2, T) = M_{12}(1, T)$ is therefore not excessively laborious.

The evaluation of $R(k, T)$ using the recursion (12) is properly broken into two parts. The first is the precalculation of $\Gamma(j(1), \cdots, j(k))$ for every possible value of the indexes $j(\nu)$. This requires $(k-1)\, m\, N^k$ multiplications and N^k storage locations, where

$$N = \left[\prod_{\nu=1}^{k} n(\nu) \right]^{1/k} \tag{15}$$

is the geometric mean of the number of different states in the various HMM's and is not necesarily an integer. If $N = 8$ and if there are $m = 16$ different observation symbols, then computing and storing Γ for $k = 16$ requires 262 144 storage locations and 2.1×10^7 multiplications. Storage is clearly more crucial an issue than multiplications.

It is possible in some cases to utilize the underlying symmetries of Γ to reduce both storage and computational effort. For example, if $\text{HMM}(2) = \cdots = \text{HMM}(k+1)$, then

$$\Gamma(j(1), j(2), \cdots, j(k+1)) = \Gamma(j(1), \sigma(j(2)), \cdots, \sigma(j(k+1))) \tag{16}$$

for every permutation σ of the k integers $j(2), \cdots, j(k+1)$. The proof of (16) follows easily from (13) because multiplication is commutative. Thus, one only need consider indexes that satisfy

$$1 \le j(1) \le n(1) \quad \text{and}$$
$$1 \le j(2) \le j(3) \le \cdots \le j(k+1) \le n(2).$$

The number of ordered index sets $(j(2), \cdots, j(k+1))$ is equal to the number of combinations of $n(2)$ letters taken k at a time, when each letter may be repeated any number of times up to k. Storage is therefore proportional to

$$N_{k+1} = \left(\frac{n(2)(n(2)+1)\cdots(n(2)+k-1)}{k!} \right) n(1)$$

which is significantly smaller than the $[n(2)]^k n(1)$ storage that would otherwise be necessary. The total multiplication count is also reduced proportionately.

Once Γ has been computed and stored for a given value of k, the recursion (12) can be computed for any length T of the observation sequence. For each of the N^k sets of indexes $\{j(\nu)\}$ in (12), the sum over all N^k indexes $\{i(\nu)\}$ must be undertaken. This sum appears to require kN^k multiplications; however, by using the nested form,

$$\sum_{i(1)=1}^{n(1)} a^1_{i(1),j(1)} \left[\sum_{i(2)=1}^{n(2)} \cdots \left[\sum_{i(k)=1}^{n(k)} a^k_{i(k),j(k)} \mu_{t-1}(i(1), \cdots, i(k)) \right] \cdots \right],$$

it is possible to use approximately

$$N^k + N^{k-1} + \cdots N^2 + N = \left(\frac{N}{N-1} \right)(N^k - 1)$$

instead. If lower order terms are neglected, computing one iteration of (12) requires about N^{2k} multiplications. For an observation sequence of length T, computing μ_T requires on the order of $N^{2k}T$ multiplications. If $N = 8$ and $T = 32$, then 2.2×10^{12} multiplications are required for $k = 6$. Assuming a multiplication takes 1 μs, the calculation requires 611 h and is clearly impractical.

Significant reduction in computational effort is possible in some cases by utilizing the underlying symmetries in μ_t. For example, if $\text{HMM}(2) = \cdots = \text{HMM}(k+1)$, then

$$\mu_t(j(1), j(2), \cdots, j(k+1)) = \mu_t(j(1), \sigma(j(2)), \cdots, \sigma(j(k+1))) \tag{17}$$

for every permutation σ of the k integers $j(2), \cdots, j(k+1)$. The proof of (17) follows easily by induction from (12) because multiplication is commutative and because Γ satisfies the same symmetry property (16) in this case.

Thus, the recursion (12) need be computed for only N_{k+1} sets of indexes. The total multiplication count is reduced to $4N_{k+1}^2 T$, which is significantly smaller than the $N^{2k}T$ multiplications that would otherwise be needed. For the above example requiring 611 h, if $N = n(1) = n(2) = 8$ and if the symmetry (17) is utilized, the calculation would be reduced to roughly a 96 min calculation. Utilizing symmetry is clearly significant in that it can turn an impractical long calculation into a feasible shorter one.

Underflow is potentially a problem when the recursion (12) is computed. It can be easily overcome in exactly the same manner as pointed out in [2] for preventing numerical underflow during the calculation of the forward-backward algorithm. Specifically, let μ_t be computed according to (12) and then multiplied by a scale factor c_t defined by

$$c_t = \left[\sum_{\substack{j(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \mu_t(j(1), \cdots, j(k)) \right]^{-1}. \quad (18)$$

Then use the scaled values of μ_t in the recursion (12) to compute μ_{t+1}, which is in turn scaled as shown in (18). If we continue in this fashion and recall the expression in (10), it follows that

$$R(k, T) = \left(\prod_{t=1}^{T} c_t \right)^{-1}. \quad (19)$$

Because the product cannot be evaluated without underflow, we compute instead

$$\log R(k, T) = -\sum_{t=1}^{T} \log c_t. \quad (20)$$

Any convenient scale factor can be used instead of (18). A potentially useful one might be to take $\bar{c}_t = N^k$. Using $\bar{c}_t$ would eliminate the effort of computing the sum in (18) before scaling.

B. Continuous Symbol HMM's

The objective of this section is to show that the moment algorithm for discrete symbol HMM's can be carried over essentially unchanged to continuous symbol HMM's. In fact, it holds also for continuous vector symbol HMM's; however, only the continuous symbol HMM's are treated here for simplicity.

Throughout this section, it is assumed that each output symbol $O(t)$ is a real random variable defined on some underlying event space V. The probability density function of $O(t)$ is uniquely defined for each state $i(\nu) = 1, \cdots, n(\nu)$ of each HMM(ν), $\nu = 1, 2, \cdots$, and is denoted as $b^{\nu}_{i(\nu)}(x)$. Thus, for real numbers α and β with $\alpha < \beta$, we have

$$\int_{\alpha}^{\beta} b^{\nu}_{i(\nu)}(x)\, dx = \Pr\left[\alpha \le O(t) \le \beta \,|\, \text{HMM}(\nu) \text{ and state } = i(\nu)\right]. \quad (21)$$

An observation sequence $O_T = \{x_t, t = 1, 2, \cdots, T\}$ is a sequence of real numbers x_t, with x_t being a realization of the random variable $O(t)$. The posterior likelihood function $f_{\nu}(O_T)$ is a probability density function for continuous symbol HMM's, as opposed to a simple probability [see (1)] for discrete symbol HMM's. Thus, for real vectors $\vec{\alpha}$ and $\vec{\beta}$ with $\vec{\alpha} < \vec{\beta}$, we have

$$\int_{\vec{\alpha}}^{\vec{\beta}} f_{\nu}(O_T)\, dO_T = \Pr\left[\vec{\alpha} \le O_T \le \vec{\beta} \,|\, \text{HMM}(\nu)\right] \quad (22)$$

where $dO_T = dx_1 \cdots dx_T$.

For continuous symbol HMM's, the functions $F_{ij}(x)$ are defined just as in (2), but with a T-fold integral over O_T replacing the T-fold sum over O_T. Thus, we have the differential

$$dF_{ij}(x) = \int_{O_T} \delta(x - f_i(O_T)) f_j(O_T)\, dO_T\, dx$$

where $\delta(\cdot)$ denotes the Dirac delta function. From (3), the moments are given by

$$\begin{aligned} M_{ij}(k, T) &= \int_{-\infty}^{\infty} x^k dF_{ij}(x) \\ &= \int_{-\infty}^{\infty} x^k \int_{O_T} \delta(x - f_i(O_T)) f_j(O_T)\, dO_T\, dx \\ &= \int_{O_T} f_j(O_T) \int_{-\infty}^{\infty} x^k \delta(x - f_i(O_T))\, dx\, dO_T \\ &= \int_{O_T} \{f_i(O_T)\}^k f_j(O_T)\, dO_T \end{aligned} \quad (23)$$

which is the continuous analog of (5). It is clear from (23) that $M_{ij}(k, T) = M_{ji}(k, T)$ in general only for the special case $k = 1$. The analog of (6) for continuous symbol HMM's is

$$R(k, T) = \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{T\text{-fold}} \prod_{\nu=1}^{k} f_{\nu}(O_T)\, dO_T. \quad (24)$$

The forward-backward algorithm for computing the posterior likelihood function for continuous symbol HMM's is modified [5] as follows:

$$f_{\nu}(O_T) = \sum_{j(\nu)=1}^{n(\nu)} \alpha^{\nu}_T(j(\nu)) \quad (25)$$

where $\alpha^{\nu}_T(j(\nu))$ is computed exactly as given by the recursion (8) and (9), with the only difference being that $b^{\nu}_{j(\nu)}(O(t))$ in (8) is now interpreted as the probability density function implicit in (21). Consequently, (10) still holds exactly if we define

$$\mu_t(j(1), \cdots, j(k)) = \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{t\text{-fold}} \prod_{\nu=1}^{k} \alpha^{\nu}_t(j(\nu))\, dO_t \quad (26)$$

as the analog of (11). Proceeding as before with t-fold integrals replacing t-fold summations gives exactly the recursion (12), but with the one-dimensional integral

$$\Gamma(j(1), \cdots, j(k)) = \int_{-\infty}^{\infty} \prod_{\nu=1}^{k} b_{j(\nu)}^{\nu}(x)\, dx \quad (27)$$

in place of (13).

The remarks in the preceding section concerning storage, multiplication counts, and symmetry properties all apply for continuous symbol HMM's. The primary difference is that (27) requires an integral evaluation instead of a finite sum as in (13). This evaluation increases the initial computational overhead, but once (27) is computed, the algorithm (12) proceeds exactly as before.

C. The Forward-Backward Moment Algorithm

The moment algorithm presented above in this section used the forward probabilities defined by (8)–(9). It is equally feasible to use the backward probabilities for the same purpose. They are defined by

$$\beta_T(j(\nu)) = 1$$

and, for $T - 1 \geq t \geq 1$, by

$$\beta_t(j(\nu)) = \sum_{i(\nu)=1}^{n(\nu)} a_{j(\nu),i(\nu)} b_{i(\nu)}(O(t+1))\beta_{t+1}(i(\nu)).$$

The backward moment algorithm computes, for $1 \leq t \leq T - 1$, the function

$$\tau_t(j(1), \cdots, j(k)) = \sum_{\tilde{O}_t} \prod_{\nu=1}^{k} \beta_t(j(\nu))$$

where $\tilde{O}_t = \{O(t+1), \cdots, O(T)\}$. The backward recursion is given by

$$\tau_T(j(1), \cdots, j(k)) = 1$$

and, for $T - 1 \geq t \geq 1$, by

$$\tau_t(j(1), \cdots, j(k)) = \sum_{\substack{i(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \left[\prod_{\nu=1}^{k} a_{j(\nu),i(\nu)}\right] \cdot \tau_{t+1}(i(1), \cdots, i(k))\Gamma(i(1), \cdots, i(k)).$$

The derivation of this recursion is similar to that of (12)–(13).

It is straightforward to show that for any t, $1 \leq t \leq T$,

$$R(k, T) = \sum_{\substack{j(\nu)=1 \\ \nu=1,\cdots,k}}^{n(\nu)} \mu_t(j(1), \cdots, j(k)) \cdot \tau_t(j(1), \cdots, j(k)).$$

Note that the case $t = T$ is (10). This identity is the analog for $R(k, T)$ of the well-known Baum identity [2] for likelihood functions, i.e.,

$$f_\nu(O_T) = \sum_{i(\nu)=1}^{n(\nu)} \alpha_t(i(\nu))\beta_t(i(\nu)).$$

III. Comparison of Theoretical Moments to Simulation

Ergodic Markov chains are those for which it is possible to transition from every state to every other state, although not necessarily in one step. Left-to-right Markov chains are those for which transitions to lower numbered states are not allowed, that is, have probability zero. These two types of chains are sufficiently different that they are considered separately in the examples.

One interpretation is that ergodic HMM's are models of quasi-stationary signals, while left-to-right HMM's are models of transient signals that ultimately become stationary (because the highest numbered state is not exited once it is entered). One might therefore expect these two types of HMM's to affect classification performance in different ways. The three examples given in this section support this expectation.

Using the above interpretation, the examples may be described as follows. The first example shows that classification using the suboptimal statistic q_{subopt} reliably distinguishes between sufficiently long quasi-stationary signals with a reasonable amount of computational effort. The second example shows that short quasi-stationary and transient signals look significantly different to the HMM transient recognizer, but *not* to the HMM recognizer based on the quasi-stationary signal. The third example shows that noisy observations of transient signals adversely affect classification performance by making the transient signal appear to have a stationary component, which is then misclassified by the HMM transient recognizer.

A. Two Ergodic HMM's

HMM(1) and HMM(2) are five-state, eight-symbol ergodic models whose parameters are given (rounded to three significant decimals) in Tables I and II, respectively. HMM(1) clearly generates observation sequences of uniformly distributed symbols. HMM(2) is more complex in structure, but every symbol can be generated in every state. The fundamental question of interest here is the following. How long must an observation sequence be to guarantee that the suboptimal classification statistic q_{subopt} is highly reliable (say, 99% correct) and has a low false dismissal rate (say, of 0.5%)? We will give what may best be described as a semiempirical answer to this question.

Because of the nature of HMM(1), it is easy to see that

$$f_1(O_T) = \Pr\{O_T \mid \text{HMM}(1)\} = 8^{-T}.$$

In other words, the posterior likelihood function based on HMM(1) is constant because all observation sequences are equally likely if $O_T \in$ HMM(1). In particular, $f_1(O_T)$ cannot distinguish $O_T \in$ HMM(1) from $O_T \in$ HMM(2) and thus is useless for classification.

The posterior likelihood function based on HMM(2), instead of HMM(1), is useful for classification. Ten-thousand observation sequences O_T of each HMM were generated, and the posterior likelihood $f_2(O_T)$ was com-

TABLE I
PARAMETERS OF HMM(1)

NUMBER OF MARKOV STATES = 5

NUMBER OF SYMBOLS PER STATE = 8

INITIAL STATE PROBABILITY VECTOR:

2.00E-01	2.00E-01	2.00E-01	2.00E-01	2.00E-01

TRANSITION PROBABILITY MATRIX:

2.00E-01	2.00E-01	2.00E-01	2.00E-01	2.00E-01
2.00E-01	2.00E-01	2.00E-01	2.00E-01	2.00E-01
2.00E-01	2.00E-01	2.00E-01	2.00E-01	2.00E-01
2.00E-01	2.00E-01	2.00E-01	2.00E-01	2.00E-01
2.00E-01	2.00E-01	2.00E-01	2.00E-01	2.00E-01

SYMBOL PROBABILITY MATRIX (TRANSPOSED):

1.25E-01	1.25E-01	1.25E-01	1.25E-01	1.25E-01
1.25E-01	1.25E-01	1.25E-01	1.25E-01	1.25E-01
1.25E-01	1.25E-01	1.25E-01	1.25E-01	1.25E-01
1.25E-01	1.25E-01	1.25E-01	1.25E-01	1.25E-01
1.25E-01	1.25E-01	1.25E-01	1.25E-01	1.25E-01
1.25E-01	1.25E-01	1.25E-01	1.25E-01	1.25E-01
1.25E-01	1.25E-01	1.25E-01	1.25E-01	1.25E-01
1.25E-01	1.25E-01	1.25E-01	1.25E-01	1.25E-01

TABLE II
PARAMETERS OF HMM(2), ROUNDED TO THREE SIGNIFICANT DIGITS

NUMBER OF MARKOV STATES = 5

NUMBER OF SYMBOLS PER STATE = 8

INITIAL STATE PROBABILITY VECTOR:

1.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TRANSITION PROBABILITY MATRIX:

1.40E-01	2.35E-01	3.08E-01	1.24E-01	1.94E-01
1.40E-01	1.14E-01	2.99E-01	2.13E-01	2.34E-01
4.37E-02	3.20E-01	1.72E-01	1.27E-01	3.38E-01
9.73E-02	4.97E-01	1.53E-02	1.15E-01	2.75E-01
2.36E-01	2.49E-02	4.27E-01	2.82E-01	2.98E-02

SYMBOL PROBABILITY MATRIX (TRANSPOSED):

1.81E-01	1.22E-01	7.89E-03	1.48E-01	7.04E-02
1.39E-01	8.28E-02	3.23E-02	9.13E-02	1.33E-01
2.67E-02	1.60E-01	5.87E-02	1.08E-01	2.34E-01
1.79E-01	1.66E-01	2.18E-01	1.30E-01	5.97E-02
1.56E-01	1.58E-01	2.15E-01	2.09E-01	2.35E-01
1.19E-01	5.75E-02	1.11E-01	1.02E-01	1.03E-01
1.76E-01	1.32E-01	2.40E-01	6.61E-02	1.76E-02
2.37E-02	1.22E-01	1.17E-01	1.46E-01	1.47E-01

puted using the forward-backward algorithm. Fig. 2 shows a histogram of the natural logarithm of $dF_{22}(x)$ for $T = 25$. The observation sequences are thus matched to the posterior likelihood function. Fig. 3 shows a histogram of log $dF_{21}(x)$ for $T = 25$. In Fig. 3, then, O_T is mismatched to the likelihood function. As is clear from Figs. 2 and 3, the difference between the mean values of the log likelihood functions is about 1.4 standard deviations. Thus, the potential exists for using log $dF_{22}(x)$ to classify observation sequences; however, $T = 25$ is not long enough to classify with a high probability of detection (i.e., P_D) and a low false alarm probability (i.e., P_F).

A useful observation drawn from Figs. 2 and 3 is that the probability density function of log $dF_{2j}(x)$ is nicely approximated by the normal distribution. Let μ_{ij} and σ_{ij} denote the mean and standard deviation of log $dF_{ij}(x)$. Then, if $dF_{ij}(x)$ is log-normal, it is easy to show that μ_{ij}

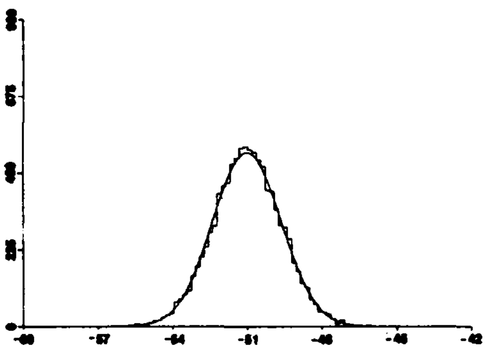


Fig. 2. Histogram of 10 000 values of log $dF_{22}(x)$ for $T = 25$. (The normal curve has the sample mean and variance given in Table III.)

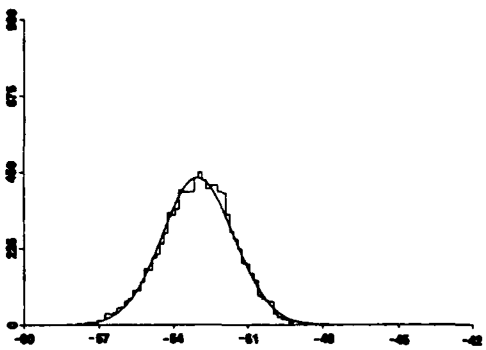


Fig. 3. Histogram of 10 000 values of log $dF_{21}(x)$ for $T = 25$. (The normal curve has the sample mean and variance given in Table III.)

and σ_{ij} are related to the moments $M_{ij}(k, T)$ by the formulas

$$\mu_{ij} = 2 \log M_{ij}(1, T) - (1/2) \log M_{ij}(2, T) \quad (28)$$

$$\sigma_{ij}^2 = \log M_{ij}(2, T) - 2 \log M_{ij}(1, T). \quad (29)$$

It is stressed that (28) and (29) hold exactly if and only if $dF_{ij}(x)$ is truly log-normal. For finite symbol HMM's, $dF_{ij}(x)$ is necessarily discrete, so that both (28) and (29) must be viewed as approximations. Sufficient conditions under which it may be proved that $dF_{ij}(x)$ is, in some sense, approximately log-normal are unknown. Although the central limit theorem is surely responsible for this log-normal behavior, it is not clear how to apply it in this setting.

Table III gives a comparison between the mean and standard deviations of log $dF_{2j}(x)$ estimated from 10 000 observation sequences O_T and those calculated from (28) and (29). This table shows good agreement between the approximations of (28) and (29) and the sample means and variances. It also establishes that observation sequences of length $T = 400$ are long enough to distinguish between $O_T \in$ HMM(1) and $O_T \in$ HMM(2) with high reliability. That is, the difference between the mean value of log $dF_{21}(x)$ and the mean value of log $dF_{22}(x)$ is about 5.2 standard deviations. Assuming log $dF_{21}(x)$ and log $dF_{22}(x)$ are normally distributed, as they appear to be,

TABLE III
COMPARISON OF TWO ESTIMATES FOR THE MEAN AND STANDARD DEVIATION OF $\log dF_{2j}(x)$ FOR $j = 1, 2$

	T	Mean Value		Standard Deviation	
		Sample	Eq. 28	Sample	Eq. 29
j = 1	5	-10.8	-10.6	0.95	0.71
	10	-21.3	-21.2	1.11	0.92
	15	-31.9	-31.8	1.24	1.10
	20	-42.4	-42.4	1.35	1.25
	25	-53.0	-52.9	1.47	1.38
	50	-105.8	-105.8	1.93	1.91
	100	-211.4	-211.5	2.62	2.67
	200	-422.6	-423.0	3.60	3.76
	400	-845.0	-845.8	5.09	5.30
j = 2	5	-10.1	-10.1	0.69	0.59
	10	-20.3	-20.3	0.90	0.84
	15	-30.6	-30.5	1.08	1.03
	20	-40.8	-40.8	1.23	1.20
	25	-51.0	-51.0	1.37	1.34
	50	-102.1	-102.1	1.92	1.90
	100	-204.4	-204.4	2.66	2.69
	200	-408.9	-409.0	3.77	3.80
	400	-818.0	-818.1	5.33	5.37

TABLE IV
PARAMETERS OF HMM(3)

NUMBER OF MARKOV STATES = 5

NUMBER OF SYMBOLS PER STATE = 8

INITIAL STATE PROBABILITY VECTOR:

1.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TRANSITION PROBABILITY MATRIX:

6.00E-01	4.00E-01	0.00E+00	0.00E+00	0.00E+00
0.00E+00	7.00E-01	2.00E-01	1.00E-01	0.00E+00
0.00E+00	0.00E+00	6.00E-01	4.00E-01	0.00E+00
0.00E+00	0.00E+00	0.00E+00	7.00E-01	3.00E-01
0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00

SYMBOL PROBABILITY MATRIX (TRANSPOSED):

9.00E-01	1.00E-01	0.00E+00	0.00E+00	0.00E+00
1.00E-01	6.00E-01	0.00E+00	0.00E+00	0.00E+00
0.00E+00	2.00E-01	3.00E-01	0.00E+00	0.00E+00
0.00E+00	1.00E-01	6.00E-01	1.00E-01	0.00E+00
0.00E+00	0.00E+00	1.00E-01	2.00E-01	0.00E+00
0.00E+00	0.00E+00	0.00E+00	4.00E-01	1.00E-01
0.00E+00	0.00E+00	0.00E+00	3.00E-01	6.00E-01
0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.00E-01

then classification using q_{subopt} has a probability of correct classification of 99% for a false alarm rate of 0.5%.

Computing the posterior likelihood function $f_2(O_T)$ for $T = 400$ requires $n^2T = 10\,000$ multiplications; thus, computational requirements for $f_2(O_{400})$ are small enough for practical application. Furthermore, the forward–backward algorithm for computing $f_2(O_T)$ is mathematically equivalent to a nested sequence of matrix–vector multiplications. Consequently, it is possible to reduce total computation time by the design of a "black box" to exploit this special structure in hardware.

B. Mixed Ergodic and Left-to-Right HMM's

HMM(3) is a five-state, eight-symbol left-to-right model whose parameters are given in Table IV. It has a structure that might conceivably arise in the SIIWR problem. Note that HMM(3) never leaves the fifth state once it is entered. Consequently, all sufficiently long observation sequences ultimately contain only the three symbols V_6, V_7, and V_8. Note also that the symbol V_8 occurs if and only if the fifth state has been entered. It follows that an observation sequence O_T containing the symbol V_8 and subsequently containing any of the five symbols V_1, V_2, V_3, V_4, or V_5 must have posterior likelihood zero, i.e., $f_3(O_T) = 0$. Other forbidden symbol sequences may also be noticed. It will be seen that these facts make $f_3(O_T)$ a powerful discriminator against ergodic observation sequences. To summarize briefly, this example will show that short observation sequences of quasi-stationary and transient HMM's look very different to the transient HMM recognizer. On the other hand, all observation sequences look somewhat alike to ergodic HMM recognizers.

When HMM(3) enters its fifth state, it becomes stationary and, consequently, significantly less interesting. Insight into the length of the transient portion of HMM(3) observation sequences is gained by estimating the first passage time of HMM(3) into its fifth state, that is, the number of transitions in the Markov chain before its fifth state is entered. The mean and variance of first passage times may be computed explicitly [6]; however, simulation was used here instead. In 10 000 observation sequences generated for HMM(3), it was found that the mean and standard deviation of the first passage time was 10.9 and 4.8, respectively. The least first passage time was three transitions, and the largest first passage time was 43 transitions. Thus, observation sequences for practical purposes become stationary for $t \geq 50$.

Fig. 4 and Table V clearly show that $dF_{33}(x)$ is a "well-behaved" distribution, even though HMM(3) is not ergodic. However, $dF_{33}(x)$ is not as closely approximated by a log-normal distribution as are $dF_{21}(x)$ and $dF_{22}(x)$, as evidenced by the discrepancy in Table V between the sample statistics and the statistics that would hold if $dF_{33}(x)$ were truly log-normal.

Ten-thousand observation sequences of HMM(1) and HMM(2) were generated and the posterior likelihood $f_3(O_T)$ was computed using the forward–backward algorithm. The observation sequences are thus mismatched to the posterior likelihood function. Table VI gives the number of sequences for which $f_3(O_T) = 0$. Better than 99% rejection of the simulated ergodic HMM observations was attained when $T = 10$, that is, when the observation sequences were about as long as the mean first passage time of HMM(3) into state 5. Total rejection of the 10 000 ergodic observations occurred for $T = 20$.

The ability of $f_3(O_T)$ to reject observations of $O_T \in$ HMM(2) is much more impressive than the $f_2(O_T)$ rejection of $O_T \in$ HMM(3). The lack of symmetry $F_{ij}(x) \neq F_{ji}(x)$ is striking in this instance. Table VII gives estimates of the mean and standard deviation of $\log dF_{23}(x)$, and Fig. 5 is a histogram of the case $T = 25$. The mean values of the 10 000 samples and those predicted by (28) agree very well; however, $dF_{23}(x)$ is not as well approximated by a log-normal as $dF_{22}(x)$ and $dF_{11}(x)$, as seen from the discrepancy in the sample versus the predicted standard deviations. In any event, it is clear by comparing Table VII to the lower half of Table III that $f_2(O_T)$ cannot

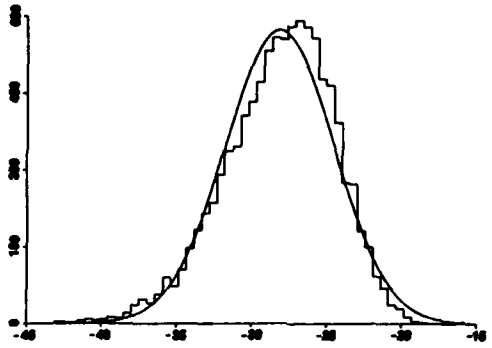


Fig. 4. Histogram of 10 000 values of $\log dF_{33}(x)$ for $T = 25$. (The normal curve has the sample mean and variance given in Table V.)

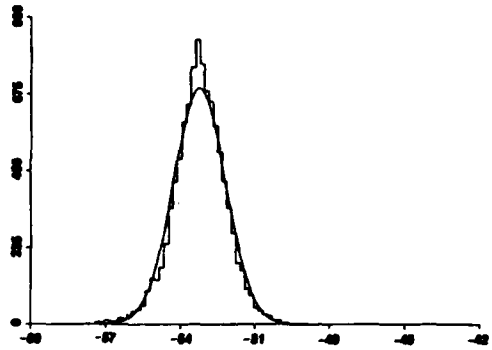


Fig. 5. Histogram of 10 000 values of $\log dF_{23}(x)$ for $T = 25$. (The normal curve has the sample mean and variance given in Table VII.)

TABLE V
COMPARISON OF TWO ESTIMATES FOR THE MEAN AND STANDARD DEVIATION OF $\log dF_{33}(x)$

T	Mean Value		Standard Deviation	
	Sample	Eq. 28	Sample	Eq. 29
5	- 5.6	- 4.9	1.92	1.13
10	-12.8	-11.8	2.30	1.91
15	-18.6	-18.2	2.72	2.33
20	-23.5	-22.6	3.22	2.29
25	-28.1	-26.3	3.61	2.15
50	-50.5	-47.2	4.59	2.75

TABLE VI
NUMBER OF $O_T \in$ HMM(i) FOR WHICH $f_3(O_T) = 0$, $i = 1, 2$

T	HMM(1)	HMM(2)
5	9389	9172
10	9937	9918
15	9997	9988
20	10000	10000

TABLE VII
COMPARISON OF TWO ESTIMATES FOR THE MEAN AND STANDARD DEVIATION OF $\log dF_{23}(x)$

T	Mean Value		Standard Deviation	
	Sample	Eq. 28	Sample	Eq. 29
5	10.8	-10.8	0.51	0.60
10	21.4	-21.4	0.91	0.89
15	- 32.0	-32.0	1.02	1.04
20	- 42.6	-42.7	1.04	1.15
25	- 53.2	-53.5	1.05	1.12
50	-106.4	-106.8	1.11	1.43

reliably distinguish $O_T \in$ HMM(3) from $O_T \in$ HMM(2) when $T = 50$. However, since the first passage time of HMM(3) is almost certainly less than $T = 50$, increasing the observation sequence length to improve reliability is not appropriate if the underlying intent is the classification of the transient portion of HMM(3).

C. Left-to-Right HMM with Noise

In this example, the effect of noise on the reliability of the q_{subopt} classifier is assessed for the left-to-right model HMM(3). The right way to study noise in finite symbol HMM's is to add the noise to the original time series $s(t)$ and then analyze the particular preprocessor under consideration to determine the noisy symbol sequence. However, no particular preprocessor is proposed here, and so we resort to modeling noise in much the same way that Shannon modeled noisy discrete memoryless channels [7]. This approach can give an indication of the successful classification rate as a function of the probable number of incorrect symbols in an observation sequence, but it cannot provide an assessment of the effect of signal-to-noise ratio on classification because such an assessment requires knowledge of the preprocessor.

Denote by h_{kj} the probability that the observation symbol V_k is altered to symbol V_j by the noise mechanism and define the $m \times m$ noise probability matrix $H = [h_{kj}]$. It is assumed that H is independent of the state of the Markov chain and of time t. Consequently, the output of a given HMM corrupted by noise is equivalent to another HMM that is noiseless. If $\lambda = (\pi, A, B)$ are the parameters of a given HMM with noise matrix H, the parameters of the equivalent noiseless HMM are $\tilde{\lambda} = (\pi, A, BH)$. The proof is straightforward: the product $b_{ik}h_{kj}$ is the probability that the state of the Markov chain is i and that symbol j is produced, given that symbol k was the output of the given HMM. The sum over k of $b_{ik}h_{kj}$ gives the component $\tilde{b}_{ij}$ of the equivalent noiseless HMM symbol probability matrix $\tilde{B}$. Clearly, $\tilde{b}_{ij}$ equals the (i, j) component of the product BH, so that $\tilde{B} = BH$.

The noise probability matrix H must be row stochastic, that is, every row sum must equal one. The HMM-generated symbol V_k is altered by noise to one of the available symbols, so that row k must sum to one.

Because H has row sums equal to one, the matrix $\tilde{B}$ is a valid symbol probability matrix for the equivalent noiseless HMM, that is, each row of $\tilde{B} = BH$ sums to one. We have

$$\begin{aligned}\sum_{j=1}^{m} \tilde{b}_{ij} &= \sum_{j=1}^{m}\sum_{k=1}^{m} b_{ik}h_{kj} \\ &= \sum_{k=1}^{m} b_{ik} \sum_{j=1}^{m} h_{kj} \\ &= \sum_{k=1}^{m} b_{ik} \\ &= 1.\end{aligned}$$

The worst case noise probability matrix, denoted H^0, has the constant entry $h_{ij}^0 = 1/m$ for all i and j. In this case,

$$b = \sum_{k=1}^{m} b_{ik} h_{kj} = \frac{1}{m} \sum_{k=1}^{m} b_{ik} = \frac{1}{m}.$$

Consequently, HMM's with noise probability matrix H^0 are indistinguishable. In fact, H^0 makes all HMM's statistically equivalent to the ergodic HMM(1) given in Table I.

Let $\Pr[V_i]$ be the relative frequency of occurrence of the symbol V_i in observation sequences of length T before the addition of noise. Thus, we have $\Sigma \Pr[V_i] = 1$. After alteration by noise, the probability of correct occurrences of V_i in O_T is then $\Pr[V_i] h_{ii}$. The probability that the symbol $O(t) \in O_T$ is correct is

$$D_T = \sum_{i=1}^{m} \Pr[V_i] h_{ii} \tag{30}$$

and the probability that $O(t)$ is incorrect is

$$E_T = 1 - D_T. \tag{31}$$

For the examples here, given a specific value of E_T, we choose the simple noise probability matrix H defined by

$$h_{ii} = 1 - E_T, \quad \text{all } i$$

$$h_{ij} = \frac{E_T}{m - 1}, \quad \text{all } i \neq j. \tag{32}$$

For this choice of H, D_T is independent of the actual values of $\Pr[V_i]$, as is clear from (30) and the fact that $\Sigma \Pr[V_i] = 1$.

Noise tends to make observations of all HMM's look like observations of HMM(1), and ergodic observation sequences tend to have forbidden symbol sequences for the left-to-right HMM(3). The first natural issue is therefore to determine how many forbidden symbol sequences occur as a function of the incorrect symbol probability E_T. Table VIII gives the results for various values of T and E_T, based on simulations of 10 000 observation sequences. It shows that forbidden symbol sequences are less likely for small T than for large T. This table also shows that noisy observations of HMM(3) do not have as high a proportion of forbidden symbol sequences as observations of HMM(1) and HMM(2), even for E_T = 10%, as can be seen by comparing Tables VI and VIII. One may conclude from Table VIII that E_T must be small and T must be short to minimize misclassification due to forbidden symbol sequences. For instance, if $T = 25$ and $E_T = 0.001$, the false dismissal probability is apparently at least 1.21%. Shorter T, however, causes smaller shifts in the statistics in the likelihood function, and thus increases the misclassification rate. Consequently, a tradeoff exists between short T and long T.

The total false dismissal probability can be expressed as the sum of the false dismissal probability due to forbidden symbol sequences and the false dismissal probability due to noise-induced shift in the statistics of the nonzero values of the posterior likelihood function. We examine the total false dismissal probability for HMM(4), which is the HMM equivalent to HMM(3) with the noise matrix H given by (32) with $E_T = 0.001$. The parameters of HMM(4) are given explicitly in Table IX.

TABLE VIII
NUMBER OF $O_T \in$ HMM(3) + NOISE FOR WHICH $f_3(O_T) = 0$ AT VARIOUS VALUES OF E_T

T	E_T = 0.1	0.01	0.001	0.0001
5	2194	236	23	1
10	3906	443	37	1
15	5305	651	64	11
20	6625	986	103	13
25	7643	1303	121	11
50	9643	2684	345	34

TABLE IX
PARAMETERS OF HMM(4), ROUNDED TO THREE SIGNIFICANT DIGITS

NUMBER OF MARKOV STATES = 5

NUMBER OF SYMBOLS PER STATE = 8

INITIAL STATE PROBABILITY VECTOR:

1.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TRANSITION PROBABILITY MATRIX:

6.00E-01	4.00E-01	0.00E+00	0.00E+00	0.00E+00
0.00E+00	7.00E-01	2.00E-01	1.00E-01	0.00E+00
0.00E+00	0.00E+00	6.00E-01	4.00E-01	0.00E+00
0.00E+00	0.00E+00	0.00E+00	7.00E-01	3.00E-01
0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E+00

SYMBOL PROBABILITY MATRIX (TRANSPOSED):

8.99E-01	1.00E-01	1.43E-04	1.43E-04	1.43E-04
1.00E-01	5.99E-01	1.43E-04	1.43E-04	1.43E-04
1.43E-04	2.00E-01	3.00E-01	1.43E-04	1.43E-04
1.43E-04	1.00E-01	5.99E-01	1.00E-01	1.43E-04
1.43E-04	1.43E-04	1.00E-01	2.00E-01	1.43E-04
1.43E-04	1.43E-04	1.43E-04	4.00E-01	1.00E-01
1.43E-04	1.43E-04	1.43E-04	3.00E-01	5.99E-01
1.43E-04	1.43E-04	1.43E-04	1.43E-04	3.00E-01

Denote by $F_{ij}^0(x)$ the cumulative distribution function

$$F_{ij}^0(x) = [F_{ij}(x) - F_{ij}(0)]/[F_{ij}(\infty) - F_{ij}(0)]. \tag{33}$$

Ten-thousand observation sequences O_T were generated from HMM(4) for $T = 25$. As given in Table VIII, 121 sequences resulted in zero posterior likelihood function values (that is, $f_3(O_T) = 0$) and the remaining 9879 nonzero values of $f_3(O_T)$ give the histogram shown in Fig. 6. By comparison to Fig. 4, it is clear that no significant difference between $\log dF_{34}^0(x)$ and $\log dF_{33}(x)$ is evident. Therefore, the misclassification rate due to noise-induced shifts in the statistics of $dF_{34}^0(x)$ is very small. The suboptimal classifier q_{subopt} for HMM(3) thus gives 98.8% correct classification and a 1.2% false dismissal probability when used with noisy observations characterized by $O_T \in$ HMM(4).

Because $E_T = 0.001$ in this example, each observation sequence O_{25} has probability 0.025 of having at least one incorrect symbol. Of 10 000 observation sequences, the expected number with at least one incorrect symbol is 250. Nearly half (121) contained forbidden symbol sequences and caused the only significant misclassification problem. The other half apparantly made no contribution to the probability of false dismissal.

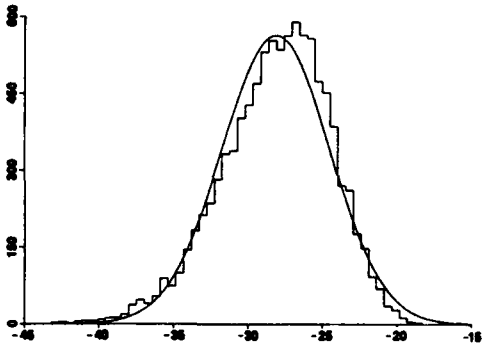


Fig. 6. Histogram of 9879 samples of $\log dF_{34}^{0}(x)$ for $T = 25$. (The normal curve has the sample mean $= -28.156$ and the variance $= 3.6167$.)

It would be desirable to be able to compute the moments of $F_{ij}^{0}(x)$ instead of $F_{ij}(x)$. Alternatively, it would be desirable to be able to compute the amplitude of the impulse (delta function) in $dF_{ij}(x)$ that seems to be present in the left-to-right HMM's considered here. In other words, if we write

$$dF_{ij}(x) = A_0\delta(x) + dF_{ij}^{0}(x), \qquad (34)$$

then an algorithm to compute A_0 directly would be worthwhile. Knowing A_0 and the moments of F_{ij} gives the moments of $F_{ij}^{0}(x)$. However, developing such an algorithm requires further work.

IV. Concluding Remarks

If the distribution $dF_{ij}(x)$ is approximately log-normal, the first two moments $M_{ij}(1, T)$ and $M_{ij}(2, T)$ can be used to develop a continuous approximation to $dF_{ij}(x)$. Simulations suggest that $dF_{ij}(x)$ is approximately log-normal whenever HMM(i) and HMM(j) are ergodic and nontrivial. (A "trivial" HMM is an HMM whose likelihood function $f(O_T)$ is constant.) A proof of approximate log-normality that relies on the central limit theorem is not obvious in the present context. If the distribution $dF_{ij}(x)$ is not approximately log-normal, the higher order moments $M_{ij}(k, T)$ are needed to develop reasonable continuous approximations to $dF_{ij}(x)$. The forward-backward moment algorithm presented in this paper computes these moments explicitly from the defining HMM parameter sets.

The use of the suboptimal classification statistic q_{subopt} in preference to the optimum statistic q_{opt} is probably not appropriate in many speech applications because of the ready availability of both likelihoods needed to form the likelihood ratio q_{opt}. Unfortunately, simulations are required to determine the ROC curves for q_{opt}. Consequently, for applications that require small incorrect classification probability and high probability of correct classification, very large simulations are necessary to confidently establish the required performance. An alternative in this case is to use the suboptimal statistic q_{subopt} because the ROC curves can be approximated in principle to any required accuracy without simulations.

The suboptimal statistic q_{subopt} is identical (to within a constant scale factor) to the optimal statistic q_{opt} when the problem is more akin to detection than to classification. That is, if the application is that of distinguishing the presence of a signal embedded in noise from the presence of noise alone, and if the HMM noise model is a "trivial" model as defined above, then the optimal detection statistic and q_{subopt} are identical. As a result, in this case, the moments of the optimal detection statistic can be computed using the forward-backward moment algorithm, and the ROC curves for the optimal detection statistic can be approximated to any required accuracy.

References

[1] L. R. Rabiner, S. E. Levinson, and M. M. Sondhi, "On the application of vector quantization and hidden Markov models to speaker-independent isolated word recognition," *Bell Syst. Tech. J.*, vol. 62, pp. 1075-1105, Apr. 1983.

[2] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst. Tech. J.*, vol. 62, pp. 1035-1074, Apr. 1983.

[3] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part I*. New York: Wiley, 1968, sect. 2.2.2.

[4] A. Papoulis, *Probability, Random Variables and Stochastic Processes*. New York: McGraw-Hill, 1965, p. 158.

[5] L. A. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 729-734, Sept. 1982.

[6] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*. Princeton, NJ: Van Nostrand, 1960, ch. 3.

[7] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, 1948.

[8] R. L. Streit and R. F. Barrett, "Frequency line tracking using hidden Markov models," *IEEE Trans. Acoust., Speech, Signal Processing*, this issue, pp. 586-598.

Roy L. Streit (SM'84) was born in Guthrie, OK, on October 14, 1947. He received the B.A. degree (with Honors) in mathematics and physics from East Texas State University, Commerce, in 1968, the M.A. degree in mathematics from the University of Missouri, Columbia, in 1970, and the Ph.D. degree in mathematics from the University of Rhode Island, Kingston, in 1978.

He was a Visiting Scholar in the Department of Operations Research, Stanford University, Stanford, CA, during 1981-1982, and an Exchange Scientist in the Signal Processing and Classification Group at the Defence Science and Technology Organization, Adelaide, South Australia, from 1987 to 1989. He joined the staff of the Naval Underwater Systems Center (then the Navy Underwater Sound Laboratory), New London, CT, in 1970. He is an Applied Mathematician and has published work in several areas, including towed array design, complex function approximation, semi-infinite programming, and applications of hidden Markov models. His current interests include image analysis, tracking problems, and training algorithms for neural networks.

Connection Machine Implementation Of Hidden Markov Models For Frequency Line Tracking

J. L. Muñoz and R. L. Streit

Chapter 14
Connection Machine Implementation of Hidden Markov Models for Frequency Line Tracking

José L. Muñoz* Roy L. Streit*

1 Introduction

In [1] Streit and Barrett explored the utilization of Hidden Markov Models (HMMs) for frequency line tracking and detection. This paper explores the implementation of [1] on the Connection Machine, a massively parallel SIMD machine, and its data parallel programming model. The reader is referred to [1] for specific details of the HMM and its application to frequency line tracking and detection. Frequency tracking is the estimation of frequency trajectories that are a result of a tone changing in frequency as a function of time.

As presented in [1] HMM processing produces two principal outputs (1) the Viterbi track (a discrete track), and (2) the Mean Cell Occupancy track (a continuous track). The CM implementation of HMM produces these two tracks and augments them with an additional probability field display, not previously explored, representing quality of the track estimate.

2 Background

It is necessary to first discuss the elements of HMM in order to present its particular implementation details on the CM. The frequency space is divided into *gates*, a contiguous set of FFT frequency bins, with one signal allowed in a gate. Input to HMM is a sequence of measurements, $z[t]$, over time within a gate, where a measurement represents an estimated frequency state. A measurement at frequency state i is said to exist if the magnitude of the signal at frequency bin i is greater than a predetermined threshold and is greater than the magnitude of other frequency bins. Such a measurement is said to be in the HMM state i. If no magnitude within the gate meets these criteria then the measurement is said to be in the HMM zero state. A *batch* of size T consists of a sequence of these measurements, Z, over T

*Naval Underwater Systems Center, New London, CT 06320, e-mail: munoz@nusc.arpa.

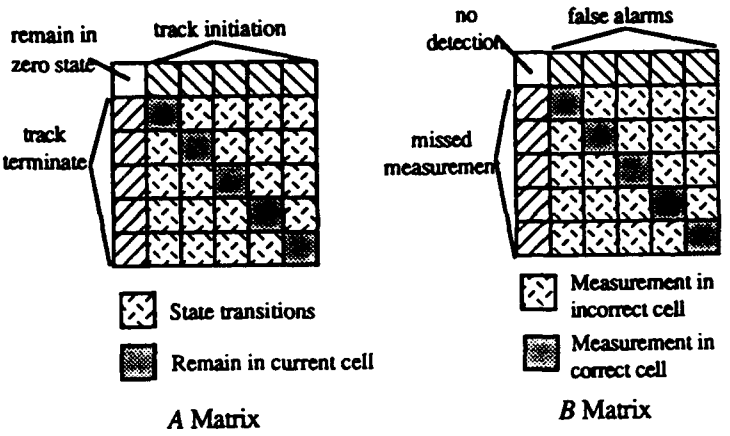


Figure 1: Characteristics of A and B matrices.

consecutive FFT samples, i.e., $Z = z[0], z[1], z[2], \ldots, z[T-1]$ where $z[0]$ represents the *oldest* measurement and $z[T-1]$ the *current* measurement.

Other key elements of HMM consist of an A matrix and a B matrix each of size $(n+1) * (n+1)$, where n represents the number of HMM states (the one is required to include the HMM zero state). The A matrix is the transition probability matrix of a Markov chain that represents the likely extent of the frequency fluctuations, track initiation and track termination. Element $a[i,j]$ is the likelihood that the signal will be in state j at time $t+1$ given that it is in state i at time t, where i and $j = 0, \ldots, n$. The B matrix represents the connection between the measurement at time t and the underlying state at time t. Element $b[i,j]$ is the likelihood that the measurement was made at state j given that the signal was actually in state i, where i and $j = 0, \ldots, n$. Finally, an initiation vector π represents the signal likelihood at time 0. Element $\pi[j]$ is the probability that the Markov chain is at state j at time $t = 0$ where $j = 0, \ldots, n$. The π vector is typically an A row. It is via the π vector that one batch of data is associated with the next batch of data. A π vector for the new batch is based on information obtained from the previous batch. Characteristics of the A and B matrices can be seen in Figure 1. Details on calculating A and B can be found in [1]. The B matrix is a function of the signal-to-noise ratio (SNR), and therefore a particular HMM frequency tracker is SNR specific.

The CM implementation has two principal sections: (1) initialization and (2) processing. In the initialization section, processing and display geometries are defined and the A and B matrices are calculated and loaded into

the CM. The processing section is divided into two parts, one for Viterbi and the other for Mean Cell Occupancy (MCO). They share a common display and data generation section.

3 Approach and Observations

The CM is a data parallel architecture and consequently performs best when it can work on massive amounts of data in parallel. The approach, therefore, attempts to maximize the potential for data parallelization.

Because the gates are independent of each other, parallelization among the gates is straightforward and such a parallelization would work just as well in an MIMD architecture. The opportunity for data parallelization exists only within the gate and among the various time steps. Both the Viterbi and MCO are dynamic programming models with a significant sequential dependence along the time domain. Consequently, it is not possible to capitalize on any parallelization in the time domain with the algorithms as described in [1]. Therefore, the initial effort focused on parallelizing data at each time instant within a gate.

This was achieved by replicating data within a *segment*, where a segment consisted of the gated data plus the zero state. This provided the ability to take advantage of vector multiplications and/or additions wherever possible. To exploit the parallel computation a transposed version of the A matrix was required, A^T, in order to facilitate working with the π vector and with (10). Hence, the π vector is obtained from columns of A^T.

Because *send* communication is about twice as fast as *get*, every attempt was made to avoid a get operation (indeed, none are used). Strict interpretation of the algorithms imply a *get* operation, either getting a value from a past or future time step. Therefore, it was found necessary to implement an equivalent form of the backward recursions so that the data for the previous iteration (i.e., past time step) was calculated from the current time step (details may be found in the algorithm description).

The CM supports two types of *send* operations: NEWS (or nearest-neighbor) *sends* and general communication (or remote) *sends*. For the virtual processor (VP) ratios and data element sizes typically encountered in the application, the performance timings of remote *sends* to NEWS *sends* was found to be 2:1. Therefore, NEWS *send* is the preferred communication method. However, information from any particular time step to its "past" or "future" neighbor nominally requires remote communication since we do not assume any a priori relationship between the measurements $z[t]$ and $z[t+1]$ or $z[t-1]$. In order to replace the remote *send* with a NEWS *send*, the data is first replicated everywhere within the segment and then a NEWS *send* is executed. Data replication was accomplished via scan_with_add "upward"

followed by scan_with_copy "downward" operations within the defined segments. Note that the remote forms of the *send* would have required (1) obtaining the state in the "future" or "past" time frame which is to receive the data, (2) either calculating the send address *in situ* or from a predefined table of entries (via *aref*), and (3) then performing the *send* operation. Execution timings of each of these approaches verified, although by a narrow margin, that the replicate/NEWS send is the preferred method over the remote *send*. Once the data is transferred to the appropriate time frame it is guaranteed that the correct state has immediate local access to the data since all states have identical copies of the data. This made the implementation of (2), (3), (5), (8) and (10) straightforward.

3.1 Geometry and Initialization

A 3-D geometry is used for the processing, as shown in Figures 2a and 2b. The x-axis performs double duty; both frequency and state information is maintained along the x-axis. The y-axis represents time, with the oldest data existing at $y = 0$ and the newest data at $y = (T - 1)$ where T is the batch size (i.e., number of FFT samples to be processed as a unit). The z-axis represents, along with the x-axis, state space. The size of the geometry is:

$$([\text{frequency}+\text{frequency}/(n\_\text{states}+1)])*([\text{T}])*([n\_\text{states}+1\ ])/2$$

where the square brackets represent the smallest power of two that is greater than or equal to the value being evaluated. The value *n_states* is the size of the gate used for the HMM processing with one added to account for the zero state. The frequency axis is augmented by one zero state per gate (represented by the frequency/(n_states+1) term). Finally, the division by 2 is required to account for the FFT process.

The z-axis is loaded, from the front-end, at $y = 0$ with copies of the A, B, and A^T matrices that have been previously calculated, with each segment getting the same copy. Once loaded the natural log equivalent of the matrix is evaluated to facilitate Viterbi processing and an initial π vector is obtained as column zero of A^T. All of the matrices are loaded such that rows are along the x-axis and columns are along the z-axis. Finally, all of this data is replicated along the y-axis via a *spread* operation. The x-axis is segmented into size of (nstate+1) with the frequency information loaded into the cells (no frequency data is entered at the zero state cells). Previously evaluated measurements are first scrolled "up," i.e., towards older time with the oldest time information removed from the batch. New FFT data is inserted at $T - 1$ from the front-end host and a measurement is performed at each gate in parallel with the measurement subsequently copied along the z-axis.

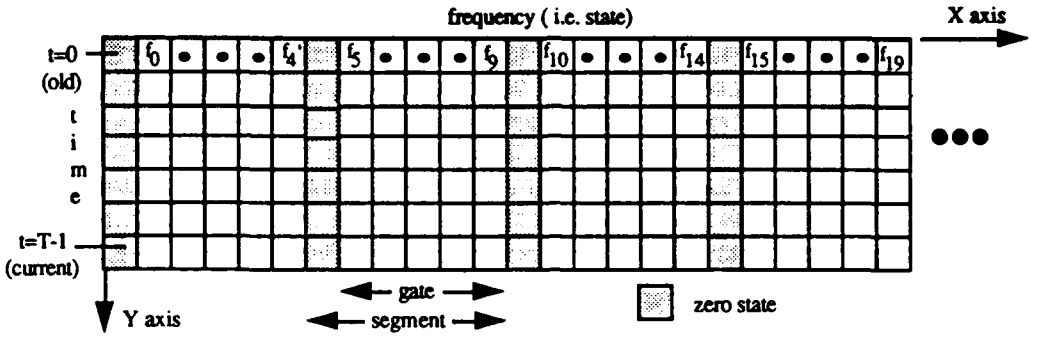


Figure 2a: X-Y plane of HMM compute geometry.

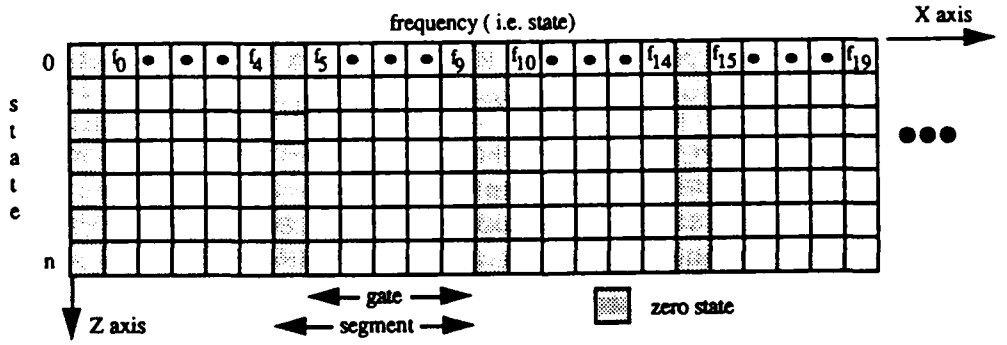


Figure 2b: X-Z plane of HMM compute geometry.

4 Processing

4.1 Viterbi Processing

4.1.1 Viterbi forward processing

Calculation of the Viterbi track as presented in [1] begins with the following algorithm:

Forward Viterbi : $z[t]$, measurement at time t

$$t = 0: \quad \delta[0,j] = \ln \pi[j] + \ln b[j, z[0]] \quad j = 0, \ldots, n \tag{1}$$

$$\Psi[0,j] \quad \text{arbitrary}$$

$$t = 1, \ldots, T-1:$$

$$\delta[t,j] = \ln b[j, z[t]] + \max\{\delta[t-1,i] + \ln a[i,j]\} \quad i = 0, \ldots, n \tag{2}$$

$$\Psi[t,j] = \text{argmax}\{\delta[t-1,i] + \ln a[i,j]\} \quad i = 0, \ldots, n \tag{3}$$

where argmax returns the smallest index for the maximum attained. The natural logarithm is used to control potential numerical underflow.

From (1) through (3) we see that Viterbi forward processing starts at time $t = 0$ (i.e., $y = 0$), selects a column of $\ln B$ as identified by $z[0]$ and adds the π vector to that column resulting in $\delta[0,j]$. This δ is first replicated and then *sent* down to time = 1 into the CM field *prev_delta*. For time > 0, the prev_delta vector is added to each column of $\ln A$. A maximum is evaluated along the z-axis (columns) as well as the smallest index at which the maximum occurred (argmax function) is obtained. This row vector of maximums is transposed into a column vector and replicated everywhere in the segment. $z[t]$ is used to select a column from $\ln B$ and the vector of maximums is added to it, creating a $\delta[t,j]$ for this time step which is subsequently replicated within the segment. The results of the argmax evaluation are stored in vector Ψ for use by the backward processing. The $\delta[t,j]$ just calculated is then sent down to the next time step (more recent) into its prev_delta value and the process is repeated for $T-1$ time steps.

4.1.2 Viterbi backward processing

The following algorithm is used for Viterbi backward processing:

Backward Viterbi :

$$t = T-1: \quad V[T-1] = \text{argmax}\{\delta[T-1,j]\}, \tag{4}$$

$$t < T-1: \quad V[t] = \Psi[t+1, V[t+1]], \quad t = (T-2), \ldots, 0. \tag{5}$$

At this point $\delta[t,j]$ and the Ψ vector are defined for all time steps. The Viterbi backward processing begins at time = $(T-1)$ by determining the

index at which $\delta[t,j]$ is a maximum resulting in $V[T-1]$. Once initialized it is possible to calculate V for the previous time steps from the current time step by rewriting (5) as:

$$V[t-1] = \Psi[t, V[t]]. \tag{6}$$

Consequently, it is only necessary to index into the current Ψ vector using the current V to obtain the V for the previous time step. The value obtained is *sent up* to the V field of the previous time step. This process is repeated $T-1$ time steps. The Viterbi track estimate for the batch of size T is taken to be V at time $= T/2$.

4.1.3 Display and new data

Once the Viterbi track has been evaluated the V value at time $= T/2$ is displayed on the CM frame buffer in a *waterfall* display, i.e., newest data at the top with the oldest data "falling off of the display" at the bottom. Each HMM gate is associated with a particular color. (More generally, the V value at any particular time could be selected for display; time $= T/2$ was chosen on the basis of estimated track variance.)

Following data display a new p vector is selected to associate the next batch with the current batch. The V field at time $= 0$ is used to index into the A^T matrix to select the appropriate column to be used as the next p vector. The entire Viterbi process is then repeated when new FFT data is made available.

4.2 Mean Cell Occupancy Processing (MCO)

4.2.1 Calculation of alpha (forward processing)

MCO processing is analogous to Viterbi processing. The algorithm for forward processing is:

Forward probabilities : $z[t]$, measurement at time$t$

$$t = 0: \quad \alpha[0,j] = \pi[j] * b[j, z[0]]/s[0], \quad j = 0, \ldots, n \tag{7}$$

$$t = 1 \ldots T-1:$$

$$\alpha[t,j] = b[j, z[t]] * \sum_{i=0}^{n} (\alpha[t-1,i] * a[i,j])/s[t], \quad j = 0, \ldots, n. \tag{8}$$

The quantity $s[t]$ in (7) and (8) is a scale factor required to control numerical underflow in the α's.

For time $= 0$: An α vector at time $= 0$ is initially calculated by multiplying the π vector with a column of the B matrix as identified by the measurement $z[0]$. A scale factor, $s[0]$, using this α is evaluated and used to

scale the α just obtained. This same scale factor is later used in the β (backward) processing. The resulting α is then replicated within the segment and subsequently *sent down* to the field *prev_alpha* in the next time frame (i.e., more recent time).

For time > 0: The *prev_alpha* vector received is multiplied by the A matrix producing ($n$state+1) column vectors. A sum along the columns is performed and the resulting row vector is transposed into a column vector that is then replicated everywhere in the segment. This is then multiplied by a column from the B matrix as identified by the measurement $z[t]$ producing $\alpha[t,j]$. A scale factor, $s[t]$, is calculated and used to scale the $\alpha[t,j]$ just obtained and saved for later use by the β processing. The resulting $\alpha[t,j]$ is then replicated along the x-axis and subsequently *sent down* to the next time frame with the process repeated for $T-1$ time steps.

4.2.2 Calculation of beta (backward processing)

The beta, or backward processing, follows the alpha processing. The algorithm is:

Backward probabilities : $z[t]$, measurement at time t

$$t = T-1: \qquad \beta[T-1,j] = 1/s[T-1] \tag{9}$$

$$t = T-2,\ldots,0$$

$$\beta[t,j] = \sum_{i=0}^{n}(\alpha[j,i] * b[i,z[t+1]] * \beta[t+1,i])/s[t], j = 0,\ldots,n. \tag{10}$$

As before, $s[t]$ in (9) and (10) is a scale factor required to control numerical underflow. The same scale factor previously calculated for $\alpha[t,j]$ is used for the β's.

Beta processing goes backward in time from $(T-1)$ to zero. β for time $= (T-1)$ is set to 1.0 scaled by the scale factor obtained at $T-1$. Once initialized in this manner it is possible to calculate β for the previous time step from the current time step by rewriting (10) as:

$$\beta[t-1,j] = \sum_{i=0}^{n} a[j,i] * b[i,z[t]] * \beta[t,i]. \tag{11}$$

The beta processing therefore proceeds by selecting the appropriate column from the B matrix using $z[t]$ as the index and replicating that column everywhere within the segment. This is then multiplied by the β for the current time step (which has been previously scaled). This result is then multiplied by the A^T matrix producing ($n$state+1) column vectors. A summation along each column is then performed and the resulting row vector is subsequently transposed to form a column vector. The resulting vector is

then replicated everywhere within the gate along the x-axis and is then *sent up* as the β for the previous time step. Once it is received by the previous time step it is scaled using the scale factor for that time step. The process is then repeated $T-1$ times.

4.2.3 Gamma and MCO processing

Following the calculation of the α and the β the next step is to calculate the state occupancy probabilities, γ's, and the Mean Cell Occupancy. The algorithm is:

State occupancy probabilities (track quality):

$$\gamma[t,i] = (\alpha[t,i] * \beta[t,i]) / \sum_{i=1}^{n} (\alpha[t,i] * \beta[t,i]) \tag{12}$$

Mean Cell Occupancy (M):

$$M[t] = \sum_{i=1}^{n} (\gamma[t,i] * f_c[i]) / \sum_{i=1}^{n} \gamma[t,i], \tag{13}$$

where $f_c[i] = (f[i] + f[i+1])/2$

$$\sigma^2[t] = \sum_{i=1}^{n} \gamma[t,i][f_c[i] - M[t]]^2 / (1 - \gamma[t,0]). \tag{14}$$

The γ calculations are executed in parallel for all time steps.

The α and the β vectors are multiplied and the resulting vector stored temporarily. The elements of this resulting vector are summed and subsequently applied to the stored vector via a division operation resulting in a $\gamma[t,i]$ for each time step. This value is then copied as a column vector into *gamma_pi* for subsequent use as the π vector for the next batch. The γ column vector is then transformed into a row vector via a transposition. The resulting row vector is then multiplied by a row vector representing the center frequency of the ith cell, f_c, that was previously calculated and stored. Elements $1 \ldots n$state of the resultant row vector are added and the resultant is divided by $(1.0 - \gamma[t,0])$. The result is the mean cell occupancy, MCO, a single variable that is then replicated everywhere in the segment. $(f_c - MCO)^2$ is then calculated and subsequently multiplied by $\gamma[t,i]$ with the resultant row vector elements $1 \ldots n$state summed and divided by $(1.0 - \gamma[t,0])$ terminating in a square root operation resulting in the standard deviation σ, completing the γ calculations.

4.3 Mean Cell Occupancy Display

The π vector for the next batch is obtained by selecting the second to oldest time slot (time = 1) and copying over its *gamma_pi* field into the p vector.

Next, the time slot at which data is to be displayed is selected, $T/2$. The σ value obtained is replicated everywhere in the segment and is then subtracted from MCO. The resulting value is compared to the x-coordinate with points greater than or equal to the x-coordinate set to a color particular to that gate. σ is then added to MCO and that value is compared to x-coordinate, with points less than or equal to the result set to the gate color. The result is a series of points set to a color if they are within σ of MCO. This data is then sent to the framebuffer for display.

4.4 Probability Field Display

Equation (12) results in the γ vector. As defined, each element of γ is bounded by $[0...1]$ and represents for each HMM state i, the likelihood that the signal is in that particular state. γ can therefore be used as an estimate of track quality. The track quality is displayed for the non-zero states in grey-scale at each gate location with a greater confidence in the track estimate appearing as brighter areas on the display.

Using this information an analyst can more intelligently interpret the track trajectories displayed. Initial experimentation with the probability field has been found to be very effective as an image enhancement device in the context of presenting track trajectory information and shall be further explored.

5 Performance

HMM was first implemented on a Sun 4/260 to act as a baseline. FFT size, number of HMM states and batch sizes were selected that were felt to be representative for problems of interest resulting in CM VP ratios of 16, with 25 gates. The HMM algorithms described above were implemented on a CM-2a with 8192 processors and 32-bit hardware floating point, Sun 4/260 front-end, executing Version 5.2 of the CM software. The CM code was instrumented to obtain performance information. A detailed geometry was defined by analyzing the data movement patterns. Little data is moved along the y-axis and what is moved uses strictly NEWS communication that happens only once in a cycle; along the z-axis there is a requirement for data replication for various operations; the majority of the data movement occurs along the x-axis with requirements for data replication and orientation. As a result, the CM's defined detailed geometry instruction was used with weights of 10, 1 and 8 for the x, y and z axes, respectively (only their relative ordering is significant, the actual values are not important). Table 1 provides timing information obtained from the Sun4 execution and Table 2 CM execution timings for both default geometry and the detailed geometry.

Sun 4/260 Times(sec)	Single Gate	25 Gates
Viterbi	0.03	0.75
MCO	0.05	1.25

Table 1: Processing time on Sun4 (display and signal generation times not included).

CM (VP=16) Times (sec)	Baseline Geometry	Detailed Geometry	Improvement
Viterbi	1.42	1.28	10%
MCO	1.94	1.59	18%

Table 2: Processing times on CM. 25 gates (display and signal generation times not included).

Clearly, for a 25 gate problem sequential execution on a Sun4 outperforms the CM implementation. It would appear that the CM can outperform the sequential implementation only by virtue of executing multiple gates in parallel, or via the MIMD model. This would indicate that:

1. The particular implementation described in this paper was not able to take full advantage of any inherent data parallelism, or
2. There is insufficient data parallelism in the application.

The "break even" point between the sequential and parallel implementations is 43 gates for Viterbi and 32 gates for MCO.

5.1 Enhancements

5.1.1 Multi-line HMM

The CM implementation provided the opportunity to investigate enhancing HMM. As previously discussed, HMM requires that at most only one signal be present in a gate. A slight modification to the CM implementation provided a suboptimal approach supporting more than one signal in a gate. The modification consisted of staging the data so that the first stage has access to the complete frequency magnitude information as described in previous sections. Once a measurement determination has been executed by this first stage, the data at that frequency bin is removed, either by making it zero or replacing it with noise, and the altered data is then passed, via *spread* along the z-axis, to a second stage for processing.

Second stage processing is an exact copy of the first stage processing, the only difference is that the second stage does not have access to the complete data set. The second stage processing then proceeds to make a measurement using the altered data; it therefore has the opportunity to find a second line

in a gate if it exists. Viterbi and MCO processing then proceeds as before, with each stage using the measurements pertaining to it, and the stages running in parallel.

The performance of multi-line HMM depends on (1) the frequency stability of the lines within a gate, and (2) the SNR difference between the lines. Preliminary results obtained have found such an implementation to provide adequate capabilities when the SNR of the two lines are sufficiently different. An optimal HMM approach to multi-line tracking is presented in [4].

5.1.2 "Mostly Forward" HMM

The dynamic programming requirements of HMM result in an implementation that is $O(T)$ in computational performance, i.e., a problem of size $2T$ will take twice as long as a problem of size T, all other parameters being equal. Therefore, the sequential aspects of the problem are driving its performance since the data for $(t+1)$ or $(t-1)$ must be available before one can proceed with the forward or backward process.

In an attempt to moderate this behavior, the Viterbi algorithm was slightly modified. The initial batch, i.e., T time samples, was treated as previously described. However, subsequent batches did not update the π vector. In this manner it was not necessary to recalculate the $T-1$ previous $\Psi[t]$'s. It is only therefore required to calculate $\Psi[T-1]$ using $z[T-1]$ and $\delta[T-2]$, going "mostly forward." The backward Viterbi process remained unchanged.

This modification improved run time performance by over 50%, resulting in an execution time of 0.55 sec for the same size problem identified in Table 1. Preliminary results with this modification provided reasonable results, suggesting that updating the π vector may not be required (for comparable tracking accuracy, the batch size T must be increased slightly). Clearly, a similar modification could be made to the MCO tracker, with a comparable improvement in its run time performance, although such a modification has not been implemented at this time.

6 Conclusions

Clearly it is possible to implement a dynamic programming model, such as HMM, on the CM. Its performance will be strongly affected by the number of dependent steps required to solve the problem. While some improvements were identified that alleviated this "problem," such as the "mostly forward" approach, these improvements would be just as applicable to a purely sequential implementation such as that implemented on the Sun with an attendant improved run time performance.

While previous CM efforts at NUSC have achieved dramatic improvements by replicating data, it was not possible to effect a corresponding benefit with the straightforward intuitive implementation as described in this paper. Data replication did not perform as well as initially expected due to (1) no convenient method for replicating data within a segment other than combining a *scan_with_oper* followed by a *scan_with_copy* (where oper represents some arithmetic operation), and (2) the data reference patterns, as implemented, required several vector transpose operations requiring remote sends. Future efforts will focus on perhaps a different topology that would keep information in in-processor arrays and thereby avoid some of the NEWS communication (beneficial only if *aref* costs are less than *NEWS* communication); or an implementation avoiding vector transpositions. Neither of these approaches, however, is truly exploiting the data parallelism model as exemplified by the CM. What is required is an approach that considers the input measurement sequence space as a unit and treats the output probability field accordingly. This is indeed a topic for future work.

There also exists a need to enhance the HMM approach via (1) development of an HMM capable of handling more than a single line in a gate (the multi-line approach presented above is a viable "workaround" in the interim), (2) development of an HMM capable of handling lines that cross, (3) development of an HMM such that track termination would be sensitive to "direction," i.e., leaving the gate on the left or right, thereby enabling initiation of a new track at the adjacent gate, or a true loss of signal, (4) an implementation with adaptive gates, i.e., gates that are not assigned a priori but are defined where a measurement is made, *in situ*, with the measurement placed in the center of the gate, and finally (5) implementation of HMM that can take advantage of the amplitude and phase information inherent in the data [3].

In addition, the CM implementation provides the opportunity to run multiple SNR hypotheses in parallel (recall that the B matrix is sensitive to SNR). This could be accomplished by running each of the possible candidate SNRs along different dimensions of a CM geometry. The data fusion aspect of such a multiple model approach is an additional area of future investigation.

Experience garnered from this and related CM efforts at NUSC has resulted in the identification of CM capabilities for possible future consideration: (1) VP ratios at other than powers of 2, since a small change in the size of a problem can have significant effects on its performance, (2) the need for higher-order languages to support both implicit *and* explicit communication, (3) specification of other than cartesian geometries, e.g., spherical or cylindrical, that could *efficiently* utilize the processing and communication resources, (4) improved debugging aids, (5) addition of performance monitoring, code instrumentation (profiling), (6) efficient implementation

of a "segmented spread," i.e., from an identified coordinate copy a datum everywhere within a segment (*spread* works along an axis), and finally (7) application specific subroutine packages such as image processing, signal processing, etc.

Acknowledgements

The authors would like to especially recognize Mr. R. Bernecky for his significant contributions to this effort. His insight into the problem domain, as a result of similar work, and the many hours of discussions provided are appreciated. The authors would also like to thank Mr. R. Kneipfer for his suggestions on this manuscript.

References

[1] R. L. Streit and R. F. Barrett, "Frequency Line Tracking Using Hidden Markov Models," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. 38, No. 4, pp. 586-598, April 1990.

[2] A. V. Aho, J. E. Hopcroft, J. D. Ullman, "The Design and Analysis of Computer Algorithms," *Addison-Wesley Publishing Company*, pp. 67-69, 1974.

[3] R. F. Barrett and R. L. Streit, "Frequency Line Tracking Using Hidden Markov Models with Phase Information," Proceedings of the Second International Symposium on Signal Processing and Its Applications, 27-31 August 1990, Gold Coast, Australia.

[4] X. Xianya and R. J. Evans, "Multiple Frequency Line Tracking Using Hidden Markov/A Models," Technical Report EE9007, February 1990, University of New Castle, New South Wales, Australia.

ARTIFICIAL NEURAL NETWORK STUDIES

Foreword

Artificial neural networks (NNs) are specialized computer architectures for classifying multivariate feature sets. Paper [22] establishes that feed-forward NN architectures can implement asymtotically optimum classifiers, when properly trained, and that maximum likelihood estimation methods can be used as NN training algorithms. The statistical approach to NN classification and training is pursued much further in paper [23]. In this paper, particular attention is given to training algorithms suited to small data problems, that is, to problems in which only a small amount of training data for one or more classes is available. The approach enables general nonlinear discrimination, and it generalizes Fisher's classical method for linear discrimination. A maximum likelihood NN training algorithm is derived, using a variant of the Expectation-Maximization algorithm. Paper [24] shows how to use the output of the maximum likelihood training phase to develop maximum entropy estimates for the class *a priori* probabilities.

A Neural Network For Optimum Neyman-Pearson Classification

R. L. Streit

A NEURAL NETWORK FOR OPTIMUM NEYMAN-PEARSON CLASSIFICATION

Roy L. Streit

Naval Underwater Systems Center
New London Laboratory
New London, CT 06320

ABSTRACT

A three-layer feed-forward neural network (NN) that implements the optimum Neyman-Pearson (N-P) classifier is described. This NN is useful whenever it is appropriate to characterize (1) input classes as multivariate random variables, and (2) input data vectors as realizations of one of the multivariate random variables. The purpose of the NN is thus simply to compute the conditional likelihoods necessary for the N-P classifier. Because the N-P classifier is optimal, the classification performance of the NN is optimal too. Therefore, **three-layer feed-forward NN classifiers can equal but not exceed the performance of the well known N-P classifier.**

The optimal N-P classifier requires multivariate probability density functions (PDFs) characterizing the input classes. Class PDFs are approximated (arbitrarily closely) by mixtures of multivariate Gaussian PDFs. Supervised training of the class PDFs from input data vectors is, thus, equivalent to training the NN. Maximum likelihood training of the PDFs is performed by the EM algorithm (or by any other suitable optimization method).

1. INTRODUCTION

The discussion of NNs in this paper is limited to the fundamental problem of classification, or recognition, of a given input vector as one of several possible outcome pattern classes. For the purposes of this paper, then, a NN is a specialized device for pattern recognition. The discussion is also limited to conventional feed-forward three-layer NNs. A node is a computational unit that forms a weighted sum of all its inputs and then passes the result through a nonlinearity. The nonlinearity is characterized by a real-valued nonlinear function h, together with a threshold τ. Thus the output of a node is the numerical value of the function h when its argument is the weighted summation minus the threshold τ. Different nonlinear functions h have been proposed; typically, they are monotone non-decreasing functions, and are asymptotically constant for large arguments (positive and negative). The weights used to form the weighted sum within a node are called the interconnection weights. They are uniquely defined for each interconnection, that is, between each communication link between node and node and between node and input. The weights can be positive, negative, or zero. The problem of assigning appropriate interconnection weights and nodal thresholds is known as the training problem.

The maximum likelihood methodology presented in this paper provides a unified theoretical viewpoint for understanding NNs and for developing computationally effective training algorithms. The methodology is rooted in a classical statistical and mathematical framework that is esthetically

satisfying and potentially very significant for the future development of NN architectures. One significant advantage of this methodology is that the important "exclusive-or" classification problem is easily solved.

N-P classification requires knowledge of the PDF of the input data vector X, conditioned on each of the various class membership hypotheses. For M outcome classes, N-P classification requires M conditional PDFs. Let $g_j(X)$ denote the conditional PDF of the j-th class, and let α_j denote the *a priori* probability of class J. Then the N-P classification estimate for the input vector X is the class j* for which

$$g_{j^*}(X) = \max_{1 \le j \le M} \alpha_j g_j(X). \qquad (1)$$

Since $\{\alpha_j\}$ are typically unknown, they are often treated as free parameters and are used to adjust the probability of incorrect classification. N-P classification is optimum in the sense that, for a given probability of incorrect classification, the probability of correct classification is maximized. Hypothesis testing is described in many places, e.g., [1].

The NN presented in this paper requires that the output layer have M nodes, and that each output node evaluate one of the outcome class conditional PDFs. Estimation of the conditional PDFs is therefore a central issue. In this paper, conditional PDFs are approximated by mixtures of multivariate Gaussian PDFs. (It is well known that very general PDFs can be approximated to arbitrary accuracy by Gaussian mixtures.) Both the training algorithm and the NN structure are affected by this approximation. The PDFs of the Gaussian components in the mixtures are computed by the first two, or hidden, layers.

Training the NN presented in this paper is undertaken on the training data to obtain estimates of the outcome class mixture PDFs. Knowledge of the class PDFs enables the design of a NN architecture with a performance that can be readily tested without building the NN. The supervised training algorithm is an established maximum likelihood algorithm for estimating mixture PDFs known as the EM (Estimation and Maximization) algorithm [2].

The NN described in [3] is in some ways similar to the NN described in this paper. For instance, both approaches model the output classes statistically using mixtures of multivariate Gaussian PDFs. However, they also differ in significant ways. In [3], the number of first layer nodes equals the number of training vectors, and the interconnection weights between the inputs and the first layer nodes are proportional to the components of the training vectors. In this paper, however, the number of first layer nodes is independent of the size of the training set, and the interconnection weights are nontrivial functions of the training vectors.

2. NEYMAN-PEARSON NEURAL NETWORK DESCRIPTION

2.1 The Output Layer

In the context of this paper, an output node is conceptually identified with an outcome class hypothesis. The defined purpose of an output node is thus to compute a numerical value equal to the likelihood of the input vector X, conditioned on an outcome class hypothesis characterized by a multivariate

mixture PDF. The NN performs N-P classification by selecting the outcome class of the output node having the largest numerical value.

It is assumed in this section that the multivariate PDFs for every component population in the various class mixture PDFs are computed by the second hidden layer. The design of the hidden layers to accomplish the required PDF evaluations is described for Gaussian components in Section 2.2.

Let G denote the total number of different components in the outcome class mixture PDFs, that is, each of the M possible outcome classes is comprised of some combination of the G population components. The number of required second hidden layer nodes is therefore G. Let $p_i(X)$ denote the multivariate probability density function of the i-th mixture component, and let π_{ij} denote the proportion of population i in outcome class j. The numbers π_{ij} are non-negative and satisfy the equations

$$\sum_{i=1}^{G} \pi_{ij} = 1, \qquad j = 1, \ldots, M. \tag{2}$$

It follows that the output of the j-th output layer node is given by

$$g_j(X) = \sum_{i=1}^{G} \pi_{ij}\, p_i(X), \qquad j = 1, \ldots, M. \tag{3}$$

The output from the second hidden layer, given the input vector X, is the set of numbers $\{p_i(X)\}$. Consequently, to enable the NN to compute the M class PDFs $\{g_j\}$ in (3), the interconnection weight between the i-th node in the second hidden layer and the j-th output node must be the mixing proportion π_{ij}. Estimating the class PDFs thus sets these NN weights directly.

Because the NN is intended to be the optimum N-P classifier, the nonlinear function of an output node is given by the *linear* function

$$h_3(x) = \alpha\, x, \tag{4}$$

for all real values x, where α is the *a priori* probability of the output class. The output node thresholds are zero for N-P classification.

2.2 Gaussian Component Architecture For The Hidden Layers

The purpose of a node in the second hidden layer is to compute and pass to the output layer a numerical value equal to the likelihood of the input vector X (of length N) under the hypothesis that it is a realization of a component population in one of the mixture outcome classes. Consequently, in the case that the component population is a multivariate Gaussian with mean vector μ and positive definite covariance matrix Σ, the output p(X) of a second layer node has the mathematical form

$$p(X) = \left[(2\pi)^N\, |\Sigma| \right]^{-1/2} \exp\left\{ -\frac{1}{2} (X - \mu)^T\, \Sigma^{-1}\, (X - \mu) \right\}, \tag{5}$$

where $|\Sigma|$ denotes the determinant of the matrix Σ. The mean vector μ and covariance matrix Σ are estimated during the NN training phase. One way to compute p(X) within the confines of the hidden layers of a conventional NN architecture is described in this section.

The covariance matrix Σ of the Gaussian distribution (5) is positive definite, so it has the Cholesky factorization $\Sigma = L\, L'$, where the matrix L is lower triangular (i.e., the entries of L above the diagonal are zero) and

has positive diagonal entries, and where ' denotes the matrix transpose. Substituting into (5) and simplifying gives

$$p(X) = \left[(2\pi)^N |\Sigma| \right]^{-1/2} \exp\left\{ -\frac{1}{2} \| L^{-1}X - L^{-1}\mu \|^2 \right\}, \qquad (6)$$

where $\| \cdot \|$ is the usual Euclidean norm on R^N.

The second layer nodes are required to have no inputs in common. This requirement probably increases the number of first layer nodes in the overall NN, but it does not fundamentally alter the NN structure. That is, a fully-connected NN in which certain of the interconnection weights between the hidden layer nodes are set to zero satisfies the requirement.

The hidden layer architecture from the input layer to a second layer node is characterized by the following quantities:

1. Nodal nonlinear functions in both layers
2. Nodal thresholds in both layers
3. Number of first layer nodes
4. Interconnection weights between the input and first layer
5. Interconnection weights between first and second layers.

In the following paragraphs, these quantities are explicitly defined in terms of the mean μ and covariance Σ of the Gaussian PDF.

Implementation of expression (6) within the hidden layers requires a different nodal nonlinearity in each layer. The nonlinear function required for a first hidden layer node is given by

$$h_1(x) = |x|^2. \qquad (7)$$

The first layer nonlinear function is thus especially simple to implement. The nonlinear function required for a second hidden layer node is given by

$$h_2(x) = \left[(2\pi)^N |\Sigma| \right]^{-1/2} \exp\left[-\frac{1}{2} x \right]. \qquad (8)$$

These nodal nonlinear functions are different from those typically utilized in NN applications.

Expression (6) yields a mathematical expression for the thresholds of the first hidden layer nodal nonlinearities in terms of the trained parameters μ and Σ characterizing the Gaussian component. Specifically, the threshold τ_i for the i-th first layer node is given by

$$\tau_i = \left[L^{-1} \mu \right]_i, \qquad i = 1, \ldots, N. \qquad (9)$$

Thresholds for the second layer nodes are identically zero.

The expression (6) implies that the number of first layer nodes connected to a given second layer node is equal to N, the dimension of the input vector X. Because of the requirement that the G second layer nodes have no input first layer nodes in common, the total number of first layer nodes in the overall NN is equal to the product N·G.

The expression (6) also implies that the interconnection weights between the components of the input vector X and the nodes of the first hidden layer be proportional to the components of the inverse of the Cholesky factor L. This follows from the nature of the argument of the norm $\| \cdot \|$ in (6). The

interconnection weight between i-th input and the j-th first layer node is therefore the (i,j) component of the inverse of L. About half of these weights are zero because L^{-1} is upper triangular. The interconnection weights between the two hidden layers are all unity, because the sum (implicit in the squared norm in (6)) is unweighted.

3. TRAINING THE NEURAL NETWORK

Supervised training in the context of this paper means that the correct outcome class of each data vector in the training set is known beforehand. Thus, an outcome class is known only from a collection of data vectors, and training proceeds by using these vectors to estimate the defining parameters of the outcome class mixture PDF. The algorithm proposed in this paper is the EM algorithm that was originally described in [2]. The EM algorithm is applicable to very general mixture PDF estimation problems, but it takes on an especially simple form when applied to multivariate Gaussian PDFs. Details of the EM algorithm for Gaussian and non-Gaussian PDFs are given in [2] and [4].

G denotes the maximum number of different component populations in the mixture PDF of a particular outcome class. If G is initially unknown, it must be selected in some fashion and adapted as necessary. The number G is not estimated by the EM algorithm. After training is completed, however, G can be changed as desired, and training restarted.

Let $\mathcal{X}$ denote the set of data vectors in the training set corresponding to a given outcome class, and let $T \geq 1$ denote the number of vectors in the set $\mathcal{X}$. It is assumed that the vectors in $\mathcal{X}$ are realizations of independent identically distributed trials of the outcome class. Let π_i denote the mixing proportion of the i-th component population, and let the i-th multivariate Gaussian component PDF have mean vector μ_i and covariance matrix Σ_i. The parameter vector characterizing the mixture PDF is $\lambda \equiv \{ (\pi_i, \mu_i, \Sigma_i) , i = 1, \ldots, G \}$. The likelihood function $\mathcal{G}(\mathcal{X}|\lambda)$ is defined by the product

$$\mathcal{G}(\mathcal{X}|\lambda) = \prod_{t=1}^{T} g(X_t|\lambda), \tag{10}$$

where the mixture PDF $g(X|\lambda)$, for any $X \in \mathbb{R}^N$, is given by (*cf.*, (3) and (5))

$$g(X|\lambda) = \sum_{i=1}^{G} \pi_i \left[(2\pi)^N |\Sigma_i|\right]^{-1/2} \exp\left\{-\frac{1}{2}(X-\mu_i)^T \Sigma_i^{-1} (X-\mu_i)\right\}. \tag{11}$$

It is clear that computing a (global) maximum likelihood estimate for λ is a highly nonlinear problem. In addition, the likelihood function $\mathcal{G}(\mathcal{X}|\lambda)$ often has distinct local maxima. In practice, an appropriately selected local maximum is usually used instead of the global maximum.

The EM algorithm is an iterative algorithm that computes stationary points of the posterior likelihood function (10) without taking gradients, or derivatives. It begins with an initial guess, say $\tilde{\lambda}$, for the optimum parameter vector, and each iteration gives a new parameter vector, say $\hat{\lambda}$, that is *guaranteed to increase* the value of the posterior likelihood (10) unless $\tilde{\lambda}$ is a stationary value for $\mathcal{G}$, in which case $\tilde{\lambda} = \hat{\lambda}$. Consequently, if the EM iterates are bounded, the EM algorithm must converge to a stationary

point of $\mathcal{F}$. In practice, stationary points found by the EM algorithm are usually also points of local maxima. By restarting the the EM algorithm with different initial guesses $\tilde{\lambda}$, and choosing the best of the local maxima so obtained, a satisfactory maximum likelihood estimate for λ can be found. Convergence rates of the EM algorithm are discussed in [4].

4. CONCLUDING REMARKS

Training the NN architecture described in this paper can be performed using readily available computing resources. This is a significant feature, and it is a consequence of the general statistical methodology described herein. Explicit formulae for nodal thresholds and interconnection weights in terms of the estimated mean vectors and covariance matrices of the Gaussian mixture components are given. These formulae eliminate the need to train the NN explicitly, and refocus the training effort directly onto established statistical algorithms for estimating mixture PDFs.

Alternative NN implementations for Gaussian PDFs are possible. In addition, NNs can be designed for centrally symmetric non-Gaussian distributions. Details are given in [5] and [6]. A maximum likelihood method for unsupervised training is also discussed in [7].

The EM algorithm is a normalized counting algorithm, and it is readily incorporated into adaptive schemes for updating the NN weights and thresholds as new data is added to the training set or, alternatively, as old data is deleted. The advantages of such adaptive training methods for NN applications are potentially significant. Feedback from the training data into the defining parameters of the NN described in the paper is greatly facilitated by the computational simplicity of the EM training algorithm.

REFERENCES

1. B. W. Lindgren, Statistical Theory, Third Ed., Macmillan, New York, 1976.
2. A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood From Incomplete Data via the EM Algorithm (with discussion)," Journal of the Royal Statistical Society, Series B, 39(1977), 1-38.
3. D. F. Specht, "Probabilistic Neural Networks For Classification, Mapping, or Associative Memory," Proceedings of the IEEE International Conference on Neural Networks, July 1988, Vol. 1, pp. 525-532.
4. G. J. McLachlan and K. E. Basford, Mixture Models, Inference and Applications to Clustering, Marcel Dekker, New York, 1988.
5. R. L. Streit, "A Dual Neural Network Architecture For Gaussian Components of a Mixture Density Function," Naval Underwater Systems Center Invention Disclosure, 24 March 1989.
6. R. L. Streit, "Neural Network Architectures For Non-Gaussian Components of a Mixture Density Function," Naval Underwater Systems Center Invention Disclosure, 24 March 1989.
7. R. L. Streit, "A Neural Network For Maximum Likelihood Classification With Supervised And Unsupervised Training Capability," Naval Underwater Systems Center Invention Disclosure, 24 March 1989.

Maximum Likelihood Training Of Probabilistic Neural Networks

R. L. Streit and T. E. Luginbuhl

Abstract

A maximum likelihood method is presented for training probabilistic neural networks (PNNs), using a Gaussian kernal, or Parzen window. The proposed training algorithm enables general nonlinear discrimination and is a generalization of Fisher's method for linear discrimination. Important features of maximum likelihood training for PNNs are (1) it economizes the well known Parzen window estimator while preserving feed-forward NN architecture, (2) it utilizes class pooling to generalize classes represented by small training sets, (3) it gives smooth discriminant boundaries that often are "piece-wise flat" for statistical robustness, (4) it is very fast computationally compared to back-propagation, and (5) it is numerically stable. The effectiveness of the proposed maximum likelihood training alogrithm is assessed, using nonparametric statistical methods to define tolerance intervals on PNN classification performance.

1 Introduction

Classification is the following decision problem: given an input vector X, decide to which of several known classes the input X belongs. The classes are assumed to be mutually exclusive and exhaustive. Useful characterizations of the classes are assumed to be either unknown or unavailable and must be estimated from a given collection of labelled training samples (i.e., input vectors corresponding to each class). The absence of *a priori* class characterizations is the major difficulty in classification.

The training samples available for each class reflect the intrinsic variability of the class. Measurement errors are normally present in the training samples also, but such errors are subsumed here in the guise of class variability. Class variability models developed in this paper are based on the following fundamental assumptions:

1. Each class is a multivariate random variable with a continuous class conditional probability density function (PDF).

2. Every input vector X is a realization of one of the classes.

3. Each vector in the training sample set $\mathcal{T}$ is a realization of the random variable corresponding to its class label.

From the first two assumptions it follows that the well known Bayesian classifier [1, Chapter 13] is the optimum classifier in the sense of minimizing the overall misclassification risk. The implemention of a homoscedastic Gaussian mixture (defined in Section II) approximation to the optimum classifier in a probabilistic neural network (PNN) structure is discussed in Section II.

Obtaining meaningful class conditional PDF estimates for classes represented by only a few samples in the training set $\mathcal{T}$ is a difficult and thorny problem, but one that occurs often in practice. We treat the small sample size problem by a new sample pooling method that generalizes a classical statistical technique due to Fisher

[2, Chapter 4]. We refer to this new method as Generalized Fisher (GF) training, and it yields (local) maximum likelihood estimates of the class conditional PDFs. GF training is discussed in Section III and derived in the Appendix. A discussion of the training of *a priori* class probabilities and misclassification costs is given in Section IV. Examples of GF training on small, moderate, and large size training sets $\mathcal{T}$ are presented in Section V, and the effectiveness of GF training for these examples is assessed in Section VI using a nonparametric statistical method called tolerance intervals. Concluding remarks are given in Section VII.

2 Neural Network Implementation of Mixture Gaussian PNNs

The purpose of this section is to show that a four layer feed-forward PNN using a general Gaussian kernal, or Parzen window, can implement exactly the general homoscedastic Gaussian mixtures used in this paper to approximate the optimum classifier. Maximum likelihood training of the PNN is discussed below in Sections III and IV. The structure of the required PNN is represented in Figure 1. Each component of this PNN has a specific interpretation and, moreover, all the interconnection weights and nodal thresholds are given explicitly by mathematical expressions involving the defining parameters of the mixture Gaussian PDF estimates and the *a priori* class probabilities and misclassification costs.

Let N denote the dimension of the input vector X, and let M denote the number of different class labels in the training set $\mathcal{T}$. For $j = 1, \ldots, \mathrm{M}$, let $\mathrm{G}_j \geq 1$ denote the total number of different components in the j-th class mixture PDF. Let $p_{ij}(X)$ denote the multivariate PDF of the i-th component in the mixture for class j, and let π_{ij} denote the proportion of component i in class j. The "within-class" mixing

proportions π_{ij} are non-negative and satisfy the equations

$$\sum_{i=1}^{G_j} \pi_{ij} = 1,\ j = 1, \ldots, M. \tag{1}$$

The PDF of class j, denoted by $f_j(X)$, is approximated by a general mixture PDF, denoted by $g_j(X)$, that is,

$$f_j(X) \approx g_j(X) = \sum_{i=1}^{G_j} \pi_{ij} p_{ij}(X),\ j = 1, \ldots, M. \tag{2}$$

In this paper, only multivariate homoscedastic Gaussian mixtures are considered, hence $p_{ij}(X)$ has the form

$$p_{ij}(X) = (2\pi)^{-N/2} |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(X - \mu_{ij})^t \Sigma^{-1} (X - \mu_{ij})\right\}, \tag{3}$$

where μ_{ij} is the mean vector and Σ is the positive definite covariance matrix of $p_{ij}(X)$, and where superscript t denotes transpose. The covariance matrix Σ is chosen independent of the class index j and the component index i for reasons discussed in Section III. The results presented in this Section are readily extended to the more general (heteroscedastic) case of different covariance matrices. For details, see [3].

Let α_l denote the *a priori* probability of class l, that is, $\{\alpha_l\}$ are the "between-class" mixing proportions. Let c_{jl} denote the loss associated with classifying an input vector X into class j when the correct decision should have been class l. The risk $\rho_j(X)$ of classifying the input X into class j is the expected loss, so that

$$\rho_j(X) \approx \sum_{l=1}^{M} c_{jl} \alpha_l g_l(X). \tag{4}$$

The decision risk $\rho_j(X)$ is, thus, approximated by a mixture of Gaussian PDFs, as is seen by substituting (2) into (4). The minimum risk decision rule is to classify X into that class j having the minimum risk, that is, $j = \arg\min\{\rho_j(X)\}$. The decision j is the optimum Bayesian classification decision [1, Chapter 13] if $g_j(X) \equiv f_j(X)$ for all j, that is, provided the approximation (2) is an equality. (Ties for minimum

risk occur with probability zero, so they can be decided arbitrarily in practice.) The simple task of selecting the minimizing index j can be performed in many ways, and it has been pointed out [4] that a NN structure can be used for this task if desired.

The nodes in each of the four layers play specific roles in the PNN. The output of a fourth layer node is the risk $\rho_j(X)$ of choosing class j, as given by the approximation (4). The fourth layer therefore requires as many nodes as classes, namely M. A fourth layer node is conceptually equivalent to a decision risk. The output of a third layer node is $g_j(X)$, the approximate class conditional PDF given by equation (2). A node is needed for each class, so there are M third layer nodes. A third layer node is conceptually equivalent to a statistical hypothesis. The output of a second layer node is the likelihood $p_{ij}(X)$ of a component in a class mixture. A second layer node is needed for each Gaussian component in each class, so there are

$$G \equiv G_1 + G_2 + \ldots + G_M$$

second layer nodes. A second layer node is conceptually equivalent to a multivariate Gaussian random variable. A first layer node is needed for each degree of freedom in the χ^2 distributed exponent (the expression in braces in equation (3)) of every multivariate Gaussian component. There are N degrees of freedom and G components, so there are NG nodes in the first layer.

The activation function appropriate for a node depends upon the layer in which it resides. All fourth and third layer nodes use the identity function with a zero threshold, or bias. The second layer nodes use the function $exp(-x/2)$ with a zero bias. The first layer nodes all use the activation function $|x|$, but the biases vary from node to node across the layer. Explicitly, the first layer biases are given by

$$\tau_{ijk} = \left[L^{-1}\mu_{ij}\right]_k, \; i = 1,\ldots,G, \; j = 1,\ldots,M, \; k = 1,\ldots,N,$$

where L is any square root matrix factor of the covariance matrix Σ, that is $\Sigma = LL^t$. The Cholesky factor is one such square root. The bias τ_{ijk} depends on the destination

second layer node via the mean vector μ_{ij}. Further discussion of the activation functions and biases of the first three layers of the PNN are given in [3].

The description of the trained PNN is completed by defining interconnection weights between the layers, and giving their specific roles in the PNN. We begin with the top two layers and work down the NN. The interconnection weight between node l in the fourth layer and node j in the third layer is the product $\alpha_l c_{jl}$. These weights characterize decision risk formation. The interconnection weight between a third layer node (class mixture) and a second layer node (Gaussian component) is zero if the component does not belong to the class mixture, and is the mixing proportion π_{ij} if it is component i of the class j mixture. These weights characterize mixture formation. The interconnection weight between a second and a first layer node is either 1 or 0, depending on whether or not a given degree of freedom (first layer node) belongs to a given Gaussian random variable (second layer node). These weights characterize χ^2 random variable formation. Finally, the interconnection weights between the first and input layers are given by the entries of the inverse of the square root matrix factor L of the covariance matrix Σ. There are a total of G components, and L^{-1} is $N \times N$, so this gives GN^2 interconnection weights. If L is chosen to be the Cholesky factor of Σ, then L^{-1} is lower triangular and nearly half the weights between the first and input layers are zero. Alternatively, the matrix L can be chosen so that it characterizes the discrete Karhunen-Loève transformation corresponding to Σ, that is, $L^{-1} = \Lambda^{-1/2}U^t$, where $\Sigma = U\Lambda U^t$ is the singular value decomposition of Σ. In this case, the sparsity of L^{-1} is not immediately evident. A more detailed description of interconnection weights is given in [3].

3 Generalized Fisher Training of Probabilistic Neural Networks

The PNN proposed by Specht [5] is a special case of the PNN described in Section II, as is seen by setting the costs $c_{ll} = 0$ for all l and $c_{jl} = 1$ for $j \neq l$, and noting that the fourth layer is essentially superfluous in this case. Specht's PNN implements the Parzen window PDF estimator [6] using the so-called product Gaussian (i.e., uncorrelated Gaussian) window. The Parzen window sets the interconnection weights and nodal activation functions. Specht's PNN is thus a three layer feed-forward NN that uses mixtures of uncorrelated Gaussians to estimate the class conditional PDFs.

Specht's PNN is an excellent tool for initial exploration of new large training sets. Nonetheless, its usefulness in practice is limited by two factors. Firstly, because it is based on the Parzen window estimator, the total number G of Gaussian components must equal the number of samples in the training set $\mathcal{T}$. Therefore, it requires large amounts of data storage when extensive training sets are available. Secondly, an intrinsic smoothing parameter must be estimated on the basis of classification performance. Since robust estimates of classification performance are difficult to establish for small sample size, estimates of the smoothing parameter may be unreliable in practice. Both factors can often be mitigated by heuristics suited to the particular application. The contribution of generalized Fisher (GF) training is that it successfully treats both these problems without the need of heuristics.

The GF trained PNN requires significantly fewer nodes and interconnection weights than Specht's PNN in most problems of practical interest. A careful comparison of the two architectures below the third layer shows that the GF trained PNN is more efficient in both nodes and weights if

$$G \leq \frac{T}{\beta N}, \tag{5}$$

where β is the sparseness index of the inverse square root matrix factor of the

covariance matrix Σ, that is, L^{-1} has βN^2 nonzero entries. L^{-1} is fully dense if $\beta = 1$ and least dense (diagonal) if $\beta = 1/N$. By choosing L to be the Cholesky factor, the index β can always be made at least as small as $(N+1)/(2N) \approx 1/2$. From inequality 5, it is clear that the trained PNN is most effective in reducing node and weight requirements in large training set problems. For small training sets, the reduction in the number of nodes and weights depends on the sparseness index β of L^{-1}.

We begin the discussion of GF training by reviewing a classical treatment of the two class discrimination problem: Fisher's linear discriminant (FLD). FLD is based on the premise that both classes are multivariate Gaussian random variables with a common covariance matrix Σ, but different mean vectors μ_1 and μ_2. The available training set $\mathcal{T}$ is assumed to be correctly labelled, and we write $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, where $\mathcal{T}_j$ denotes the subset of $\mathcal{T}$ with class label j. The sample means

$$\hat{\mu}_j = \frac{1}{\#(\mathcal{T}_j)} \sum_{X \in \mathcal{T}_j} X, \; j = 1, 2, \tag{6}$$

estimate the means μ_1 and μ_2, where $\#(\cdot)$ denotes the cardinality (number of samples) of a training set. The covariance matrix Σ is estimated by Fisher's within-class scatter matrix

$$\hat{\Sigma} = \frac{1}{\#(\mathcal{T})} \sum_{j=1}^{2} \sum_{X \in \mathcal{T}_j} (X - \hat{\mu}_j)(X - \hat{\mu}_j)^t. \tag{7}$$

The estimation error for Σ is reduced by pooling the sample data, i.e., by using all samples in the training set $\mathcal{T}$. Given these estimates, the log-likelihood ratio is evaluated for an unknown vector X to be classified. The classification decision is obtained by comparing this ratio to an appropriate threshold. Contours of constant log-likelihood ratio are in this case hyperplanes in the feature space (i.e., $\mathbf{R}^N$), and the hyperplane corresponding to the threshold value is the FLD.

The FLD is known to be robust in the sense that linearly separable classes are often successfully discriminated in practice when neither class is truly Gaussian. Even when both classes are Gaussian but have different covariance matrices, some

authors have observed that the FLD is often a better classifier than the optimum quadratic discriminator. For fixed training set size, the increased estimation error in the two covariance matrices that results from not pooling the training samples is, presumably, the cause of the relatively greater robustness of the FLD in this case.

Pooling the training samples provides a natural way of developing PDF estimates for classes that have few samples in the training set $\mathcal{T}$. In applications where samples from different classes have broadly similar correlational structure, it is reasonable to pool the training samples when the sample size is small. Moreover, in practice, in the absence of *a priori* information to the contrary, it is probably inevitable that pooling will be used to generalize small sample set classes. Pooling is the basic strategy adopted in this paper.

GF training is a generalization of the FLD methodology. It uses a homoscedastic "mixture of mixtures" assumption to formulate a posterior likelihood function $\mathcal{L}$ for the entire training set $\mathcal{T}$. An ordinary homoscedastic mixture PDF is a mixture in which the components share a common covariance matrix Σ. By the term homoscedastic mixture of mixtures, we mean that a common covariance matrix Σ is used within each class mixture and also across all classes represented in the training set $\mathcal{T}$. The likelihood function $\mathcal{L}$ is highly nonlinear in the defining parameters of the mixture of mixtures, and it is not generally possible to factor it into terms depending on only one class label. Maximum likelihood parameter estimates for each class PDF are therefore jointly dependent on training samples across all classes.

Maximum likelihood parameter estimates for the mixture of mixtures are obtained numerically by utilizing an algorithm based on the Expectation-Maximization (EM) method [7]. The derivation of the GF training algorithm is given in the Appendix. The remainder of this Section is devoted to formulating the likelihood function $\mathcal{L}$, to stating the most important properties of GF training, and to interpretating its maximum likelihood solution. Equations (74) - (78) summarize the GF algorithm iteration from step n to step n+1.

The parameters λ defining the homoscedastic mixture of mixtures comprises the following variables:

- α_j = the *a priori* probability of class j,
- π_{ij} = the mixing proportion of component i in class j,
- μ_{ij} = the mean vector of component i in class j, and
- Σ = the common covariance matrix of all Gaussians.

Thus, λ comprises a total of $M + G + NG + N^2$ real variables, though not all of them are independent (e.g., Σ is symmetric and mixing proportions sum to 1). For $j = 1, \ldots, M$, let

$$\lambda_j = \{\pi_{ij}, \mu_{ij}, \Sigma\}_{i=1}^{G_j}$$

denote the parameters defining the homoscedastic Gaussian mixture for class j. The labelled training set $\mathcal{T}$ is partitioned into the disjoint subsets

$$\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \ldots \cup \mathcal{T}_M,$$

where $\mathcal{T}_j$ comprises those samples in $\mathcal{T}$ with class label j. The posterior likelihood function $\mathcal{L}(\mathcal{T}|\lambda)$ is defined on $\mathcal{T}$ by assuming that the samples in $\mathcal{T}_j$ are independent for each j, and that the class labels are assigned correctly. From these assumptions, it follows from equation (23) in the Appendix that the GF log-likelihood function is

$$\log \mathcal{L}(\mathcal{T}|\lambda) = \sum_{j=1}^{M} \sum_{X \in \mathcal{T}_j} \log \left[\alpha_j g_j(X|\lambda_j)\right],$$

where the function $g_j(X|\lambda_j)$ is identical to the class PDF $g_j(X)$ defined by equation (2). Estimating the parameter set λ is the central task of GF training.

The GF training algorithm converges to a local maximum likelihood estimate λ_{ML} for λ. The EM method for mixtures is derived in [8] for one class, i.e., the special case $M = 1$. It is extended to the GF likelihood function in the Appendix.

GF training is an iterative procedure that computes stationary points of the posterior likelihood function $\mathcal{L}$ without taking gradients or derivatives. It begins with an initial guess, say $\tilde{\lambda}$, for the optimum parameters, and each iteration gives a new parameter estimate, say λ^+, that is guaranteed to increase the value of the posterior likelihood function $\mathcal{L}$ unless $\tilde{\lambda}$ is a stationary value of $\mathcal{L}$, in which case $\tilde{\lambda} \equiv \lambda^+$. Consequently, if the GF algorithm iterates are bounded above, as they typically are in applications (see the Theorem in the Appendix), the GF training algorithm must converge to a stationary point. In practice, stationary points of $\mathcal{L}$ are also points of local maxima. By restarting the GF training algorithm with different initial guesses $\tilde{\lambda}$, and choosing the best of the local maxima so obtained, a satisfactory maximum likelihood estimate for λ can be found.

An intuitive understanding of the shape of the decision (discriminant) surface can be gained in certain instances. Consider the two class problem. The FLD always has a linear decision boundary, as remarked above, but GF training will not result in linear decision boundaries in general. The nonlinear decision boundary will be very flat (linear) wherever the input vector X lies "close" to only one component in each of the two classes. The reason is that the log-likelihood ratio behaves locally like the FLD in this case. Intuitively, then, the GF decision surface comprises several nearly flat sections that are joined together by smoothly varying transitional surfaces. This intuitive image suggests that GF training may be robust against overtraining on the sample set $\mathcal{T}$ when G is in some sense small compared to the size of the training set $\mathcal{T}$. The image also suggests a "decision directed" method for obtaining piecewise linear discriminants, and this is discussed briefly in Section VII.

Finally, GF training has an important translation property that shows clearly that GF training is based on PDF estimation and not on class discrimination. To be explicit, for $j = 1, \ldots, M$, let Ψ_j denote the training set $\mathcal{T}_j$ after translation by a given vector ϕ_j, and let Ψ denote the union of all sets Ψ_j. Suppose GF training applied to the translated training set Ψ converges to the parameter set λ_Ψ, and

that GF training applied to the set $\mathcal{T}$ converges to the parameter set $\lambda_{\mathcal{T}}$. Then the parameter sets λ_{Ψ} and $\lambda_{\mathcal{T}}$ are translates, that is, they are identical, except that the mean vector in λ_{Ψ} of the i-th component in the j-th mixture is $\mu_{ij} + \phi_j$, where μ_{ij} is the corresponding mean vector in $\lambda_{\mathcal{T}}$. This result assumes that the initial parameter sets are also translates. It follows from the translation property that the estimated class conditional PDFs are independent of the between-class sample separations. Classification performance of GF training is therefore determined by two independent factors: (1) the separation of the class means, and (2) the detailed shape of the individual class conditional PDFs.

4 Training *A Priori* Class Probabilities and Misclassification Costs

The GF training algorithm gives explicit maximum likelihood estimates for the class *a priori* probabilities $\{\alpha_j\}$ without iteration. From equation (74), the maximum likelihood estimate of α_j is $\hat{\alpha}_j = \#(\mathcal{T}_j)/\#(\mathcal{T})$, or, in words, $\hat{\alpha}_j$ represents the relative abundance of class j in the training set $\mathcal{T}$. Clearly, the training set $\mathcal{T}$ contains significant *a priori* class probability information only if it has been carefully compiled.

Standard statistical practice requires screening the training set to eliminate outliers and other anomalies. Moreover, it may also be necessary to screen the training set to ensure correctly labelled samples. If careful attention is not given to these important tasks, the resulting training set $\mathcal{T}$ will contain little or no meaningful information regarding the class *a priori* probabilities. In this case, the likelihood function $\mathcal{L}$ must be modified slightly. It becomes

$$\log \mathcal{L}(\mathcal{T}|\lambda \setminus \{\alpha_j\}) = \sum_{j=1}^{M} \sum_{X \in \mathcal{T}_j} \log g_j(X|\lambda_j)$$

where $\lambda \setminus \{\alpha_j\}$ denotes the parameter set λ with $\{\alpha_j\}$ removed. It is straight forward to show, using the methods of the Appendix, that the GF training algorithm

modified for this likelihood function is identical to equations (75) – (78). The sole difference is that $\{\alpha_j\}$ are no longer estimated. We will refer to both algorithms as GF training algorithms.

The preceding comments should not obscure the fact that it is still necessary to train class *a priori* probabilities in applications in which these quantities cannot be estimated from the training set $\mathcal{T}$. Although one may resort to information theoretic concepts such as entropy [9], a more appropriate recourse for many applications is to exploit *a priori* information not immediately available from within the training set $\mathcal{T}$, as it is defined in Section I. The same is true as well concerning the misclassification costs $\{c_{jl}\}$. Training these fundamental quantities is important if near-optimum classification performance is to be attained in practice.

The essential difficulty is that the likelihood function $\mathcal{L}(\mathcal{T}|\lambda \setminus \{\alpha_j\})$ is independent of the misclassification costs $\{c_{jl}\}$ and the probabilities $\{\alpha_j\}$. No maximum likelihood training algorithm can estimate factors missing from the fundamental likelihood structure. Thus, $\mathcal{L}$ must be modified to include dependence on *a priori* information. This task requires intimate knowledge of the particular application together, perhaps, with additional observation time history. Although time history can be included in $\mathcal{T}$, it is clear that training $\{c_{jl}\}$ and $\{\alpha_j\}$ may require an extensive modification of $\mathcal{L}(\mathcal{T}|\lambda \setminus \{\alpha_j\})$ and involve information and methods outside the scope of the present paper.

5 GF Training Examples

Three examples are presented to illustrate the effectiveness of GF training on different size training sets. To focus clearly on the small training set problem, the same classes are used in all the examples, and training is performed on different size subsets of the available training set $\mathcal{T}$. The effect of using different subsets of $\mathcal{T}$ on classification performance is assessed in Section VI.

Each example comprises three classes defined on $\mathbf{R}^2$, that is, the dimension of the input vector X is $N = 2$. The samples in $\mathcal{T}$ are measured data, not simulated. Because of the way $\mathcal{T}$ was gathered and screened in the application, the relative abundance of sample data for each class does not reflect the *a priori* class probabilities $\{\alpha_j\}$. The class prior probabilities α_j are chosen equal to 1/3. The misclassification costs c_{jl} are defined by equation (4) and are chosen equal to 0 if $l = j$ and equal to 1/2 if $l \neq j$. The optimum classifier is, with these choices, equivalent to the well known maximum likelihood classifier.

Given model orders $\{G_j\}$, GF training is defined for these examples by equations (75)–(78). The best choice of G_j, the number of (bivariate) Gaussian components in the mixture PDF for class j, is a model order selection problem, and its solution is application dependent. Typically, G_j should be chosen as small as possible without losing classification performance. The study of the order selection problem is greatly facilitated by the numerical robustness of GF training; however, this important problem is outside the scope of the present paper. Overtraining is the only aspect of this problem upon which we will comment.

GF training requires initial values for the class mixture parameters. For class j, the initial mixing proportions π_{ij} were chosen equal to $1/G_j$. The initial covariance matrix was the within-class scatter matrix (cf. equation (7) for two classes) for the training set. The initial mean vectors for Example 1 were chosen randomly within a square containing the appropriate class samples. The initial mean vectors for Example 2 were a subset of those for Example 1, and Example 2 contained those for Example 3. No effort was made to restart GF training to determine if the local maximum likelihood solutions obtained were globally optimum.

To facilitate the discussion of the examples, we define the decision risk by

$$\rho(X) = \min \{\rho_1(X), \rho_2(X), \rho_3(X)\}, \tag{8}$$

where the decision risks $\{\rho_j(X)\}$ are approximated by the right hand side of equation

(4). The decision assurance is defined by

$$\delta(X) = \max\{\alpha_1 g_1(X), \alpha_2 g_2(X), \alpha_3 g_3(X)\}, \tag{9}$$

where the estimated class PDFs $\{g_j(X)\}$ are defined by equation (2). The optimum class decision is identified by the index j^*, where

$$j^* = \arg\min\{\rho_1(X), \rho_2(X), \rho_3(X)\}. \tag{10}$$

Because of the particular choice of costs and class priors, we also have the equivalent expression $j^* = \arg\max\{g_1(X), g_2(X), g_3(X)\}$.

GF training is very efficient in the examples presented in detail below. The convergence criteria required a relative increase of 10^{-4} in the log-likelihood function, that is, iteration ceased when the current value of $\log \mathcal{L}$ increased by a factor less than or equal to $1+10^{-4}$ times the previous value of $\log \mathcal{L}$. GF training for Example 1 converged in 53 iterations and used approximately two minutes of wall-clock time (including all file handling and I/O operations). Example 2 converged in 26 iterations in about 5 seconds, while Example 3 converged in 18 iterations in well under one second. The GF algorithm is implemented in single precision FORTRAN on a Sun SPARC-station I.

5.1 Example 1: Large size training set, $\mathcal{T}$

The training set $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$ comprises 1960, 720, and 500 two-dimensional training samples in classes 1, 2, and 3, respectively. Example 1 trains using all the available samples in $\mathcal{T}$ with the choices $G_1 = 8$, $G_2 = 4$, and $G_3 = 2$ for the number of class mixture components. This choice for $\{G_j\}$ reflects the relative diffuseness (as compared to an uncorrelated Gaussian distribution) of the training samples in the various classes. The trained mixing proportions and mean vectors of the class

mixture PDFs are listed in Table 1. The inverse of the trained covariance matrix is

$$\Sigma^{-1} = 10^{-3} \times \begin{bmatrix} 17.590 & -0.76978 \\ -0.76978 & 6.0786 \end{bmatrix}.$$

The eigenvalues of Σ are 56.6850 and 165.911, so Σ is numerically stable.

	Class 1		Class 2		Class 3	
component number	π	μ	π	μ	π	μ
1	0.196423	85.165 111.075	0.454747	58.643 11.885	0.517638	50.180 49.849
2	0.174009	115.670 43.677	0.227895	75.009 18.953	0.482362	50.180 49.849
3	0.158688	115.499 79.179	0.182448	38.771 10.130		
4	0.144088	98.808 88.775	0.134909	16.085 6.652		
5	0.111436	114.690 56.922				
6	0.091604	99.392 144.360				
7	0.073182	83.966 57.609				
8	0.050569	117.437 123.854				

Table 1: Mixing Proportions and Mean Vectors for Example 1

Figure 2 is a scatter plot of the training set $\mathcal{T}$, together with a graph of the three-class discriminant function obtained after GF training is completed. The graph of

the discriminant function is the boundary line between the regions of the (input) plane that map into the three different classes under the optimum decision rule (10). Inspection of Figure 2 indicates that good generalization of the training set $\mathcal{T}$ has taken place. Overtraining has not occurred, since overtraining is characterized by highly convoluted discriminant curves and the nonlinear discriminant curves in Figure 2 are smooth.

Figures 3, 4, and 5 depict the level curves, or contours, of the estimated class PDFs. The likelihood levels corresponding to the contours in these and subsequent figures are given in decibels referenced to the maximum PDF level, or dB//max. Generally, to plot a function in dB//max, it is normalized by its maximum value, and then 10 times its base 10 logarithm is taken. Thus, -20 dB//max is equivalent to a level that is 10^{-2} times the referenced maximum PDF level. The use of decibels is justified by the dynamic range of the likelihood functions involved. The maximum values of the class PDFs are 4.29×10^{-4}, 7.91×10^{-4}, and 16.41×10^{-4} for classes 1, 2, and 3, respectively. Class 3 has the largest maximum because it has the most compact PDF.

The large $\times$ marked on Figures 3–5 is the approximate location of the point of maximum likelihood. The locations of the mean vectors of the trained Gaussian components are marked with squares. Note that in class 1 the maximum likelihood does not occur at the mean vector of one of the Gaussian components.

Scatter plots of the training data have been superimposed on the contours in Figures 3–5. Sample data are marked with simple dots. In each class virtually all the training data lies within the -20 dB//max contour. Since the -20 dB//max contours of the classes intersect and their maxima do not greatly differ, perfect separation of the three classes is not achieved.

Although two Gaussians were permitted for class 3, GF training merged them by superimposing their means, as is seen from Table 1. Merging indicates that class 3 is overmodeled, that is, $G_3 = 2$ is too large. The common covariance matrix structure

is easily seen in the PDF level curves for class 3, depicted in Figure 5. Fitting a single Gaussian to only class 3 samples would give a slightly different covariance matrix. The difference is due to pooling samples across classes, i.e., the covariance matrix Σ depends jointly on the entire training set $\mathcal{T}$, not just the samples for any one class.

The decision risk $\rho(X)$ gives much more insight into the class structure than the simple discriminant curve alone. Figure 6 depicts the risk $\rho(X)$ in dB//max, where the maximum risk is 4.94×10^{-5}. The region of greatest decision risk occurs between classes 2 and 3, and this fact agrees very well with the good visual separation between the scatter plots of class 1 and the other two classes. The risk $\rho(X)$ thus not only confirms, but also quantifies, our intuition in the matter. Note that the discriminant curve runs along the ridges of the graph of $\rho(X)$.

The decision assurance $\delta(X)$ is depicted in Figure 7. The maximum assurance is 5.47×10^{-4}. The assurance function $\delta(X)$ is useful as an indicator of the correctness of the optimum decision. An outlier is easily classified by noting its relation to the discriminant curve, and the risk $\rho(X)$ associated with this decision is very small, as can be seen from Figure 6. Nonetheless, one should not feel too comfortable with any decision concerning an outlier. Screening outliers is accepted statistical practice, and screening can be facilitated by setting a threshold on the assurance $\delta(X)$. If $\delta(X)$ is not sufficiently large, then no classification decision is made. This is equivalent to postulating an additional "null" class having an appropriate diffuse PDF. Note that the discriminant curve runs down the valleys of the the graph of $\delta(X)$.

Classification performance estimates derived from the training set are optimistically biased, as is well known. However, overtraining has not occurred, so such estimates should not in this instance be significantly biased. Classification performance estimates are given in Table 2, and are presented primarily for comparison with the next two examples. Note that the largest off-diagonal entry of the con-

fusion matrix corresponds to misclassifying class 2 samples as class 3. This also corresponds to the region of greatest risk $\rho(X)$.

	Input Class			Sample Size	
Decision	Class 1	Class 2	Class 3	Input	Training
Class 1	98.32% (1927)	1.67% (12)	0.20% (1)	1960	1960
Class 2	0.97% (19)	93.61% (674)	0.80% (4)	720	720
Class 3	0.71% (14)	4.72% (34)	99.00% (495)	500	500

Table 2: Confusion Matrix for Example 1

5.2 Example 2: Moderate size training set, $\mathcal{T}/10$

Example 2 trains on the set $\mathcal{T}/10$, a fixed (randomly selected) subset of $\mathcal{T}$ with 196, 72, and 50 training samples representing classes 1, 2, and 3, respectively. The choices $G_1 = 4$, $G_2 = 2$, and $G_3 = 1$ are made in this example to reflect the reduced training set size, given the choices of $\{G_j\}$ in Example 1. The trained mixing proportions and mean vectors of the class mixture PDFs are listed in Table 3. The inverse of the trained covariance matrix is

$$\Sigma^{-1} = 10^{-3} \times \begin{bmatrix} 12.345 & -3.2255 \\ -3.2255 & 5.8022 \end{bmatrix}.$$

The eigenvalues of Σ are 73.164 and 223.24, so Σ is numerically stable.

The discriminant boundary and decision risk $\rho(X)$ are depicted in Figure 8. The maximum risk is 4.23×10^{-5}. The discriminant boundary is effectively piecewise linear in this example because the reduced model orders $\{G_j\}$ make the GF discriminant more prone to have locally flat behavior, as described above in Section III.

The risk function $\rho(X)$ has two discernable peaks (local maxima) that correspond to binary decision problems (i.e., two-class problems). There was only one peak in Example 1.

The decision assurance in dB//max is depicted in Figure 9. The maximum assurance is 4.15×10^{-4}. The covariance matrix structure, discernable in the elliptically shaped likelihood contours of Figure 9, is slightly rotated from that of Example 1, but this difference does not significantly alter the overall class likehood distributions. It is interesting to note that the point of intersection of the three arms of the GF discriminant lies in a small valley (i.e., local minimum).

Classification performance was estimated on the full training set $\mathcal{T}$. The confusion matrix is given in Table 4. As is evident from Table 4, GF training on the reduced size set $\mathcal{T}/10$ gives excellent classification performance. Note that class 3 is never misclassified as class 1. Correct classification rates are slightly less than those for Example 1, possibly because the larger testing set has reduced the small positive bias evident in the confusion matrix of Example 1. Note that the largest off-diagonal entry in the confusion matrix corresponds to the largest peak in the risk $\rho(X)$, and the second largest off-diagonal entry corresponds to the second largest peak.

5.3 Example 3: Small size training set, $\mathcal{T}/100$

Example 3 trains on the set $\mathcal{T}/100$, a fixed (randomly selected) subset of $\mathcal{T}/10$ with 20, 7, and 5 training samples representing classes 1, 2, and 3, respectively. The number of components per class are further reduced to $G_1 = 2$, $G_2 = 1$, and $G_3 = 1$. The trained mixing proportions and mean vectors of the class mixture PDFs are listed in Table 5. The inverse of the trained covariance matrix is

$$\Sigma^{-1} = 10^{-3} \times \begin{bmatrix} 11.230 & 4.9816 \\ 4.9816 & 8.8180 \end{bmatrix}.$$

	Class 1		Class 2		Class 3	
component number	π	μ	π	μ	π	μ
1	0.366496	113.240 50.896	0.814163	61.862 15.743	1.00000	50.801 51.445
2	0.248630	89.122 93.863	0.185837	17.166 7.785		
3	0.235406	114.376 85.979				
4	0.149468	95.555 142.098				

Table 3: Mixing Proportions and Mean Vectors for Example 2

	Input Class			Sample Size	
Decision	Class 1	Class 2	Class 3	Input	Training
Class 1	97.19% (1905)	1.53% (11)	0.00% (0)	1960	196
Class 2	1.79% (35)	92.64% (667)	1.80% (9)	720	72
Class 3	1.02% (20)	5.83% (42)	98.20% (491)	500	50

Table 4: Confusion Matrix for Example 2

The eigenvalues of Σ are 66.010 and 204.149, so Σ is numerically stable.

The discriminant boundary and decision risk are depicted in Figure 10. The three arms of the discriminant boundary are nearly linear in this example because of the small model orders $\{G_j\}$. The class decision regions are unbounded in this example. The class 3 region is bounded in the other two examples. The risk function $\rho(X)$ has only one peak, and it lies in the same location as that of Example 1. The maximum decision risk is 8.72×10^{-5}.

The decision assurance in dB//max is depicted in Figure 11. The maximum assurance is 4.57×10^{-4}. The covariance matrix structure, clearly visible in Figure 11, shows that the covariance matrix Σ is significantly rotated from that of the other two examples. The reason is the small model order for class 1. It happens during training that one Gaussian models the most densely clustered samples, and the other models the most significant portion of the remaining samples in class 1. Since class 1 samples dominate the within-class covariance matrix calculation and the densely clustered samples dominate (cf. the mixing proportions of Table 5) in class 1, the covariance matrix Σ reflects the dense portion of class 1 samples. This effect is evident in Figure 11.

Classification performance was estimated on the sample set $\mathcal{T}$ that was used as the training set for Example 1. The confusion matrix is given in Table 6. Note

	Class 1		Class 2		Class 3	
component number	π	μ	π	μ	π	μ
1	0.757665	106.663 71.176	1.00000	61.286 15.857	1.00000	48.823 49.063
2	0.242335	106.816 109.032				

Table 5: Mixing Proportions and Mean Vectors for Example 3

that class 3 is never misclassified as class 1. Correct classification rates are very similar to those of Example 2. The largest off-diagonal entry in the confusion matrix corresponds to the peak of $\rho(X)$, just as in the other examples. Clearly, GF training on the greatly reduced size training set $\mathcal{T}/100$ seems to be nearly as good as that attained by using the entire training set $\mathcal{T}$.

	Input Class			Sample Size	
Decision	Class 1	Class 2	Class 3	Input	Training
Class 1	97.65% (1914)	1.11% (8)	0.00% (0)	1960	20
Class 2	0.46% (9)	91.81% (661)	0.80% (4)	720	7
Class 3	1.89% (37)	7.08% (51)	99.20% (496)	500	5

Table 6: Confusion Matrix for Example 3

6 Tolerance Intervals for Assessing Classification Performance

Example 3 is one realization of a larger "experiment" conducted by randomly selecting subsets of specified size from the training set $\mathcal{T}$. In this section we assess quantitatively how representative this example was of the larger experiment by using a nonparametric statistical method known as tolerance intervals. Tolerance intervals are similar to confidence intervals in their use, but they are defined and derived very differently.

We define a training trial Z on a fixed size subset of a given labelled training set $\mathcal{T}$ in the following manner. Firstly, a (uniform) random sample $\mathcal{S}$ of the specified

size is drawn from $\mathcal{T}$ without replacement. The subset $\mathcal{S}$ is returned to $\mathcal{T}$ before the next training trial begins. Next, GF training is conducted on the set $\mathcal{S}$. The initial parameter set λ required by GF training may be fixed or generated randomly, but the same initialization procedure must be used in all training trials. On convergence of the GF training algorithm, classification performance is assessed on the set $\mathcal{T} \setminus \mathcal{S}$. In the examples presented in Section V, classification performance is measured by a confusion matrix. In this Section we also consider the total error rate.

Training trials are independent trials of a multivariate discrete random variable. The trials are independent because of the independence of the subsets $\mathcal{S}$ drawn from $\mathcal{T}$, and the trials are discrete outcome because there are only a finite number of different possible subsets $\mathcal{S}$ that can be drawn from $\mathcal{T}$. Finally, the trials are multivariate because the outcome is the calculated confusion matrix together with the total error rate. In principle, the PDF of the training trials can be found exactly by systematically running through the entire list of all possible subsets $\mathcal{S}$ of the training set $\mathcal{T}$; however, except for very small training sets $\mathcal{T}$, such a procedure is computationally prohibitive.

Suppose momentarily that the total error rate, denoted by ϵ, is the only outcome of a training trial and that its PDF is continuous, not discrete. Denote the PDF of ϵ by $E(\epsilon)$, and let n independent training trials with outcomes $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ be given. Then the population fraction or coverage, u, of the PDF $E(\epsilon)$ between $\min \epsilon_k$ and $\max \epsilon_k$ is given exactly by

$$u = \int_{\min \epsilon_k}^{\max \epsilon_k} E(z)dz.$$

Wilks shows [10] that the PDF of u, denoted $P_n(u)$, *is independent of the PDF* $E(\epsilon)$ and is equal to

$$P_n(u) = n(n-1)u^{n-2}(1-u).$$

Robbins shows [11] that order statistics are the only statistics which yield distribution free tolerance intervals. If we want coverage $u \geq 100\beta\%$ with probability

$100\alpha\%$, then n must be chosen so that

$$Pr\{u \geq \beta\} = \alpha = \int_{\beta}^{1} P_n(u)du. \tag{11}$$

The PDF of the total error rate for a training trial is discrete. The result by Wilks is applicable to univariate discrete PDFs, but (11) changes to $Pr\{u \geq \beta\} \geq \alpha$ in this case. For a proof of Wilks' result for discrete PDFs, see [12].

In general, a training trial outcome includes the confusion matrix and total error rate. The PDF of the confusion matrix is discrete and multivariate. Wald [13] derives distribution free tolerance intervals for continous multivariate outcomes by computing order statistics on each vector component separately and by carefully choosing a multidimensional interval (block). Tukey extends Wald's results to more general choices of multidimensional blocks [14] and to discrete multivariate PDFs [15]. The curves given in [16] are valid for continuous multivariate PDFs. These curves may be used to obtain lower bounds on the confidence α for discrete multivariate PDFs [15], just as in the univariate case.

From the curves in [16], taking n equal to 50 ensures coverage $90\% = 100\beta\%$ and confidence $95\% = 100\alpha\%$. Results for $n = 50$ independent trials of the experiments $\mathcal{T}/10$ and $\mathcal{T}/100$ are shown in Table 7 and in Table 8, respectively. The corresponding tolerance intervals for the total error rate in percentages are 3.6 ± 1.1 and 8.4 ± 5.3 for $\mathcal{T}/10$ and $\mathcal{T}/100$, respectively. The tolerance intervals for the confusion matrices were obtained using Wald's method [13] for selecting the multidimensional block. Table 7 shows that GF training yields good class characterization when a $\mathcal{T}/10$ training set is used. The largest tolerance intervals are for class 2 because class 2 data overlaps both class 1 and class 3 data. Table 8 shows that GF training does not perform as well on $\mathcal{T}/100$ training sets. In particular, large performance variations are possible when trying to distinguish class 2 data from class 3 data. Table 8 clearly shows that the excellent results obtained in Example 3 are at the high end of the tolerance interval (block) for the experiment $\mathcal{T}/100$.

7 Concluding Remarks

The examples of Sections V and VI give convincing evidence of the utility of GF training. Although the classes in these examples are nearly linearly separable (cf. Figure 10 and Table 6), the important translation property discussed in Section IV implies that linear separability is not the central issue for GF training because we can always train on widely separated translates of the individual class training sets $\mathcal{T}$. The examples show that GF training has obtained very reasonable estimates of the class conditional PDFs, that is, the available class training samples have been generalized in some sense. Highly nonlinear discriminants are by-products of good class conditional PDF estimates.

A "decision directed" generalized Fisher (DDGF) method can be used, if desired, to obtain a strictly piecewise linear approximation to the GF discriminant surface. The DDGF method classifies an input vector X into the class j^* such that $(i^*, j^*) = \arg\max\{\alpha_j \pi_{ij} p_{ij}(X)\}$. The component decision i^* is also part of the DDGF decision. If the required maximum is taken first over the component index i, and then over the class index j, it is seen that the DDGF method is equivalent to a two stage decision and is implementable in a feed-forward NN structure that avoids using exponential nonlinearities. In the first stage, a "within-class" decision determines which component generated the given input vector. There are as many within-class decisions as there are classes. In the second stage, a final "between-class" decision is made using the representative class components determined by the first stage.

Decision (%)	Class 1	Class 2	Class 3
Class 1	97.1 ± 1.2	1.2 ± 0.6	0.0 ± 0.0
Class 2	1.5 ± 0.7	92.1 ± 2.5	1.2 ± 1.2
Class 3	1.4 ± 0.9	6.6 ± 2.2	98.8 ± 1.2

Table 7: Tolerance Intervals for $\mathcal{T}/10$ Experiment

Both the within-class and between-class decisions have piecewise linear discriminants because of the Gaussian PDF structures in each instance. The within-class mixing proportions $\{\pi_{ij}\}$ are the *a priori* probabilities for the within-class decisions, while the between-class mixing proportions $\{\alpha_j\}$ are used for the between-class decision.

An interesting aspect of GF training is that, because only one covariance matrix Σ is used across all classes, the principal components analysis (PCA) based on Σ is common to all M classes. A common PCA, taken together with the spread-of-the-means of the components, are potentially useful tools for investigating dimensional reduction of the feature space in all classes simultaneously. This unique aspect of GF training merits further study.

The GF training algorithm derived in the Appendix easily accommodates several useful extensions. Three extensions are mentioned here. Supervised / unsupervised GF training can be undertaken on training sets in which some of the training samples are unlabelled. Unsupervised GF training is the special case of all unlabelled training data. This extension is especially useful for applications in which the cost or difficulty of correctly labelling all the training samples is prohibitive. GF training iterations can be made adaptive and run "closed loop" if the class PDFs are time-varying. This extension requires reformulating the GF likelihood function with Bayesian prior distributions [8] for the mixing proportions, mean vectors, and covariance matrix of the class conditional mixture PDFs. Adaptive GF training is potentially useful in applications in which class statistics are either non-stationary or are treated as non-stationary to ensure robustness and a degree of fault tolerance.

Decision (%)	Class 1	Class 2	Class 3
Class 1	96.0 ± 3.0	5.8 ± 5.0	0.0 ± 0.0
Class 2	1.4 ± 1.4	$82.8 \pm 10.$	5.2 ± 5.2
Class 3	2.5 ± 1.6	$21.3 \pm 16.$	97.5 ± 2.3

Table 8: Tolerance Intervals for $\mathcal{T}/100$ Experiment

GF training can be extended to mixtures of discrete PDFs and continuous non-Gaussian PDFs [7]. These extensions may enable reduced PNN size (because of the increased modelling efficiency) in applications requiring discrete feature vectors or continuous non-Gaussian feature vectors. These extensions are not mutually exclusive. For example, adaptive GF training is possible with supervised / unsupervised training sets $\mathcal{T}$.

A Appendix: Derivation of the GF Training Algorithm

The Generalized Fisher (GF) training algorithm is based on the Expectation - Maximization (EM) method described in reference [7]. The EM method consists of two steps: The first step is called the expectation step or E-step, and the second is called the maximization step or M-step. The E-step extends the likelihood function $\mathcal{L}$ to the unobserved or "missing" data, and then computes an expectation of over the missing data to obtain an auxiliary function Q. The M-step maximizes the function Q with respect to the parameter set to be estimated. Reference [7] describes the conditions required for the EM method to converge to a local maximum of the likelihood function $\mathcal{L}$.

Suppose independent samples of a random vector X with dimension N are observed, where X has a mixture of mixtures conditional PDF given by

$$X \sim \sum_{j=1}^{M} \alpha_j g_j(X|\lambda_j) \tag{12}$$

where

$$\sum_{j=1}^{M} \alpha_j = 1, \tag{13}$$

$$g_j(X|\lambda_j) = \sum_{i=1}^{G_j} \pi_{ij} p_{ij}(X|\lambda_{ij}) \tag{14}$$

$$= \sum_{i=1}^{G_j} \frac{\pi_{ij}}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(X-\mu_{ij})^t \Sigma^{-1}(X-\mu_{ij})\right] \tag{15}$$

$$\sum_{i=1}^{G_j} \pi_{ij} = 1, \tag{16}$$

and the symbol X^t denotes the transpose of X. The number of components G_j can be different for different class mixtures. Note that the between-class mixing proportions $\{\alpha_j\}$ and the within-class mixing proportions $\{\pi_{ij}\}$ are contained in the interval $[0, 1]$.

The parameter sets

$$\lambda_{ij} = \{\mu_{ij}, \Sigma\} \tag{17}$$

$$\lambda_j = \{\pi_{ij}, \lambda_{ij}\}_{i=1}^{G_j} \tag{18}$$

$$\lambda = \{\alpha_j, \lambda_j\}_{j=1}^{M} \tag{19}$$

are unknown. In this appendix the GF training algorithm is derived for estimating the unknown parameter set λ from the training set $\mathcal{T}$, and it is based on the EM method. The following discussion will be devoted to developing the expectation step (E-step) and the maximization step (M-step) of the EM method applied to the mixture of mixtures PDF model in equation (12). The training set $\mathcal{T}$ is partitioned (labelled) so that for each component $g_j(X|\lambda_j)$ of the mixture $f(X|\lambda)$, T_j of the observations of X are from $g_j(X|\lambda_j)$:

$$\mathcal{T} = \{X_n\}_{n=1}^{T} = \left\{\{X_{kj}\}_{k=1}^{T_j}\right\}_{j=1}^{M} \tag{20}$$

where

$$\sum_{j=1}^{M} T_j = T. \tag{21}$$

The posterior likelihood function for the unlabelled training set $\mathcal{T}$ is

$$\mathcal{L}_u(\mathcal{T}|\lambda) = \begin{pmatrix} T \\ T_1 \cdots T_M \end{pmatrix} \prod_{n=1}^{T} \left(\sum_{j=1}^{M} \alpha_j g_j(X_n|\lambda_j)\right) \tag{22}$$

using equation (12) and the independence of the training samples. The multinomial coefficient is required in equation (22) because the training set $\mathcal{T}$ is unordered. Since the multinomial coefficient is a constant and only scales the likelihood function, it is dropped for the rest of this discussion. The likelihood function of the labelled training set $\mathcal{T}$ becomes

$$\begin{aligned} \mathcal{L}(\mathcal{T}|\lambda) &= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \left(\sum_{l=1}^{M} \alpha_l g_l(X_{kj}|\lambda_l)\delta(l-j)\right) \\ &= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j g_j(X_{kj}|\lambda_j) \end{aligned} \tag{23}$$

where $\delta(\cdot)$ is the Kronecker delta function. Substituting equation (14) in equation (23) yields

$$\mathcal{L}(\mathcal{T}|\lambda) = \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j \left(\sum_{i=1}^{G_j} \pi_{ij} p_{ij}(X_{kj}|\lambda_{ij}) \right). \tag{24}$$

If $\alpha_j = 0$ for some j, then $\mathcal{L}(\mathcal{T}|\lambda) = 0$. Therefore, we require that $\alpha_j > 0$, $j = 1, \ldots, M$.

The missing data in this problem is the index, i, of the PDF $p_{ij}(\cdot|\lambda_{ij})$ within the mixture $g_j(\cdot)$ from which X_{kj} originated. The "complete data" in this case is the set

$$\mathcal{T}' = \left\{ \{Y_{kj}\}_{k=1}^{T_j} \right\}_{j=1}^{M} = \left\{ \{X_{kj}, i_{kj}\}_{k=1}^{T_j} \right\}_{j=1}^{M} = \mathcal{T} \cup \mathcal{I}, \tag{25}$$

where i_{kj} denotes the component index of the PDF from which X_{kj} was drawn. Note that i_{kj} is *not* observed and that $1 \leq i_{kj} \leq \mathrm{G}_j$. The conditional PDF for $\mathcal{T}'$ is

$$\begin{aligned}
\mathcal{F}(\mathcal{T}'|\lambda) &= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j g_j(Y_{kj}|\lambda_j) \\
&= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j \left(\sum_{l=1}^{G_j} \pi_{lj} p_{lj}(Y_{kj}|\lambda_{lj}) \right) \\
&= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j \left(\sum_{l=1}^{G_j} \pi_{lj} p_{lj}(X_{kj}|\lambda_{lj}) \delta(l - i_{kj}) \right) \\
&= \prod_{j=1}^{M} \prod_{k=1}^{T_j} \alpha_j \pi_{lj} p_{lj}(X_{kj}|\lambda_{lj}) \Big|_{l=i_{kj}}
\end{aligned} \tag{26}$$

where $\delta(\cdot)$ is the Kronecker delta function. The PDF of $\mathcal{I} = \{i_{kj}\}$ conditioned on $\mathcal{T}$ and λ is then

$$\mathcal{K}(\mathcal{I}|\mathcal{T}, \lambda) = \frac{\mathcal{F}(\mathcal{T}'|\lambda)}{\mathcal{L}(\mathcal{T}|\lambda)}. \tag{27}$$

Substituting (23) and (26) into $\mathcal{K}$ yields

$$\mathcal{K}(\mathcal{I}|\mathcal{T}, \lambda) = \prod_{j=1}^{M} \prod_{k=1}^{T_j} \frac{\pi_{lj} p_{lj}(X_{kj}|\lambda_{lj})}{g_j(X_{kj}|\lambda_j)} \Big|_{l=i_{kj}} \tag{28}$$

$$= \prod_{j=1}^{M} \prod_{k=1}^{T_j} w_{lj}(X_{kj}) \Big|_{l=i_{kj}} \tag{29}$$

where

$$w_{lj}(X_{kj}) = \left. \frac{\pi_{lj} \exp\left[-\frac{1}{2}(X-\mu_{lj})^t \Sigma^{-1}(X-\mu_{lj})\right]}{\sum_{i=1}^{G_j} \pi_{ij} \exp\left[-\frac{1}{2}(X-\mu_{ij})^t \Sigma^{-1}(X-\mu_{ij})\right]} \right|_{X=X_{kj}} \tag{30}$$

Note that $w_{lj}(X_{kj}) \geq 0$, and $w_{lj}(X_{kj}) = 0$ if and only if $\pi_{lj} = 0$. It is straight forward to verify that

$$\sum_{\mathcal{I}} \mathcal{K}(\mathcal{I}|\mathcal{T}, \lambda) = 1, \tag{31}$$

where the sum over $\mathcal{I}$ is shorthand for the T-fold sum

$$\sum_{\mathcal{I}} = \sum_{i_{11}=1}^{G_1} \cdots \sum_{i_{T_1 1}=1}^{G_1} \sum_{i_{12}=1}^{G_2} \cdots \sum_{i_{T_2 2}=1}^{G_2} \cdots \sum_{i_{1M}=1}^{G_M} \cdots \sum_{i_{T_M M}=1}^{G_M}. \tag{32}$$

Similarly,

$$\sum_{\mathcal{I} \setminus i_{kj}} \mathcal{K}(\mathcal{I}|\mathcal{T}, \lambda) = w_{lj}(X_{kj}) \Big|_{l=i_{kj}} \tag{33}$$

where the sum over $\mathcal{I} \setminus i_{kj}$ is the same as the sum over $\mathcal{I}$ except that the sum over the index i_{kj} is deleted. Note that $\mathcal{K}(\cdot)$ defines a probability on the discrete space of indices $\mathcal{I}$.

The E-step of the EM method is defined to be

$$Q(\lambda|\lambda') = E\{\log \mathcal{F}(\mathcal{T}'|\lambda)|\mathcal{T}, \lambda'\}, \tag{34}$$

where the expectation is taken over the set of all possible indices $\mathcal{I} = i_{kj}$ and is conditioned on $\mathcal{T}$ and λ', where λ' is a given parameter set of the form (19). By definition of expectation,

$$\begin{aligned} Q(\lambda|\lambda') &= \sum_{\mathcal{I}} \log\left[\mathcal{F}(\mathcal{T}'|\lambda)\right] \mathcal{K}(\mathcal{I}|\mathcal{T}, \lambda) && (35) \\ &= \sum_{\mathcal{I}} \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log[\alpha_j \pi_{lj} p_{lj}(X_{kj}|\lambda_{lj})] \Big|_{l=i_{kj}} \mathcal{K}(\mathcal{I}|\mathcal{T}, \lambda') && (36) \\ &= \sum_{\mathcal{I}} \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log[\alpha_j] \mathcal{K}(\mathcal{I}|\mathcal{T}, \lambda') \\ &\quad + \sum_{\mathcal{I}} \sum_{j=1}^{M} \sum_{k=1}^{T_j} \log[\pi_{lj} p_{lj}(X_{kj}|\lambda_{lj})] \Big|_{l=i_{kj}} \mathcal{K}(\mathcal{I}|\mathcal{T}, \lambda'). && (37) \end{aligned}$$

The first term simplifies easily using equation (31) because

$$\sum_{\mathcal{I}}\sum_{j=1}^{M} T_j[\log \alpha_j]\mathcal{K}(\mathcal{I}|\mathcal{T},\lambda') = \left(\sum_{j=1}^{M} T_j \log \alpha_j\right)\sum_{\mathcal{I}}\mathcal{K}(\mathcal{I}|\mathcal{T},\lambda') \tag{38}$$

$$= \sum_{j=1}^{M} T_j \log \alpha_j. \tag{39}$$

Using equation (33), the second term in equation (37) simplifies to

$$\sum_{\mathcal{I}}\sum_{j=1}^{M}\sum_{k=1}^{T_j} \log[\pi_{lj}p_{lj}(X_{kj}|\lambda_{lj})]\Big|_{l=i_{kj}} \mathcal{K}(\mathcal{I}|\mathcal{T},\lambda')$$

$$= \sum_{j=1}^{M}\sum_{k=1}^{T_j}\sum_{l=1}^{G_j} \log[\pi_{lj}p_{lj}(X_{kj}|\lambda_{lj})]\Big|_{l=i_{kj}} \sum_{\mathcal{I}\setminus i_{kj}} \mathcal{K}(\mathcal{I}|\mathcal{T},\lambda') \tag{40}$$

$$= \sum_{j=1}^{M}\sum_{k=1}^{T_j}\sum_{l=1}^{G_j} \log[\pi_{lj}p_{lj}(X_{kj}|\lambda_{lj})]w'_{lj}(X_{kj})\Big|_{l=i_{kj}}, \tag{41}$$

where

$$w'_{lj}(X_{kj}) = \frac{\pi'_{lj}\exp\left[-\frac{1}{2}(X-\mu'_{lj})^t(\Sigma')^{-1}(X-\mu'_{lj})\right]}{\sum_{i=1}^{G_j}\pi'_{ij}\exp\left[-\frac{1}{2}(X-\mu'_{ij})^t(\Sigma')^{-1}(X-\mu'_{ij})\right]}\Bigg|_{X=X_{kj}}. \tag{42}$$

Therefore, from equations (37), (39) and (41)

$$Q(\lambda|\lambda') = \sum_{j=1}^{M} T_j \log \alpha_j + \sum_{j=1}^{M}\sum_{i=1}^{G_j}\sum_{k=1}^{T_j} \log\left[\pi_{ij}p_{ij}(X_{kj}|\lambda_{ij})\right] w'_{ij}(X_{kj}). \tag{43}$$

Note that if $\pi_{ij} = 0$ for some i and j, then $Q(\lambda|\lambda') = -\infty$.

The M-step of the EM method is achieved by maximizing $Q(\lambda|\lambda')$ with respect to the parameter set λ given the previous estimate λ'. In [17], Juang proves that maximizing Q is equivalent to maximizing $\mathcal{L}$; hence, an iterative procedure for maximizing Q over the parameter set λ will cause the likelihood function $\mathcal{L}$ to monotonically increase. This maximization problem is solved either by differentiating Q with respect to each parameter in the set λ, setting the resulting partial derivatives equal to zero, and then solving for each parameter, or by the method of Lagrange multipliers if parameter constraints are necessary. In the following development, it

turns out that the parameters $\{\alpha, \pi, \mu\}$ are uncoupled from each other and from Σ in the equations defining the necessary conditions for maximization, so they can be solved for separately. The parameter Σ is a function of the parameters $\{\mu_{ij}\}$.

The expression for Q in equation (43) may be rewritten as

$$\begin{aligned} Q(\lambda|\lambda') &= \sum_{j=1}^{M} T_j \log \alpha_j + \sum_{j=1}^{M}\sum_{i=1}^{G_j}\sum_{k=1}^{T_j} w'_{ij}(X_{kj}) \log \pi_{ij} \\ &\quad + \sum_{j=1}^{M}\sum_{i=1}^{G_j}\sum_{k=1}^{T_j} w'_{ij}(X_{kj}) \log p_{ij}(X_{kj}|\lambda_{ij}) \qquad (44) \\ &= Q_\alpha + \sum_{j=1}^{M} Q_\pi^j + Q_p. \qquad (45) \end{aligned}$$

Note that Q_α depends only on $\{\alpha_j\}$, that Q_π^j depends only on $\{\pi_{ij}, i = 1, \ldots, G_j\}$, and that Q_p depends jointly on Σ and the vectors $\{\mu_{ij}, i = 1, \ldots, G_j, j = 1, \ldots, M\}$. In addition, to maximize Q_π^j for each j, we require that $\pi_{ij} > 0$ for all i. Consequently, if Q_α, Q_π^j and Q_p are each maximized separately, then Q is also maximized (see Juang [17] for the special case of M = 1).

Starting with the parameter set $\{\alpha_j,\}$, Q_α is maximized subject to constraint (13). The appropriate Lagrangian for $\{\alpha_j\}$ is

$$\tilde{Q} \equiv Q_\alpha + \gamma \left(\sum_{j=1}^{M} \alpha_j - 1 \right), \qquad (46)$$

where γ is the Lagrange multiplier. Differentiating with respect to α_j yields

$$\frac{\partial \tilde{Q}}{\partial \alpha_j} = \frac{T_j}{\alpha_j} + \gamma = 0; \qquad (47)$$

hence,

$$\alpha_j = -\frac{T_j}{\gamma}. \qquad (48)$$

Substituting into the constraint (13) yields

$$1 = -\sum_{j=1}^{M} \frac{T_j}{\gamma} = -\frac{1}{\gamma} \sum_{j=1}^{M} T_j = -\frac{T}{\gamma}. \qquad (49)$$

Therefore, $\gamma = -T$, and

$$\hat{\alpha}_j = \frac{T_j}{T}. \tag{50}$$

Note that the estimates $\{\hat{\alpha}_j\}$ are an immediate consequence of the labelling (partitioning) of the training set $\mathcal{T}$and that $\hat{\alpha}_j > 0$, as required. By Lemma 2 in [10], $\{\hat{\alpha}_j\}$ is the unique global maximum of Q_α.

To estimate $\{\pi_{ij}, i = 1, \ldots, G_j\}$, the term Q_π^j is maximized subject to the constraint (16). In this case, the appropriate Lagrangian is

$$\tilde{Q} = Q_\pi^j + \gamma \left(\sum_{i=1}^{G_j} \pi_{ij} - 1 \right), \tag{51}$$

where γ is the Lagrange multiplier. Taking the partial derivative with respect to π_{ij} yields

$$\frac{\partial \tilde{Q}}{\partial \pi_{ij}} = \sum_{k=1}^{T_j} w'_{ij}(X_{kj}) \left(\frac{1}{\pi_{ij}} + \gamma \right) = 0 \tag{52}$$

or

$$\pi_{ij} = -\frac{1}{\gamma} \sum_{k=1}^{T_j} w'_{ij}(X_{kj}). \tag{53}$$

Substituting (53) into constraint equation (16) results in

$$-\frac{1}{\gamma} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w'_{ij}(X_{kj}) = 1. \tag{54}$$

Since

$$\sum_{i=1}^{G_j} w'_{ij}(X_{kj}) = 1, \tag{55}$$

it follows that $\gamma = -T_j$. Hence, the estimate of the mixing proportion is, from equation (53),

$$\hat{\pi}_{ij} = \frac{1}{T_j} \sum_{k=1}^{T_j} w'_{ij}(X_{kj}). \tag{56}$$

Note that $\hat{\pi}_{ij} > 0$, as required. Again, by Lemma 2 of [10], the estimate $\{\hat{\pi}_{ij}\}$ is the unique global maximum of Q_π^j.

The new estimate of the covariance matrix Σ is found by differentiating Q_p with respect to $\Sigma = [\Sigma_{ij}]$:

$$\begin{aligned} \nabla_\Sigma Q_p &= \sum_{j=1}^{M}\sum_{i=1}^{G_j}\sum_{k=1}^{T_j}\frac{1}{2}w'_{ij}(X_{kj})\left[-\Sigma^{-1}+\Sigma^{-1}(X_{kj}-\mu_{ij})(X_{kj}-\mu_{ij})^t\Sigma^{-1}\right] \\ &= 0, \end{aligned} \tag{57}$$

where ∇_Σ is defined as the matrix operator

$$\nabla_\Sigma \equiv \left[\frac{\partial}{\partial\Sigma_{ij}}\right]_{i,j=1}^{N}. \tag{58}$$

From equation (42),

$$\sum_{j=1}^{M}\sum_{k=1}^{T_j}\sum_{i=1}^{G_j}w'_{ij}(X_{kj}) = \sum_{j=1}^{M}\sum_{k=1}^{T_j}1 = \sum_{j=1}^{M}T_j = T, \tag{59}$$

and the estimate of the covariance matrix is therefore

$$\hat{\Sigma} = \frac{1}{T}\sum_{j=1}^{M}\sum_{i=1}^{G_j}\sum_{k=1}^{T_j}w'_{ij}(X_{kj})(X_{kj}-\mu_{ij})(X_{kj}-\mu_{ij})^t. \tag{60}$$

Note that $\hat{\Sigma}$ is in the convex hull of outer products of vectors, and that the mean vectors μ_{ij} in equation (60) are the maximum likelihood estimates of the true means.

The estimate of the mean vector μ_{ij} is also found by differentiating Q_p:

$$\nabla_{\mu_{ij}}Q_p = \sum_{k=1}^{T_j}w'_{ij}(X_{kj})\Sigma^{-1}(X_{kj}-\mu_{ij}) = 0. \tag{61}$$

Note that in this case, μ_{ij} is defined as a vector of length N, and that ∇_μ is defined for general vectors $\mu = (\mu_1, \ldots, \mu_N)$ as

$$\nabla_\mu \equiv \left[\frac{\partial}{\partial\mu_l}\right]_{l=1}^{N}. \tag{62}$$

Hence, from equation (61) the estimate of the mean vector μ_{ij} is

$$\hat{\mu}_{ij} = \frac{\sum_{k=1}^{T_j}w'_{ij}(X_{kj})X_{kj}}{\sum_{k=1}^{T_j}w'_{ij}(X_{kj})}. \tag{63}$$

Note that the estimate $\hat{\mu}_{ij}$ is in the convex hull of the training set for label class j, and that $\hat{\mu}_{ij}$ is independent of the covariance matrix estimate $\hat{\Sigma}$.

Equations (60) and (63) produce the unique global maximum of Q_p, provided a requlority condition is imposed on the training set $\mathcal{T}$. For vectors $c_j \in \mathbf{R}^N$, $j = 1, \ldots, M$, define the set

$$\Gamma(c_1, \ldots, c_M) = \mathcal{H}\left[\bigcup_{j=1}^{M} \{X_{kj} + c_j\}_{k=1}^{T_j}\right] \subset \mathbf{R}^N, \tag{64}$$

where $\mathcal{H}[\cdot]$ denotes the closed convex hull. The training set $\mathcal{T}$ is defined to be "full rank" if the set $\Gamma(c_1, \ldots, c_M)$ has at least $N + 1$ extreme points for every choice of the vectors $\{c_j\}$. The full rank assumption on $\mathcal{T}$ guarantees that the estimated covariance matrix $\hat{\Sigma}$ given by equation (60) is positive definite.

The following theorem proves that iteration step of the EM method is well defined for GF training. The theorem *does not* imply global maximum likelihood convergence of the GF training algorithm.

Theorem *If the training set $\mathcal{T}$ is full rank, and if the mixing proportions $\pi_{ij} \neq 0$ for all i and j, then Q_p has a unique global maximum as a function of the covariance matrix Σ and the mean vectors $\{\mu_{ij}, i = 1, \ldots, G_j, j = 1, \ldots, M\}$.*

Before proving this theorem, note that the full rank requirement on $\mathcal{T}$ is not very restrictive. For example, if any one class has $N + 1$ training samples, and after arbitrary translation, any N of the translated samples are linearly independent, then the pooled training set $\mathcal{T}$ is full rank. However, the set $\mathcal{T}$ can also have full rank even if *none* of the individual classes in the pool have full rank. This is an important consequence of pooling across classes. For instance, if $M = N$ and each class has two samples, that is, $\mathcal{T}_j = \{X_{1j}, X_{2j}\}$, then the set $\mathcal{T}$ is full rank if and only if the vectors $\{X_{1j} - X_{2j}\}_{j=1}^{M}$ span $\mathbf{R}^N$. Also, note that class training sets that contain only one sample do not contribute to the rank of $\mathcal{T}$. Clearly, the geometric condition on the convex hull embodies many equivalent algebraic statements.

The following proof of the theorem is in the spirit of Liporace's proof of a similar result [18, Appendix]. From the definition of Q_p in (45) and $p_{ij}(X|\lambda_{ij})$ in (15),

$$Q_p = \frac{1}{2}\sum_{j=1}^{M}\sum_{i=1}^{G_j}\sum_{k=1}^{T_j} w'_{ij}(X_{kj})\left[-N\log(2\pi) + \log|\Lambda| - (X_{kj}-\mu_{kj})^t\Lambda(X_{kj}-\mu_{kj})\right]. \tag{65}$$

where $\Lambda = \Sigma^{-1}$. Parameterizing Q_p in terms of the precision matrix Λ, instead of the covariance matrix Σ, allows the development of an explicitly negative expression for the second derivative of Q_p at a critical point. Let the point $\lambda = \{\Lambda, \mu_{ij}, i = 1,\ldots,G_j, j = 1,\ldots,M\}$ satisfy the necessary conditions (60) and (63) for a critical point of Q_p. Expressing λ as a convex combination of two arbitrary points λ^1 and λ^2 in the domain of Q_p such that $\lambda^1 \neq \lambda^2, \lambda^1 \neq \lambda$ and $\lambda^2 \neq \lambda$ yields

$$\mu_{ij} = \theta\mu^1_{ij} + (1-\theta)\mu^2_{ij} \tag{66}$$

and

$$\Lambda = \theta\Lambda^1 + (1-\theta)\Lambda^2. \tag{67}$$

Note that θ in (66) and (67) is uniquely defined, is independent of the indices i and j, and satisfies $0 < \theta < 1$. Substituting (66) and (67) into (65) and differentiating twice with respect to θ yields (after tedious calculation)

$$\frac{\partial^2 Q_p}{\partial\theta^2} = -R - \frac{1}{2}\sum_{j=1}^{M}\sum_{i=1}^{G_j}\sum_{k=1}^{T_j} w'_{ij}(X_{kj})\left(R^1_{ij} + R^2_{ijk}\right), \tag{68}$$

where

$$R = \frac{T}{2}\sum_{l=1}^{N}\left[\frac{D^1_l - D^2_l}{\theta D^1_l + (1-\theta)D^2_l}\right]^2, \tag{69}$$

$$R^1_{ij} = 2(\mu^1_{ij} - \mu^2_{ij})^t\Lambda(\mu^1_{ij} - \mu^2_{ij}), \tag{70}$$

$$R^2_{ijk} = 4(\mu^1_{ij} - \mu^2_{ij})^t\Lambda(X_{kj} - \theta\mu^1_{ij} - (1-\theta)\mu^2_{ij}), \tag{71}$$

D^1_l and D^2_l are the diagonal entries of $U^t\Lambda^1U$ and $U^t\Lambda^2U$ respectively, and where U is a nonsingular matrix diagonalizing Λ^1 and Λ^2 simultaneously. Note that $R \geq 0$,

and that R = 0 if and only if $\Lambda^1 = \Lambda^2$. (N.B. Had we not reparameterized in terms of Λ, this term would be nonpositive). Because the training set $\mathcal{T}$ is full rank, Λ is positive definite; thus, the term $R^1_{ij} \geq 0$, and $R^1_{ij} = 0$ if and only if $\mu^1_{ij} = \mu^2_{ij}$ for some i and j. At least one of the terms $\{R, w'_{ij}(X_{kj})R^1_{ij}\}$ is strictly positive because $\lambda^1 \neq \lambda^2$ and because $\pi_{ij} \neq 0$ for all i and j (this implies $w'_{ij}(X_{kj}) > 0$). The term R^2_{ijk} does not vanish, but the sum over its indices does vanish; that is,

$$\begin{aligned}
&\sum_{j=1}^{M}\sum_{i=1}^{G_j}\sum_{k=1}^{T_j} w'_{ij}(X_{kj})R^2_{ijk} \\
&= \sum_{j=1}^{M}\sum_{i=1}^{G_j} 4(\mu^1_{ij} - \mu^2_{ij})^t \Lambda \sum_{k=1}^{T_j} w'_{ij}(X_{kj})(X_{kj} - \theta\mu^1_{ij} - (1-\theta)\mu^2_{ij}) \\
&= 0. \qquad (72)
\end{aligned}$$

because of the necessary condition (61) at a critical point (recall the definition of μ_{ij} at a critical point from equation (66)). It follows that

$$\frac{\partial^2 Q_p}{\partial \theta^2} < 0. \qquad (73)$$

Hence, a critical point of Q_p is a local maximum. Since Q_p has a unique critical point, all that remains to be shown is that Q_p attains its maximum. Because the training set $\mathcal{T}$ is full rank, Q_p is bounded above. But $Q_p \to -\infty$ uniformly as its defining parameter vector goes to the point at infinity, so the supremum of Q_p must be a maximum (i.e., Q_p attains its maximum).

To summarize the GF iteration, first note that the mixing proportions $\{\alpha_j\}$ may be computed at the beginning from equation (50):

$$\alpha_j = \frac{T_j}{T}. \qquad (74)$$

Now let $\lambda^{(n)}$ be available from the previous iteration, and define the weights

$$w^{(n)}_{ij}(X_{kj}) = \frac{\pi^{(n)}_{ij} \exp\left[-\frac{1}{2}(X_{kj} - \mu^{(n)}_{ij})^t (\Sigma^{(n)})^{-1}(X_{kj} - \mu^{(n)}_{ij})\right]}{\sum_{l=1}^{G_j} \pi^{(n)}_{lj} \exp\left[-\frac{1}{2}(X_{kj} - \mu^{(n)}_{lj})^t (\Sigma^{(n)})^{-1}(X_{kj} - \mu^{(n)}_{li})\right]}. \qquad (75)$$

The new intercomponent mixing proportions are updated using equation (56):

$$\pi_{ij}^{(n+1)} = \frac{1}{T_j} \sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj}). \tag{76}$$

Since $\pi_{ij}^{(n+1)} = 0$ if and only if $\pi_{ij}^{(n)} = 0$, GF training *can not* be initialized with any zero mixing proportions. Specifically, we require that $\pi_{ij}^{(0)} \neq 0$ for all i and j. The mean vectors are updated using equation (63):

$$\mu_{ij}^{(n+1)} = \frac{\sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj}) X_{kj}}{\sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj})}. \tag{77}$$

The new covariance matrix is found from equation (60):

$$\Sigma^{(n+1)} = \frac{1}{T} \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj})(X_{kj} - \mu_{ij}^{(n+1)})(X_{kj} - \mu_{ij}^{(n+1)})^t. \tag{78}$$

Convergence of the GF algorithm can be tested in two ways. First, $Q(\lambda^{(n+1)}|\lambda^{(n)})$ can be computed using equation (43):

$$Q(\lambda^{(n+1)}|\lambda^{(n)}) = \sum_{j=1}^{M} T_j \log \alpha_j - \log\left((2\pi)^{N/2}|\Sigma^{(n+1)}|^{1/2}\right) \tag{79}$$

$$+ \sum_{j=1}^{M} \sum_{i=1}^{G_j} \sum_{k=1}^{T_j} w_{ij}^{(n)}(X_{kj}) \left\{ \log \pi_{ij}^{(n+1)} - \frac{1}{2}(X_{kj} - \mu_{ij}^{(n+1)})^t (\Sigma^{(n+1)})^{-1}(X_{kj} - \mu_{ij}^{(n+1)}) \right\}.$$

Then $Q(\lambda^{(n+1)}|\lambda^{(n)})$ can be compared to $Q(\lambda^{(n)}|\lambda^{(n-1)})$ to determine if the parameter estimates have stabilized. If the estimates have stabilized, then the algorithm is terminated. Alternatively, GF training can be terminated if the likelihood function $\mathcal{L}(\mathcal{T}|\lambda^{(n)})$ as a function of n ceases to increase at a sufficient rate.

References

[1] S. J. Press, *Applied Multivariate Analysis.* Malabar, Florida: Kreiger Publishing, 1982.

[2] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley-Interscience, 1973.

[3] R. L. Streit, "A neural network for optimum Neyman-Pearson classification," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 685–690, June 1990. San Diego, California, Volume I.

[4] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, pp. 4–22, April 1987.

[5] D. F. Specht, "Probabilistic neural networks for classification, mapping, or associative memory," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 525–532, July 1988. Volume I.

[6] E. Parzen, "On estimation of a probability density function," *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38, 1977. (Article includes discussion.).

[8] R. A. Redner, R. J. Hathaway, and J. C. Bezdek, "Estimating the parameters of mixture models with modal estimators," *Communications in Statistics, Part A: Theory and Methods*, vol. 16, pp. 2639–2660, 1987.

[9] R. L. Streit, "Class priors for entropy maximization," Technical Memorandum 911144, Naval Underwater Systems Center, June 1991.

[10] S. S. Wilks, "Determination of sample sizes for setting tolerance limits," *Annals of Mathematical Statistics*, vol. 12, pp. 91–96, 1941.

[11] H. Robbins, "On distribution-free tolerance limits in random sampling," *Annals of Mathematical Statistics*, vol. 15, pp. 214–216, 1944.

[12] H. Scheffé and J. W. Tukey, "Non-parametric estimation I. Validation of order statistics," *Annals of Mathematical Statistics*, vol. 16, pp. 187–192, 1945.

[13] A. Wald, "An extension of Wilks' method for setting tolerance limits," *Annals of Mathematical Statistics*, vol. 14, pp. 45–55, 1943.

[14] J. W. Tukey, "Non-parametric estimation II. Statistically equivalent blocks and tolerance regions — the continuous case," *Annals of Mathematical Statistics*, vol. 18, pp. 529–539, 1947.

[15] J. W. Tukey, "Non-parametric estimation III. Statistically equivalent blocks and multivariate tolerance regions — the discontinous case," *Annals of Mathematical Statistics*, vol. 19, pp. 30–39, 1948.

[16] R. B. Murphy, "Non-parametric tolerance limits," *Annals of Mathematical Statistics*, vol. 19, pp. 581–589, 1948.

[17] B. Juang, "Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains," *AT&T Technical Journal*, vol. 64, pp. 1235–1249, 1985.

[18] L. R. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Transactions on Information Theory*, vol. IT-28, pp. 729–734, September 1982.

[19] L. R. Rabiner, S. E. Levinson, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process in automatic speech recognition," *The Bell System Technical Journal*, vol. 62, pp. 1035–1074, 1983.

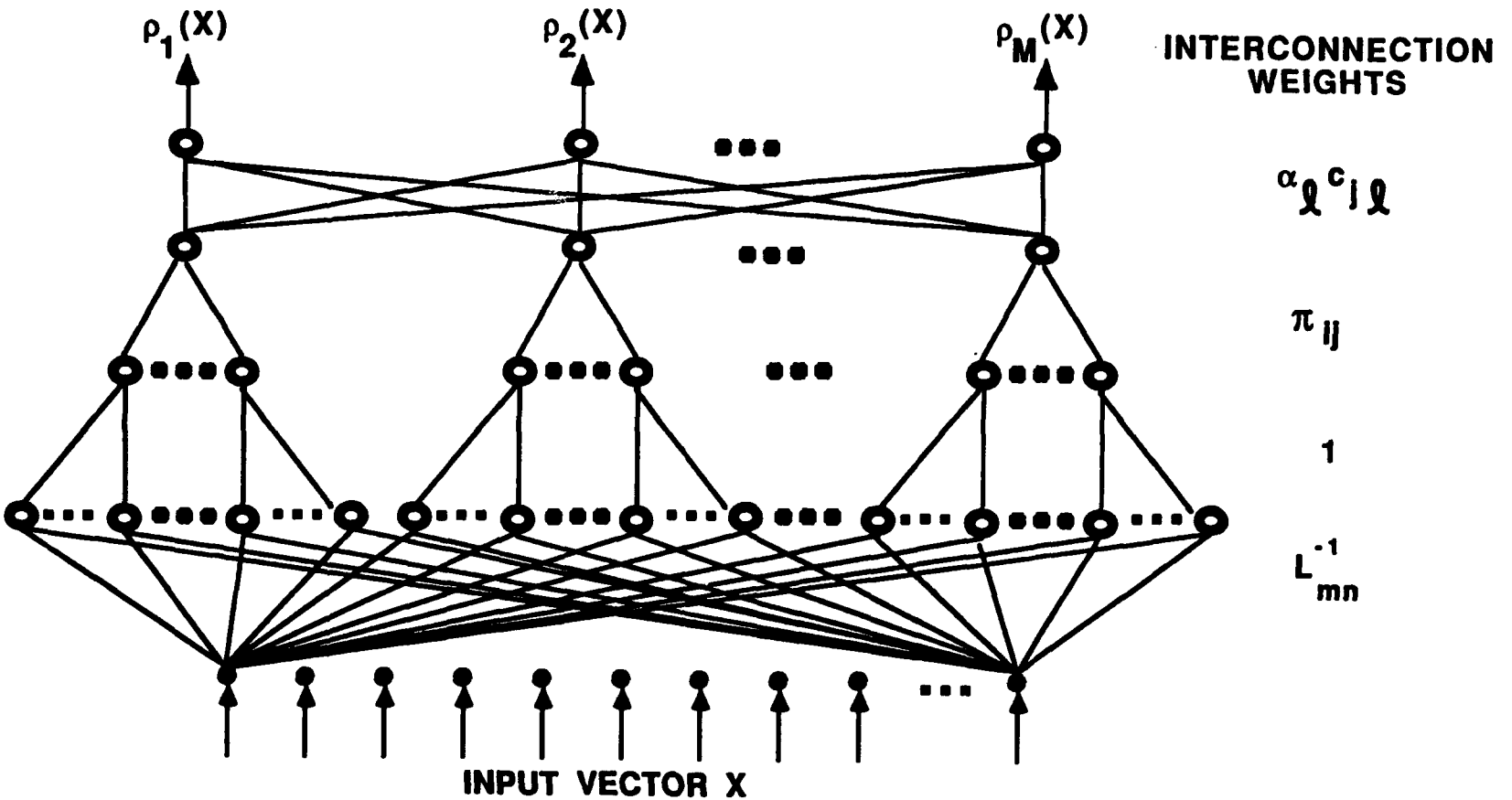


Figure 1. Four Layer Feed-Forward Probabilistic Neural Network

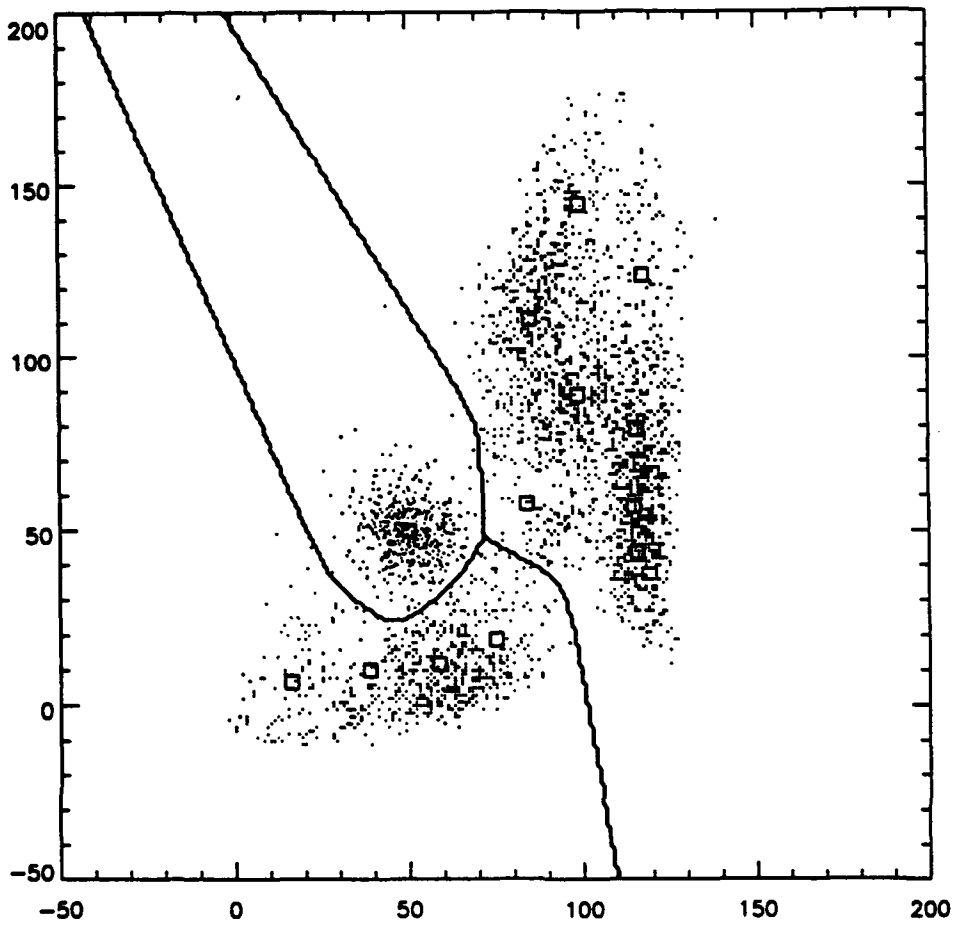


Figure 2 Optimum Discriminant Boundary Curve, With Scatter Plot of All Samples Superimposed

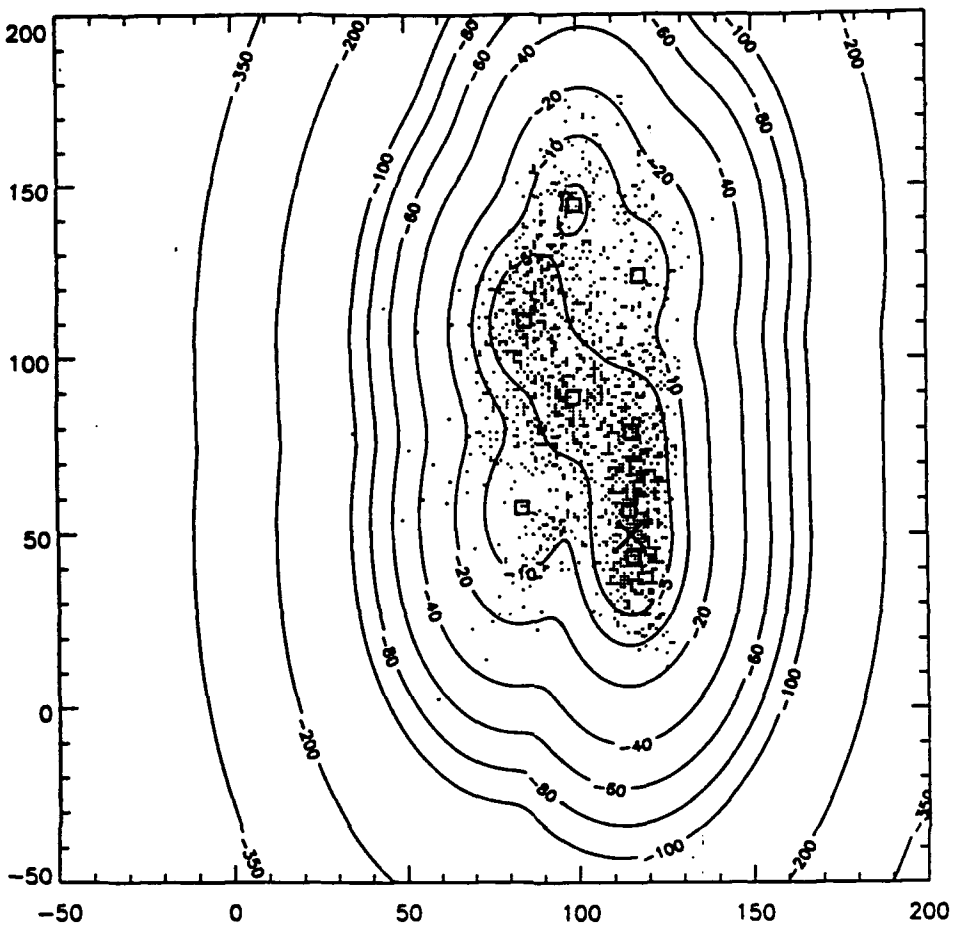


Figure 3 Contours of Class #1 Conditional PDF in dB//max, With Scatter Plot Superimposed

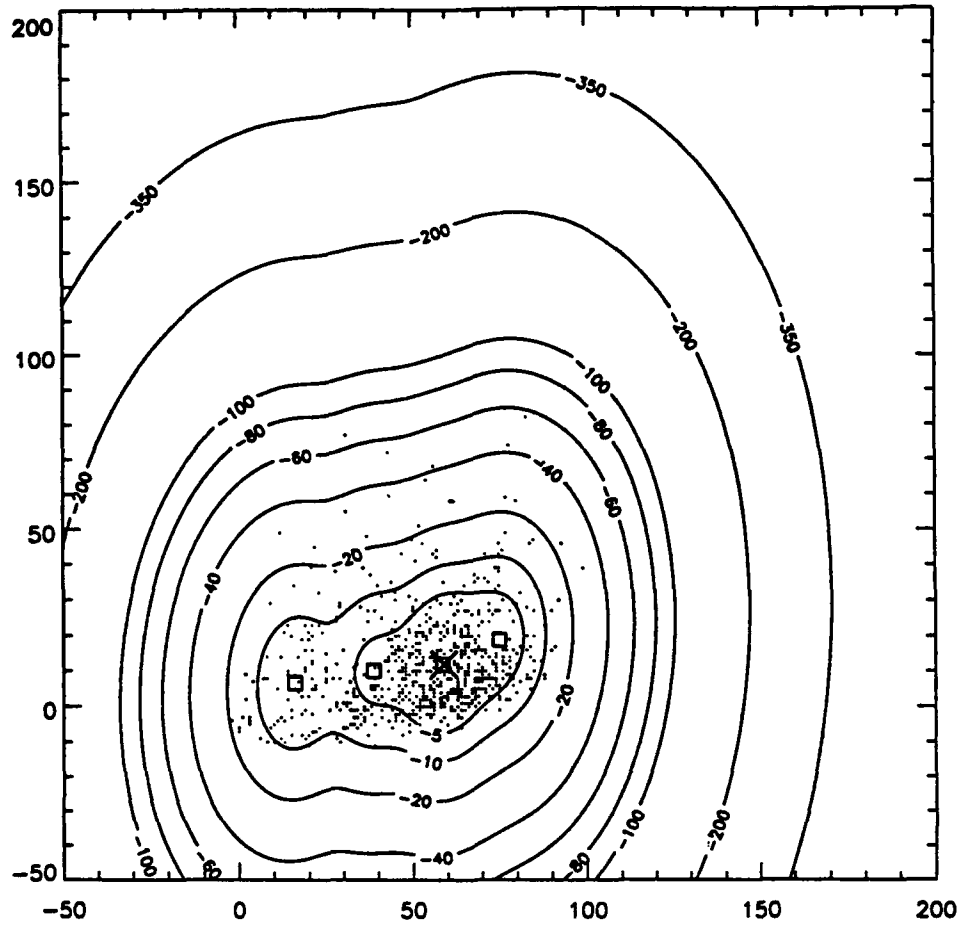


Figure 4 Contours of Class #2 Conditional PDF in dB//max, With Scatter Plot Superimposed

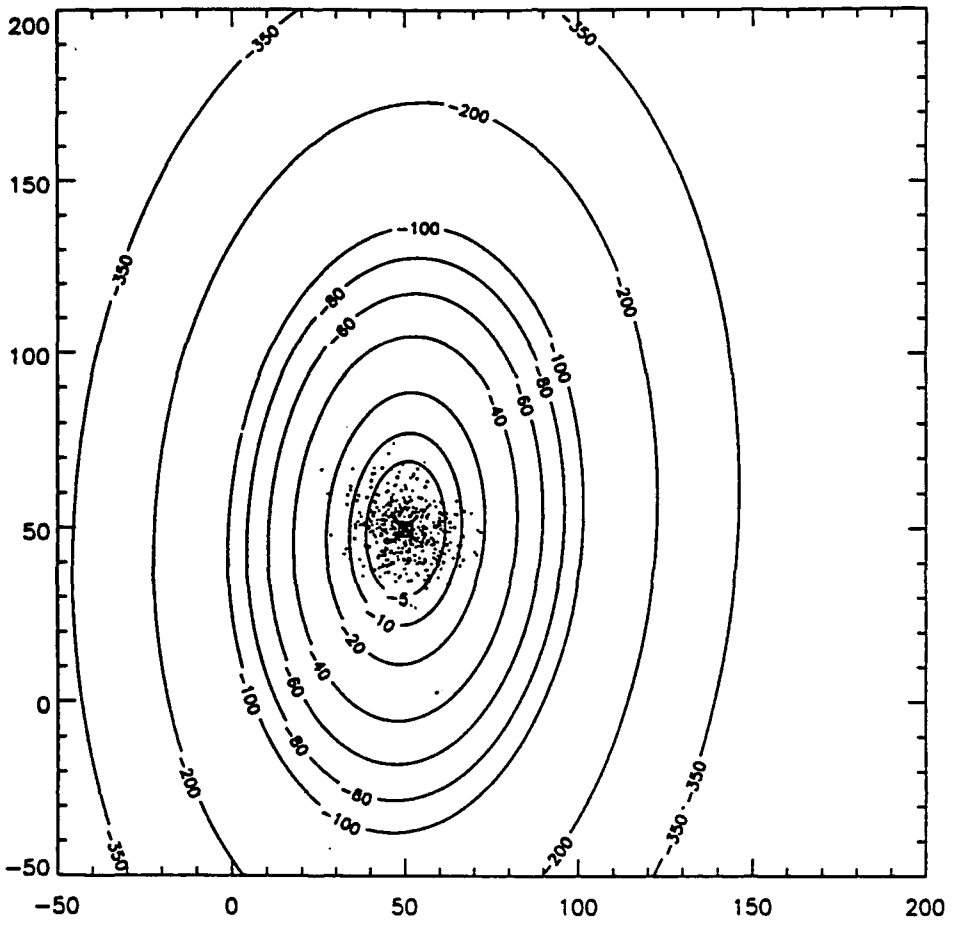


Figure 5 Contours of Class #3 Conditional PDF in dB//max, With Scatter Plot Superimposed

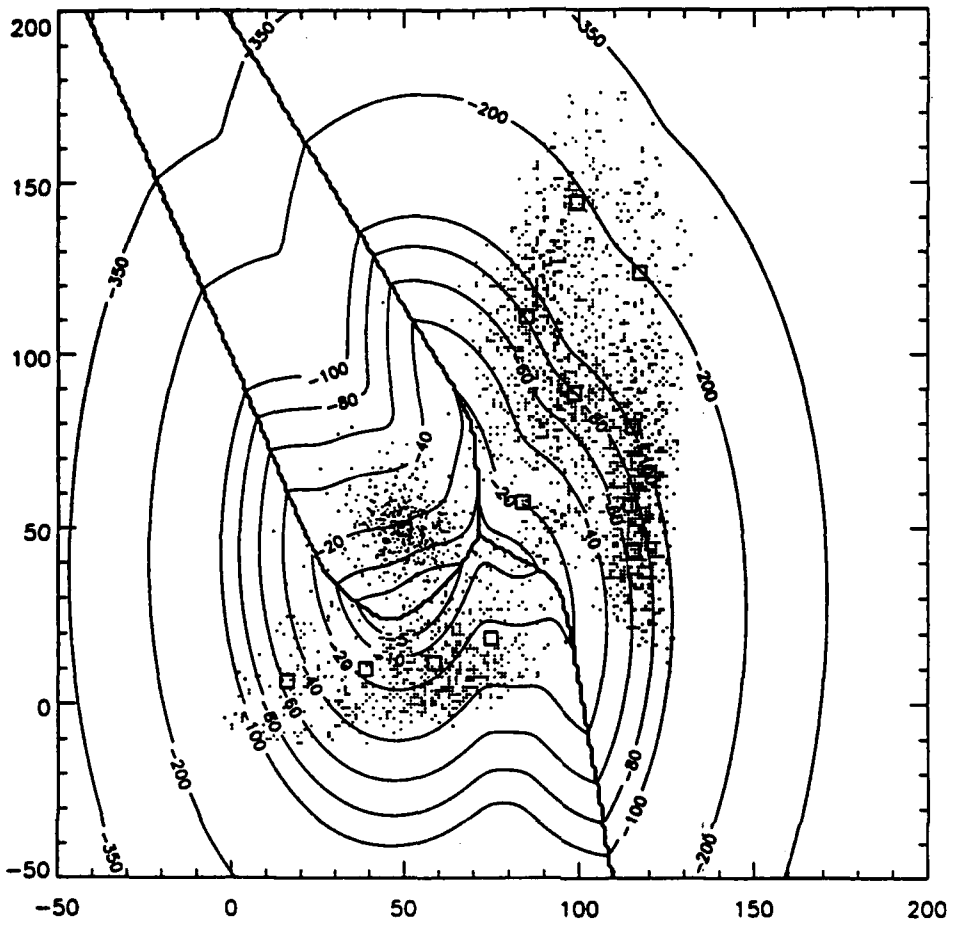


Figure 6 Contours of Decision Risk $\rho(X)$ for Example #1 in dB//max

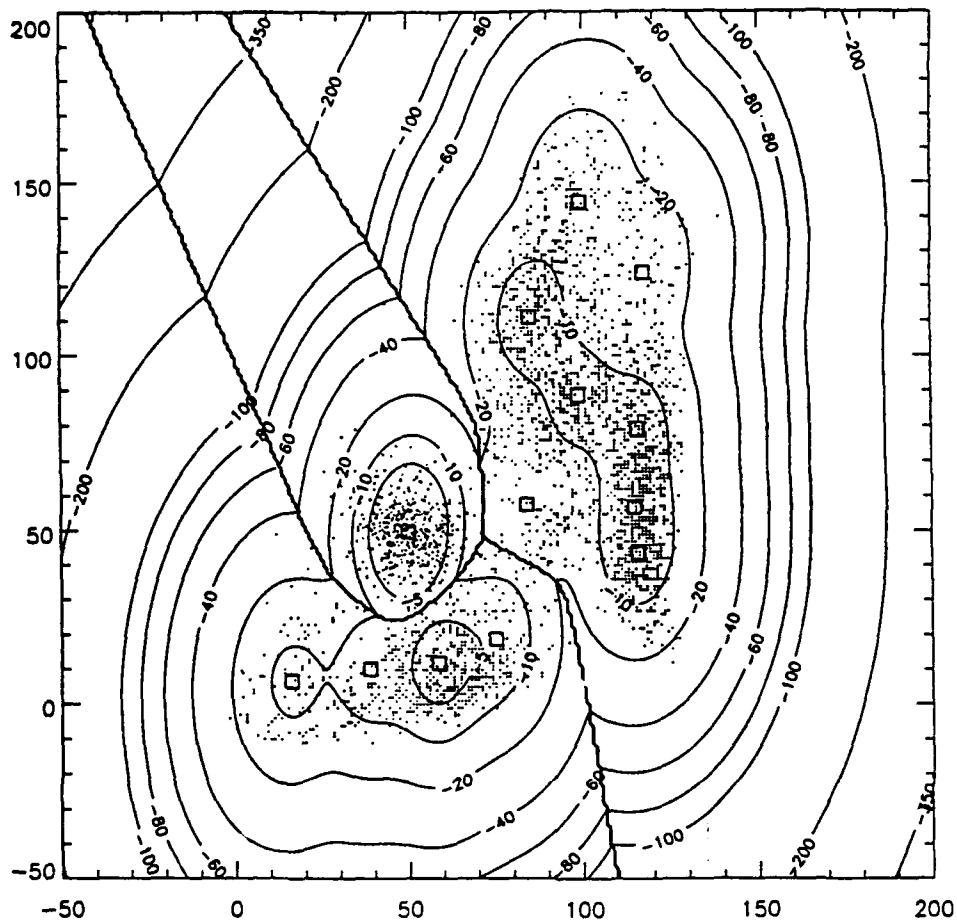


Figure 7 Contours of Decision Assurance $\delta(X)$ for Example #1 in dB//max

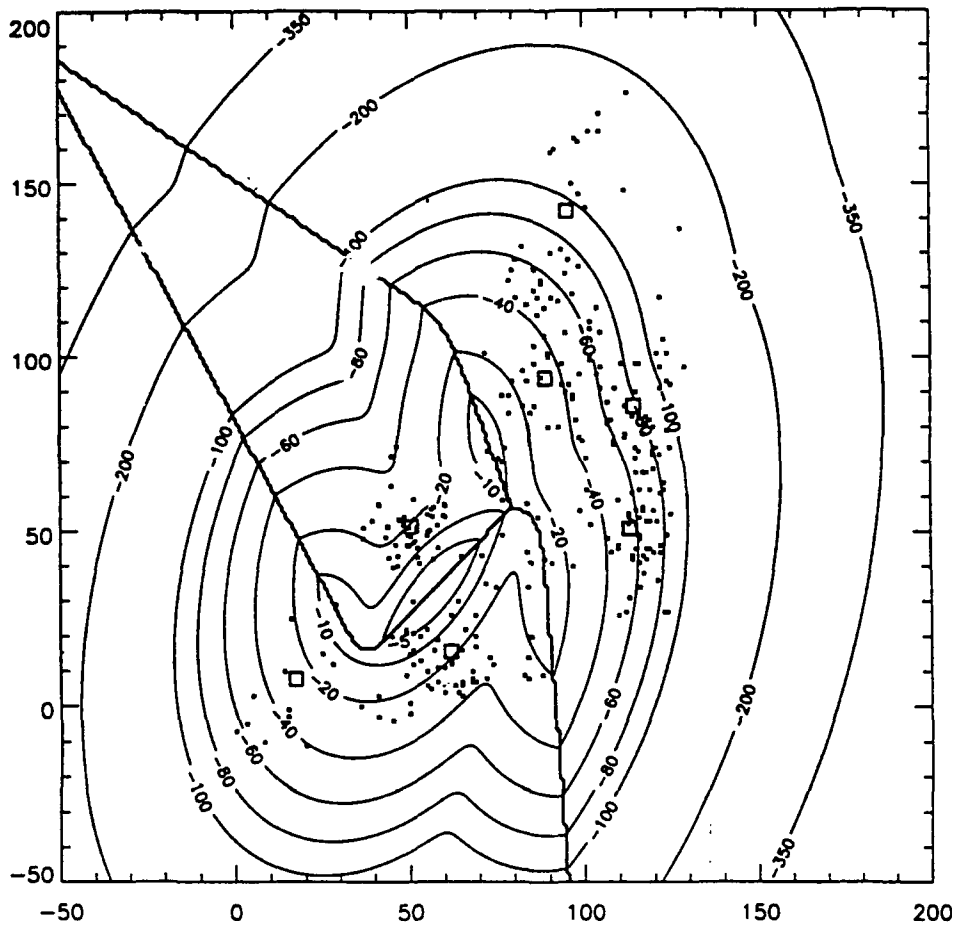


Figure 8 Contours of Decision Risk $\rho(X)$ for Example #2 in dB//max

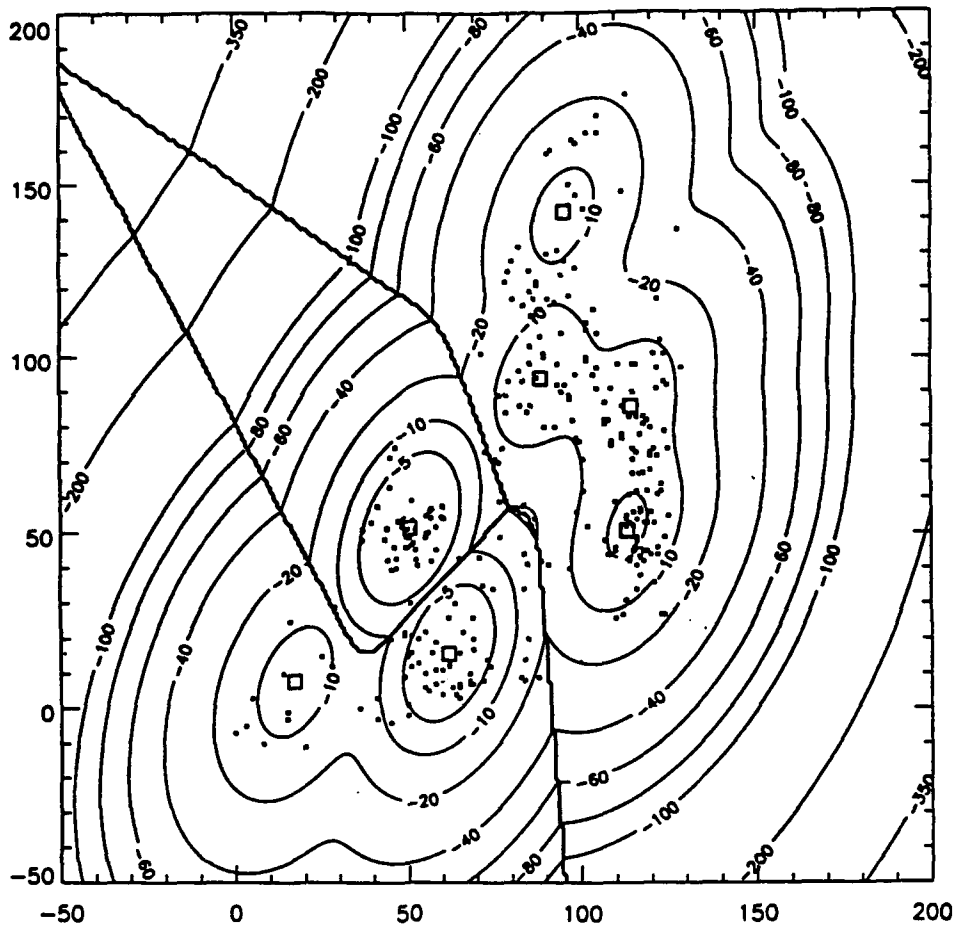


Figure 9 Contours of Decision Assurance $\delta(X)$ for Example #2 in dB//max

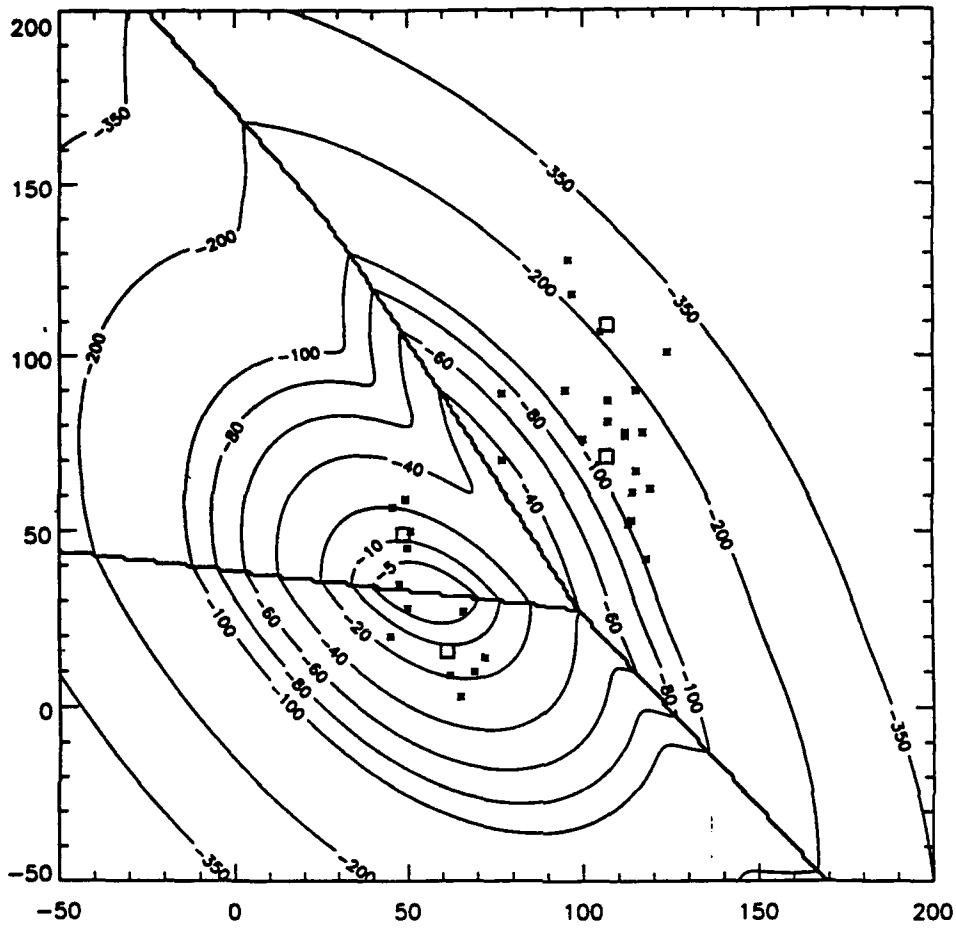


Figure 10 Contours of Decision Risk $\rho(X)$ for Example #3 in dB//max

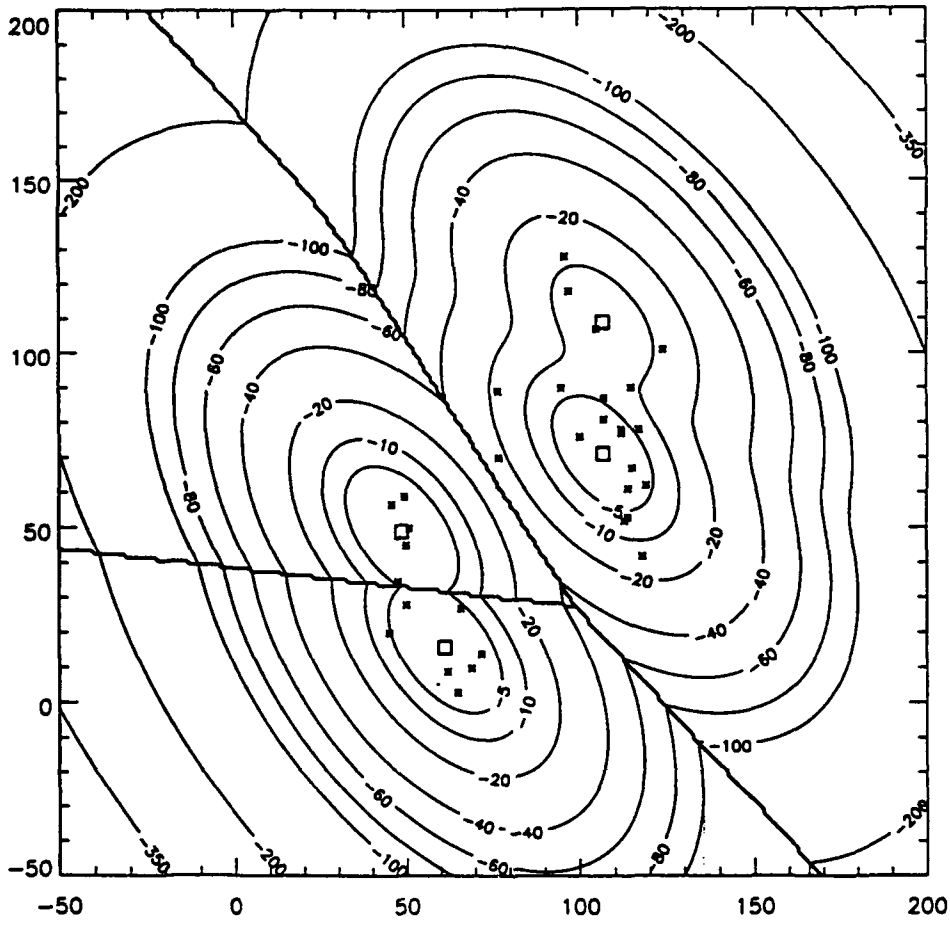


Figure 11 Contours of Decision Assurance $\delta(X)$ for Example #3 in dB//max

Class Priors For Entropy Maximization

R. L. Streit

Abstract

Optimum Bayesian classification requires knowledge of class *a priori* probabilities. Maximum entropy class priors are proposed here as a natural choice for applications in which the class *a priori* probabilities are unavailable and cannot be estimated.

CLASS PRIORS FOR ENTROPY MAXIMIZATION

INTRODUCTION

The correct scalings of class probability density functions (PDF's) for optimum classification are the class *a priori* probabilities. This note derives "natural" class priors that are useful in applications in which class *a priori* probabilities are unknown. Their utility stems from the natural way in which they scale class PDF's to accomodate relative variations in PDF support and peakedness. A theoretical justification for these priors in terms of maximum entropy is derived.

PRESENTATION

It is assumed here that class PDF's are estimated from available class training samples. The differential entropy H(X) of a random vector variable X defined on $\mathbb{R}^n$ with continuous PDF p(x) is defined by

$$H(X) = -\int_{\mathbb{R}^n} p(x) \log p(x) \, dx. \qquad (1)$$

The integral in (1) is the expected value of the function log p(x). If $T \geq 1$ samples of X are given, and from these samples an estimate $\hat{p}(x)$ for p(x) has been developed, then the (posterior) sample entropy is defined here by

$$\hat{H}(X) = -\frac{1}{T}\sum_{k=1}^{T} \log \hat{p}(x_k), \qquad (2)$$

where $\{x_1, \ldots, x_T\}$ denotes the available samples of X. Note that

$$\exp(-\hat{H}(X)) = \left[\prod_{k=1}^{T} \hat{p}(x_k) \right]^{1/T}.$$

Thus, $\exp(-\hat{H}(X))$ is exactly the geometric mean of the numbers $\{\hat{p}(x_k)\}$.

Suppose that there are $m \geq 1$ classes characterized by the random vector variables $X_1, \ldots, X_m$ defined on R^n with corresponding continuous PDF's $p_1(x), \ldots, p_m(x)$. Given training samples of these random variables, estimates $\hat{p}_1(x), \ldots, \hat{p}_m(x)$ of the class PDF's are developed by methods (e.g., Parzen windows) not relevant to the present discussion. The proposed prior for each class is proportional to the reciprocal of the geometric mean of the estimated PDF. Let $\hat{H}(X_i)$ denote the sample entropy (*cf.* equation (2)) of X_i derived from the available training samples for X_i, and let α_i denote the proposed prior for class i. Then α_i is given by

$$\alpha_i = \frac{\exp(\hat{H}(X_i))}{\sum_{k=1}^{m} \exp(\hat{H}(X_k))}. \qquad (3)$$

The denominator in equation (3) normalizes α_i so that $\Sigma\, \alpha_i = 1$. Intuitively, the priors $\{\alpha_i\}$ scale down high peaks of class PDF's whose support is compact, and scale up the broad plateaus of class PDF's whose support is widely distributed.

The class prior probabilities (3) can be written in terms of class entropy powers. The entropy power N(X) of a general random variable X is defined by

$$N(X) = \frac{1}{2\pi e} \exp(2H(X)). \qquad (4)$$

The entropy power N(X) is the average noise power of a Gaussian random variable having differential entropy H(X). The prior α_i is clearly proportional to the square root of the sample entropy power $\hat{N}(X_i)$ of

the i-th class.

Theoretical justification for use of the class priors (3) proceeds as follows. Let $\mathcal{Y}$ denote a memoryless source modeling the class sampling process. $\mathcal{Y}$ is a two step process. The first step selects a random variable, or class symbol, X_i from the list $\{X_1, \ldots, X_m\}$ with probability α_i. The second step selects a sample x from the random variable X_i selected in the first step. The likelihood of x is $p_i(x)$. The source $\mathcal{Y}$ output is the pair (X_i, x). An important distinction is that $\mathcal{Y}$ is not equivalent to a source $\mathcal{P}$ whose outputs x are selected from a random vector variable whose PDF is the mixture

$$p(x) = \sum_{i=1}^{m} \alpha_i \, p_i(x).$$

The difference between the sources $\mathcal{Y}$ and $\mathcal{P}$ is that output from $\mathcal{Y}$ contains class label information, while the output from $\mathcal{P}$ does not. The differential entropy of $\mathcal{Y}$ is

$$H(\mathcal{Y}) = - \sum_{i=1}^{m} \int_{R^n} \alpha_i \, p_i(x) \, \log [\alpha_i \, p_i(X)] \, dx. \qquad (5)$$

Equation (5) is derived by noting that the output symbol (X_i, x) occurs with likelihood $\alpha_i \, p_i(x)$, so that the differential entropy is the appropriate sum and integral of this likelihood multiplied by its logarithm. It follows easily from equation (5) that

$$H(\mathcal{Y}) = - \sum_{i=1}^{m} \alpha_i \left[\log \alpha_i - H(X_i) \right]. \qquad (6)$$

We seek a stationary point for $H(\mathcal{Y})$ over all $\{\alpha_i\}$ satisfying $\Sigma \, \alpha_i = 1$. A straightforward application of Lagrange multipliers shows that the unique stationary point is

$$\alpha_i = \frac{\exp(H(X_i))}{\sum_{k=1}^{m} \exp(H(X_k))} . \qquad (7)$$

This stationary point is a maximum for the differential entropy $H(\mathcal{Y})$, as can be shown by examining the second order derivatives. The priors (7) are therefore maximum entropy priors for the class sampling process $\mathcal{Y}$. The priors (3) are sample entropy versions of the maximum entropy priors (7). The maximum entropy $H(\mathcal{Y})$ corresponding to the priors (7) is the negative of the logarithm of the denominator of equation (7).

Given independent class training samples, class priors can be estimated by maximum likelihood (ML) methods. As is easily shown, ML class *a priori* probability estimates are proportional to the relative abundance of the individual class training samples. ML priors are thus independent of class PDF structure. Unfortunately, in practice the class training samples are very often screened in such a way that the relative abundances of class samples contains no information about class priors. In such situations, it is common practice to choose diffuse class priors, that is, all class *a priori* probabilities are set equal to $1/m$. Diffuse priors are thus blind to class training samples and class PDF structure. The maximum entropy priors proposed here are a more natural choice than diffuse priors for Bayesian classification applications because they maximize the entropy of the class selection process $\mathcal{Y}$. Maximum entropy priors defined by equation (7) depend on class PDF structure and are independent of relative class training sample abundance.

MATHEMATICS

Foreword

Complex function approximation by linear combinations of complex valued basis functions is well understood, both mathematically and numerically, if least squared error is used to measure the closeness of the approximation. However, the difficulty of the complex approximation problem is dramatically increased by a change in the error criterion. Papers [25] - [28] study the problem using the l_∞ norm error criterion. (The l_∞ norm is known by several names, e.g., Chebyshev norm and maximum error.) Only two methods are currently known for solving general l_∞ approximation problems numerically. One approach is via iteratively reweighted least squares problems, and this approach is studied theoretically in paper [25].

The other approach to l_∞ approximation is via semi-infinite programming (SIP), a specialized variant of cutting plane methods for convex optimization. The SIP approach is presented in papers [26] - [28]. Application of this work to linear acoustic array beamforming problems is given in papers [1] - [3]. The provision of magnitude constraints in the SIP problem formulation also enables its application to active conformal array beamforming problems in which acoustic effects between the elements (projectors) must be limited (*cf.*, [paper 2, page 3]). An application of SIP methods to the solution of systems of indefinite finite difference equations by polynomial iteration methods is presented in paper [29].

Papers [30] and [31] describe concertina-like variations in the detailed structure of a special class of real valued functions (Haar systems) generalizing the Chebyshev polynomials. These theoretical results support observations made while studying the linear array problems described in papers [9] and [10]. The remaining papers [32] - [34] also document results encountered in the course of other investigations.

Saddle Points And Overdetermined Complex Equations

R. L. Streit

Saddle Points and Overdetermined Complex Equations

Roy L. Streit*
Naval Underwater Systems Center
New London Laboratory
New London, Connecticut 06320

Submitted by Richard A. Brualdi

ABSTRACT

It is known that the best uniform norm solution of overdetermined complex valued systems of equations satisfying the Haar condition for matrices is also a best weighted l_p norm solution for each $p \geq 1$, for some weight vector depending on p. This paper presents an alternative proof of this result which is valid for arbitrary matrices A. The proof relies on the fundamental theorem of game theory. It is shown that a saddle point (z^*, λ^*) of a certain function gives a uniform norm solution, z^*, of $Az = b$ and a weight vector λ^* of the equivalent weighted l_p norm problem. With appropriate qualifications concerning the weights, it follows that the worst (i.e., largest) possible weighted least l_p norm error is also the best (i.e., least) possible Chebyshev error. For $p = 2$, it is shown that the weight vector λ^* solves a nonlinear optimization problem which can be posed without reference to solution vectors of $Az = b$. In other words, the problem of finding the best uniform norm solution of $Az = b$, when stated as a convex optimization problem, has a convex dual which for $p = 2$ can be posed independently of the primal variables z. The dual variables are the weights λ.

I. INTRODUCTION

This paper is an investigation of solution of overdetermined systems of complex linear equations using the uniform, or l_∞, norm. An equivalent alternative context in which results can be presented and interpreted is complex function approximation on discrete point sets. It is in the latter context that Motzkin and Walsh [1, 2] prove that the best uniform norm solution of an overdetermined real system $Ax = b$ is equivalent to a weighted least pth power solution for each $p > 0$, assuming that the matrix A satisfies

*This work was supported by the Office of Naval Research Project RR014-07-01 and by the Independent Research Program of the Naval Underwater Systems Center.

the Haar condition for matrices. Their results are extended to complex systems by Lawson [3], who also gives an algorithm for constructing the weights of equivalent least pth power problems for $p > 1$. A nice summary of these results is given in [4, 5].

An alternative proof of the Motzkin-Walsh result for $p \geq 1$ is given in this paper. The proof does not assume the Haar condition and, in fact, is valid for arbitrary complex matrices A. With appropriate qualifications concerning the weights, it is also proved that the *worst* (i.e., largest) possible weighted least pth power error is also the *best* (i.e., least) possible uniform error. This result seems to be new, although it is implicit in the proof of the convergence of Lawson's algorithm. Lawson, however, does not state it explicitly.

Theorem 1 states that the function ϕ_p, defined by (1), has a saddle point in a certain domain. All other results follow essentially as corollaries. This approach is very different from that of Motzkin and Walsh. The proof of Theorem 1 relies on a connection between the solutions of overdetermined systems of equations (or, equivalently, approximation on discrete point sets) and the fundamental theorem of game theory. This relationship does not seem to be mentioned elsewhere.

The special case $p = 2$ is particularly interesting. In Theorem 3 below, it is shown that the weights for the equivalent least squares problem solve a nonlinear mathematical programming problem. So far as is known to the author, these weights have not been previously characterized in quite this manner. A special subcase is a problem posed by J. J. Sylvester in 1857. It is discussed in Section IV. Theorem 3 can also be used to prove a result due to de la Vallée Poussin [6] for real systems and extended to complex systems by Rivlin and Shapiro [7]. It is discussed in Section V.

Motzkin and Walsh prove their function approximation results on the interval $[0, 1]$ as well as on discrete point sets. It is therefore likely that greater generality is possible in our Theorem 1 which permits its application to systems having an infinite number of equations in a finite number of unknowns. For the purposes of this paper, however, Theorem 1 in its present form is satisfactory.

The Motzkin-Walsh results do not give insight into how the correct weights for the equivalent least pth power problem might be constructed. Lawson's original algorithm is apparently the only one currently available, and its convergence proof assumes the Haar condition for A. The algorithm requires the solution of a sequence of weighted least pth power problems, updating the weights at each step of the sequence. The correct weights are obtained in the limit. The special case $p = 2$ is of the greatest computational interest, since least square problems are easily solved. The major drawback to Lawson's algorithm is that convergence can be, and often is, very slow in practice [8].

Alternatives to Lawson's algorithm can be based on Theorems 1 and 2. In other words, general algorithms for computing a saddle point of a given continuous function can be applied to the problem at hand, i.e., to ϕ_p below. In particular, such algorithms would be applicable to the case $p = 1$ which Lawson's algorithm does not treat. Conversely, since the Lawson algorithm can now be interpreted as a procedure for computing a saddle point of ϕ_p for $p > 1$, it would be interesting to know whether or not Lawson's algorithm constitutes a special case of an existing algorithm for computing saddle points. If not, perhaps it can be extended to construct saddle points of more general functions.

II. THE THEOREMS

Let the complex matrix $A = [a_{ij}] \in C^{m \times n}$ and the vector $b = (b_i) \in C^m$ be given. Let $1 \leqslant p < \infty$. Define $\phi_p : C^n \times R^m \to R$ by

$$\phi_p(z, \lambda) = \left\{ \sum_{i=1}^{m} \lambda_i \left| b_i - \sum_{j=1}^{n} a_{ij} z_j \right|^p \right\}^{1/p}, \tag{1}$$

where $z = (z_i) \in C^n$ and $\lambda = (\lambda_i) \in R^m$. Define $\Lambda = \{\lambda \in R^m : \lambda \geqslant 0$ and $\lambda_1 + \cdots + \lambda_m = 1\}$. Note that Λ is convex. A point (z^*, λ^*) is defined to be a saddle point of ϕ_p on $C^n \times \Lambda$ if $(z^*, \lambda^*) \in C^n \times \Lambda$ and

$$\phi_p(z^*, \lambda) \leqslant \phi_p(z^*, \lambda^*) \leqslant \phi_p(z, \lambda^*).$$

The central result is the following theorem. Its proof relies on the fundamental theorem of game theory and is postponed to Section III.

THEOREM 1. *For every matrix $A \in C^{m \times n}$ and vector $b \in C^m$, the function ϕ_p has a saddle point on the set $C^n \times \Lambda$.*

No assertion of uniqueness of the saddle point is made by Theorem 1. Sufficient conditions for uniqueness are not pursued in this paper.

It should be noted that Theorem 1 is valid for all $n \geqslant 1$ and $m \geqslant 1$.

Define the *uniform norm* $\|\cdot\|_\infty$ of any vector in C^m to be the maximum modulus of its components. A Chebyshev solution of the system of equations $Az = b$ is any vector z^* for which

$$\|b - Az^*\|_\infty = \min_{z \in C^n} \|b - Az\|_\infty. \tag{2}$$

A weighted least pth power solution of $Az = b$ is any vector z^* for which

$$\phi_p(z^*, \lambda) = \min_{z \in C^n} \phi_p(z, \lambda), \tag{3}$$

where λ is the given set of nonnegative weights. The next theorem connects Chebyshev solutions and weighted least pth power solutions.

THEOREM 2. *Let $(z^*, \lambda^*) \in C^n \times \Lambda$ be a saddle point for ϕ_p, $1 \leqslant p < \infty$. Then*

(i) *z^* is a Chebyshev solution of $Az = b$;*

(ii) *z^* is a weighted least pth power solution of $Az = b$ for the weight vector λ^*.*

Furthermore, the saddle value $\phi_p(z^*, \lambda^*)$ is the error of the Chebyshev solution, i.e., $\phi_p(z^*, \lambda^*) = \|b - Az^*\|_\infty$.

Proof. By a well-known result [14, Theorem 3.15], ϕ_p has a saddle point (z^*, λ^*) on $C^n \times \Lambda$ if and only if

$$\max_{\lambda \in \Lambda} \min_{z \in C^n} \phi_p(z, \lambda) = \phi_p(z^*, \lambda^*) = \min_{z \in C^n} \max_{\lambda \in \Lambda} \phi_p(z, \lambda). \tag{4}$$

Let the largest of the m quantities $|b_i - \Sigma a_{ij} z_j|$ occur for, say, $i = k$ (depending of course on z). Then

$$\max_{\lambda \in \Lambda} \phi_p(z, \lambda) = \left| b_k - \sum_{j=1}^{n} a_{kj} z_j \right|,$$

as can be seen by taking $\lambda_k = 1$ and $\lambda_i = 0$ for $i \neq k$. Equivalently,

$$\max_{\lambda \in \Lambda} \phi_p(z, \lambda) = \|b - Az\|_\infty.$$

Consequently,

$$\phi_p(z^*, \lambda^*) = \min_{z \in C^n} \max_{\lambda \in \Lambda} \phi_p(z, \lambda) = \min_{z \in C^n} \|b - Az\|_\infty,$$

and z^* is a Chebyshev solution of $Az = b$. The saddle value $\phi_p(z^*, \lambda^*)$ is the Chebyshev error. Next note that

$$\phi_p(z^*, \lambda^*) = \max_{\lambda \in \Lambda} \min_{z \in C^n} \phi_p(z, \lambda) = \min_{z \in C^n} \phi_p(z, \lambda^*), \tag{5}$$

and so z^* is a weighted least pth power solution of $Az = b$ for the weight vector λ^*. This completes the proof. ■

COROLLARY 1. *Define, for all* $\lambda \in \Lambda$,

$$\psi_p(\lambda) = \min_{z \in C^n} \phi_p(z, \lambda). \tag{6}$$

Then, under the conditions of Theorem 2,

$$\psi_p(\lambda) \leqslant \psi_p(\lambda^*) = \|b - Az^*\|_\infty.$$

Proof. Immediate from (5). ■

Another way to say this is as follows. The error of a weighted least pth power solution of the system $Az = b$ is $\psi_p(\lambda)$, and this error is maximized over allowed weights $\lambda \in \Lambda$ for λ^*. Furthermore, the maximum of such an error equals the minimum of the Chebyshev error of $Az = b$.

COROLLARY 2. *The vector b is a linear combination of the columns of the matrix A if and only if the saddle value of* ϕ_p *is zero.*

Proof. Let (z^*, λ^*) be a saddle point of ϕ_p. If $\phi_p(z^*, \lambda^*) = \|b - Az^*\|_\infty = 0$, then $b = Az^*$. Conversely, if $b = Az$ for some $z \in C^n$, then $\psi_p(\lambda) = 0$ for all λ, so that $\max \psi_p(\lambda) = 0 = \phi_p(z^*, \lambda^*)$. ■

The vectors z^* and λ^* are defined jointly via the saddle point property of ϕ_p. Theorem 2 shows that z^* also solves an optimization problem that does not require knowledge of λ^*; that is, z^* is a Chebyshev solution of $Az = b$. In many cases this property alone will uniquely determine z^*. Theorem 3 below will show that an analogous situation exists with regard to λ^* for the special case $p = 2$. The distinction of the case $p = 2$ is a consequence of the fact that $\psi_2(\lambda)$, as defined by (6), can be expressed explicitly in terms of λ alone. For other values of p, use of an implicit function theorem seems to be necessary.

Define the complex matrix $L(\lambda) = [L_{ij}(\lambda)] \in C^{n \times n}$ by

$$L(\lambda) = A^H \operatorname{diag}(\lambda) A, \tag{7}$$

where A^H is the conjugate transpose of A, and $\operatorname{diag}(\lambda)$ is the $m \times m$ diagonal

matrix whose main diagonal consists of the components of λ. Thus,

$$L_{\nu j}(\lambda) = \sum_{i=1}^{m} \lambda_i \bar{a}_{i\nu} a_{ij}, \qquad \nu, j = 1, \ldots, n. \tag{8}$$

The complex matrix $M(\lambda) \in C^{(n+1)\times(n+1)}$ is defined by bordering $L(\lambda)$:

$$M(\lambda) = \left[\begin{array}{c|c} L(\lambda) & \bar{\beta}(\lambda) \\ \hline \beta(\lambda) & \alpha(\lambda) \end{array}\right], \tag{9}$$

where $\beta(\lambda) = (\beta_j(\lambda)) \in C^n$ is given by

$$\beta_j(\lambda) = \sum_{i=1}^{m} \lambda_i \bar{b}_i a_{ij}, \qquad j = 1, \ldots, n, \tag{10}$$

and $\alpha(\lambda) \in R$ is given by

$$\alpha(\lambda) = \sum_{i=1}^{m} \lambda_i \bar{b}_i b_i. \tag{11}$$

Note that both $L(\lambda)$ and $M(\lambda)$ are Hermitian matrices.

The matrix $A \in C^{m\times n}$ with $m \geq n$ is said to satisfy the Haar condition for matrices if and only if every collection of n rows of A has rank n.

THEOREM 3. *Let $m > n$, and let A satisfy the Haar condition for matrices. Let (z^*, λ^*) be a saddle point of $\phi_2(z, \lambda)$ on $C^n \times \Lambda$ with saddle value $\phi_2(z^*, \lambda^*) > 0$. Then*

$$\min_{z \in C^n} \|b - Az\|_\infty = \|b - Az^*\|_\infty \tag{12}$$

$$= \phi_2(z^*, \lambda^*) \tag{13}$$

$$= \left[\frac{\det M(\lambda^*)}{\det L(\lambda^*)}\right]^{1/2} \tag{14}$$

$$= \max_{\lambda \in \Lambda} \left[\frac{\det M(\lambda)}{\det L(\lambda)}\right]^{1/2}, \tag{15}$$

where the maximum (15) *is taken over $\lambda \in \Lambda$ with* $\det L(\lambda) \neq 0$.

Proof. Both (12) and (13) follow from Theorems 1 and 2, so it is necessary to prove only (14) and (15). The definition (6) for $p = 2$ is

$$\psi_2(\lambda) = \min_{z \in C^n} \left\{ \sum_{i=1}^{m} \lambda_i \left| b_i - \sum_{j=1}^{n} a_{ij} z_j \right|^2 \right\}^{1/2}. \tag{16}$$

By Corollary 1,

$$\phi_2(z^*, \lambda^*) = \max_{\lambda \in \Lambda} \psi_2(\lambda). \tag{17}$$

Since the saddle value is positive, we restrict attention throughout the remainder of this proof to those vectors λ for which $\psi_2(\lambda) > 0$. (By Corollary 2, the vector b is linearly independent of the columns of A.) Since A satisfies the Haar condition for matrices, it follows that λ has $n+1$ or more positive components, for otherwise it is easy to see from (16) that $\psi_2(\lambda) = 0$. Consequently, the Hermitian form of $L(\lambda)$,

$$z^H L(\lambda) z = z^H A^H \operatorname{diag}(\lambda) A z = \sum_{i=1}^{m} \lambda_i \left| \sum_{j=1}^{n} a_{ij} z_j \right|^2,$$

must be positive for nonzero vectors z. Hence, $\det L(\lambda) \neq 0$, and the normal equations

$$L(\lambda) z = \left[A^H \operatorname{diag}(\lambda) A \right] b = \bar{\beta}(\lambda) \tag{18}$$

are nonsingular. It is convenient within the confines of this proof to define the auxiliary symbols

$$z_{n+1} = -1, \qquad a_{i,n+1} = b_i, \quad i = 1, \ldots, n. \tag{19}$$

Rewriting the normal equations

$$\sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_i \bar{a}_{i\nu} a_{ij} z_j = \sum_{i=1}^{m} \lambda_i b_i \bar{a}_{i\nu}, \qquad \nu = 1, \ldots, n, \tag{20}$$

and using the symbols (19) gives

$$\sum_{j=1}^{n+1} \sum_{i=1}^{m} \lambda_i \bar{a}_{i\nu} a_{ij} z_j = 0, \qquad \nu = 1, \ldots, n. \tag{21}$$

Now, from (16), for any z satisfying the normal equations,

$$\psi_2^2(\lambda) = \sum_{i=1}^{m} \lambda_i \left| b_i - \sum_{j=1}^{n} a_{ij} z_j \right|^2$$

$$= \sum_{i=1}^{m} \lambda_i \left| \sum_{j=1}^{n+1} a_{ij} z_j \right|^2$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n+1} \sum_{\nu=1}^{n+1} \lambda_i a_{ij} z_j \bar{a}_{i\nu} \bar{z}_\nu .$$

Reversing the order of the triple sum gives

$$\psi_2^2(\lambda) = \sum_{\nu=1}^{n+1} \bar{z}_\nu \sum_{j=1}^{n+1} \sum_{i=1}^{m} \lambda_i \bar{a}_{i\nu} a_{ij} z_j$$

$$= - \sum_{j=1}^{n+1} \sum_{i=1}^{m} \lambda_i \bar{a}_{i,n+1} a_{ij} z_j , \tag{22}$$

where, in the last equation, (21) was used to set to zero the double sum for the cases $\nu = 1, \ldots, n$, and for $\nu = n+1$ the value $z_{n+1} = -1$ was substituted. Rewriting (22) without the symbols (19) gives

$$\psi_2^2(\lambda) + \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_i \bar{b}_i a_{ij} z_j = \sum_{i=1}^{m} \lambda_i \bar{b}_i b_i . \tag{23}$$

Thus (20) and (23) constitute $n+1$ equations in the $n+1$ unknowns z_j and $\psi_2^2(\lambda)$. This system can be written

$$\left[\begin{array}{c|c} L(\lambda) & 0 \\ \hline \beta(\lambda) & 1 \end{array}\right] \left[\begin{array}{c} z \\ \hline \psi_2^2(\lambda) \end{array}\right] = \left[\begin{array}{c} \bar{\beta}(\lambda) \\ \hline \alpha(\lambda) \end{array}\right].$$

The coefficient matrix clearly has rank $n+1$ and so is nonsingular. Solving for $\psi_2^2(\lambda)$ using Cramer's rule gives

$$\psi_2^2(\lambda) = \frac{\det M(\lambda)}{\det L(\lambda)} . \tag{24}$$

Substituting (24) for ψ_2 in (17) concludes the proof. ■

If the Haar condition hypothesis on A in Theorem 3 is replaced by rank $A = n$, then the result (15) does not hold in general, because $\det L(\lambda^*)$ can vanish at a saddle point (z^*, λ^*) for ϕ_2. Consider the following example. Let $m = n + 1$. Let the first n rows of A be the identity matrix, and let the mth row of A be identically 0. Let $b_1 = \cdots = b_n = 0$ and $b_m = \gamma > 0$. One saddle point (z^*, λ^*) of ϕ_2 is $z_1^* = \cdots = z_n^* = 0$ and $\lambda_1^* = \cdots = \lambda_n^* = 0$, $\lambda_{n+1}^* = 1$. The saddle value $\phi_2(z^*, \lambda^*) = \gamma > 0$, but the $n \times n$ matrix $L(\lambda^*)$ contains only zero entries and $\det L(\lambda^*) = 0$. Note that λ^* is unique in this example.

The determinants in (24) are actually Gram determinants for the indefinite inner product on C^m defined by

$$(u, v) = \sum_{i=1}^{m} \lambda_i u_i \bar{v}_i \qquad \text{for} \quad \lambda \geq 0.$$

For a definition of Gram determinants, their properties, and a nearly equivalent derivation of (24), see [9, pp. 176–187].

III. PROOF OF THEOREM 1

The proof applies the fundamental theorem of game theory to the function ϕ_p. The variant of the fundamental theorem that is utilized is stated for functions defined on Cartesian products of convex subsets of real Euclidean spaces. Although ϕ_p is defined on $C^n \times \Lambda$, by an obvious device it can be thought of as being defined on $R^{2n} \times \Lambda$ instead, without suffering any loss of generality in what follows. The complex notation is retained for ease of exposition.

The function ϕ_p is continuous in both variables and, as the next lemma shows, it is a convex function in its first variable and a concave function in its second variable.

LEMMA 1. *For $1 \leq p < \infty$, the function ϕ_p is convex-concave on $C^n \times \Lambda$; that is, for $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$,*

$$\phi_p(\alpha z + \beta w, \lambda) \leq \alpha\phi_p(z, \lambda) + \beta\phi_p(w, \lambda),$$

$$\phi_p(z, \alpha\lambda + \beta\gamma) \geq \alpha\phi_p(z, \lambda) + \beta\phi_p(z, \gamma),$$

where z and w are elements of C^n, and λ and γ are elements of Λ.

Proof. Let $t_i = b_i - \sum a_{ij} z_j$ and $s_i = b_i - \sum a_{ij} w_j$. Then

$$b_i - \sum_1^n a_{ij}(\alpha z_j + \beta w_j) = \alpha t_i + \beta s_i.$$

From the definition of ϕ_p,

$$\begin{aligned}\phi_p(\alpha z + \beta w, \lambda) &= \left\{ \sum_{i=1}^m \left| \lambda_i^{1/p}(\alpha t_i + \beta s_i) \right|^p \right\}^{1/p} \\ &\leqslant \left\{ \sum_{i=1}^m \left| \lambda_i^{1/p} \alpha t_i \right|^p \right\}^{1/p} + \left\{ \sum_{i=1}^m \left| \lambda_i^{1/p} \beta s_i \right|^p \right\}^{1/p} \\ &= \alpha \phi_p(z, \lambda) + \beta \phi_p(w, \lambda),\end{aligned}$$

which proves that ϕ_p is a convex function in its first argument for each λ. To prove that ϕ_p is a concave function in its second argument, fix z and let $Q_\lambda = \phi_p(z, \lambda)$ and $Q_\gamma = \phi_p(z, \gamma)$. The case $p = 1$ is obvious, so assume $1 < p < \infty$ and let q satisfy $1/p + 1/q = 1$. Then

$$\begin{aligned}\alpha \phi_p(z, \lambda) + \beta \phi_p(z, \gamma) &= \alpha Q_\lambda + \beta Q_\gamma \\ &= \left(\alpha^{1/p} Q_\lambda\right)\left(\alpha^{1/q}\right) + \left(\beta^{1/p} Q_\gamma\right)\left(\beta^{1/q}\right) \\ &\leqslant \left\{ \left(\alpha^{1/p} Q_\lambda\right)^p + \left(\beta^{1/p} Q_\gamma\right)^p \right\}^{1/p} \\ &\quad \times \left\{ \left(\alpha^{1/q}\right)^q + \left(\beta^{1/q}\right)^q \right\}^{1/q} \\ &= \left\{ \alpha Q_\lambda^p + \beta Q_\gamma^p \right\}^{1/p} \\ &= \left\{ \alpha \sum_{i=1}^m \lambda_i |t_i|^p + \beta \sum_{i=1}^m \gamma_i |t_i|^p \right\}^{1/p} \\ &= \phi_p(z, \alpha\lambda + \beta\gamma).\end{aligned}$$

This completes the proof of the lemma. ∎

The following theorem is due to H. Kneser [10]. See also [11, pp. 8–13].

THEOREM 4. *Let $E \subset R^N$ and $F \subset R^M$ be convex sets. Let the function $f: E \times F \to R$ be convex on E for each fixed $y \in F$ and concave on F for each fixed $x \in E$. If one of the sets E and F is compact, and if the function f is continuous in the corresponding variable, then*

$$\sup_{y \in F} \inf_{x \in E} f(x, y) = \inf_{x \in E} \sup_{y \in F} f(x, y).$$

Applying Kneser's theorem to ϕ_p gives

$$\sup_{\lambda \in \Lambda} \inf_{z \in C^n} \phi_p(z, \lambda) = \inf_{z \in C^n} \sup_{\lambda \in \Lambda} \phi_p(z, \lambda).$$

Now, by a standard argument, for each λ the infimum

$$\inf_{z \in C^n} \phi_p(z, \lambda)$$

is attained for some z. Furthermore, the resulting function of λ is continuous on the compact set Λ and so attains its supremum. Thus, the supinf can be replaced by maxmin. Similarly, for each z, the supremum

$$\sup_{\lambda \in \Lambda} \phi_p(z, \lambda)$$

is attained for some λ, since ϕ_p is continuous on the compact set Λ. By a standard argument, the resulting function of z attains its infimum. Hence,

$$\max_{\lambda \in \Lambda} \min_{z \in C^n} \phi_p(z, \lambda) = \min_{z \in C^n} \max_{\lambda \in \Lambda} \phi_p(z, \lambda). \tag{25}$$

The existence of a saddle point follows immediately. Choose λ^* such that

$$\min_{z \in C^n} \phi_p(z, \lambda^*) = \max_{\lambda \in \Lambda} \min_{z \in C^n} \phi_p(z, \lambda).$$

Choose z^* such that

$$\max_{\lambda \in \Lambda} \phi_p(z^*, \lambda) = \min_{z \in C^n} \max_{\lambda \in \Lambda} \phi_p(z, \lambda).$$

Then

$$\phi_p(z, \lambda^*) \geq \phi_p(z^*, \lambda^*) \qquad \text{for all} \quad z \in C^n$$

and

$$\phi_p(z^*, \lambda) \leq \phi_p(z^*, \lambda^*) \qquad \text{for all} \quad \lambda \in \Lambda.$$

The last two inequalities, by definition, show that (z^*, λ^*) is a saddle point of ϕ_p. This completes the proof of Theorem 1.

IV. SYLVESTER'S PROBLEM

It is worthwhile observing the form (15) takes in the special case of finding the best complex constant to fit given complex data. Specifically, find $z^* \in C$ such that

$$\max_{1 \leq i \leq m} |b_i - z^*| = \min_{z \in C} \max_{1 \leq i \leq m} |b_i - z|. \tag{26}$$

This problem is equivalent to a problem posed by J. J. Sylvester in 1857, i.e., given m points in the plane, find the smallest circle containing them all. The center of the smallest circle is the constant z^*, and the radius is the min max in (26). In this case, the matrix $L(\lambda) = [\lambda_1 + \cdots + \lambda_m] = [1]$ is the 1×1 identity matrix, and $M(\lambda)$ is the 2×2 matrix

$$M(\lambda) = \begin{bmatrix} 1 & \sum_{i=1}^{m} \lambda_i b_i \\ \sum_{i=1}^{m} \lambda_i \bar{b}_i & \sum_{i=1}^{m} \lambda_i |b_i|^2 \end{bmatrix}.$$

Hence, from (15), we need to compute

$$\max_{\lambda \in \Lambda} \left\{ \sum_{i=1}^{m} \lambda_i |b_i|^2 - \left| \sum_{i=1}^{m} \lambda_i b_i \right|^2 \right\}^{1/2}. \tag{27}$$

The problem (27) is equivalent to the quadratic program:

$$\text{QP.} \qquad \min_{\lambda \in R^m} \lambda^T G \lambda - c^T \lambda$$

$$\text{subject to} \quad \lambda \geq 0 \text{ and } \lambda_1 + \cdots + \lambda_m = 1,$$

where the real matrix $G \in R^{m \times m}$ is given by $G = \mathrm{Re}(bb^H) = (\mathrm{Re}\, b)(\mathrm{Re}\, b)^T + (\mathrm{Im}\, b)(\mathrm{Im}\, b)^T$, and the components of the real vector $c = (c_i) \in R^m$ are given by $c_i = |b_i|^2$. It is easy to see that G is a positive semidefinite matrix of rank at

most 2. The objective function of QP is thus convex, and any locally optimal solution λ^* of QP is a global solution. It then follows from the normal equations (18) that the best constant is given by

$$z^* = \sum_{i=1}^{m} \lambda_i^* b_i. \tag{28}$$

The maximum in (27) is positive when $m > 1$ and at least two of the data points b_i are distinct. To see that (27) is nonnegative, simply note that

$$\begin{aligned}\left|\sum_{i=1}^{m} \lambda_i b_i\right|^2 &\leq \left\{\sum_{i=1}^{m} |\lambda_i^{1/2} b_i| \lambda_i^{1/2}\right\}^2 \\ &\leq \sum_{i=1}^{m} |\lambda_i^{1/2} b_i|^2 \sum_{i=1}^{m} |\lambda_i^{1/2}|^2 \\ &= \sum_{i=1}^{m} \lambda_i |b_i|^2.\end{aligned}$$

The second inequality is strict under the conditions cited, so the maximum is positive.

Discussion of the history of Sylvester's problem, together with an efficient computational algorithm for its solution, is given in [12]. The algorithm given there solves the natural extension of Sylvester's problem to data points given in higher dimensional Euclidean spaces.

It is curious that a simple alteration of the problem substantially alters the difficulty of its solution. Instead of (26), consider

$$\max_{1 \leq i \leq m} |b_i - a_i z^*| = \min_{z \in C} \max_{1 \leq i \leq m} |b_i - a_i z| \tag{29}$$

for $a = (a_i) \in C^m$ given. The matrix $L(\lambda)$ is

$$L(\lambda) = \left[\lambda_1 |a_1|^2 + \cdots + \lambda_m |a_m|^2\right] \in C^{1 \times 1},$$

and the matrix $M(\lambda)$ is

$$M(\lambda) = \begin{bmatrix} \sum_{i=1}^{m} \lambda_i |a_i|^2 & \sum_{i=1}^{m} \lambda_i \bar{a}_i b_i \\ \sum_{i=1}^{m} \lambda_i a_i \bar{b}_i & \sum_{i=1}^{m} \lambda_i |b_i|^2 \end{bmatrix} \in C^{2 \times 2}.$$

Clearly, computing the maximum of $\det M(\lambda)/\det L(\lambda)$ is not simply equivalent to a quadratic program in this case. (Computationally, however, it may be solvable by parametric quadratic programming methods.) Given a solution vector λ^*, then

$$z^* = \frac{\sum_{i=1}^{m} \lambda_i^* \bar{a}_i b_i}{\sum_{i=1}^{m} \lambda_i^* |a_i|^2} \tag{30}$$

is a solution of (29), as can be seen from the normal equations (18).

When all the given data points b_i are real, the problem (26) has a trivial solution. Let r and s be indices for which $\min b_i$ and $\max b_i$ occur, respectively. The best constant z^* in (26) is real and $z^* = (b_r + b_s)/2$. Considering (28), it is evident that a solution of (27) is $\lambda_r^* = \lambda_s^* = \frac{1}{2}$ with all other $\lambda_i^* = 0$. The maximum in (27) is thus equal to $|b_r - b_s|/2$, a fact not immediately apparent from (27) itself.

V. NEW PROOF OF DE LA VALLÉE POUSSIN'S THEOREM

Let $A \in C^{(n+1)\times n}$, and let A_i denote the $n \times n$ matrix obtained from A by deleting the ith row, $i = 1, \ldots, n+1$. De la Vallée Poussin [6] proves the following result for real systems. Rivlin and Shapiro [7, pp. 692–694] show that it holds for complex systems also.

THEOREM 5. *Let $A \in C^{(n+1)\times n}$ satisfy the Haar condition for matrices, and let $b \in C^{n+1}$. Then*

$$\min_{z \in C^n} \|b - Az\|_\infty = \frac{\left| \sum_{i=1}^{n+1} (-1)^i b_i \det A_i \right|}{\sum_{i=1}^{n+1} |\det A_i|}. \tag{31}$$

We use Theorem 3 above to derive (31). The procedure is to solve the maximum problem (15) explicitly for λ^* and, from λ^*, deduce (31). As a side benefit, once λ^* is known, the Chebyshev solution vector z^* can be constructed numerically by solving the λ^* weighted least squares problem. In principle, this is equivalent to solving the normal equations (18) with $\lambda = \lambda^*$.

In contrast to the proof based on Theorem 3, the original proof of (31) proceeds by minimizing the number $Q = \|b - Az\|_\infty$ directly. It turns out that this can be done relatively easily and in such a way that the Chebyshev solution z^* can be constructed. Thus, the original proof and the proof based on Theorem 3 solve the "primal" and the "dual" problems, respectively.

We first establish a general algebraic identity concerning determinants. Special cases of this identity will be used in the solution of (15).

LEMMA 2. *For $i = 1,\dots,m$, let $\lambda_i \in C$, $x_i \in C^n$, and $y_i \in C^n$. Then, for $m \geq n \geq 1$,*

$$\det\left(\sum_{i=1}^{m} \lambda_i x_i y_i^T\right) = \sum\left(\prod_{t=1}^{n} \lambda_{i(t)}\right) \det[x_{i(1)},\dots,x_{i(n)}] \det[y_{i(1)},\dots,y_{i(n)}], \quad (32)$$

where $[x_{i(1)},\dots,x_{i(n)}]$ and $[y_{i(1)},\dots,y_{i(n)}]$ denote the $n \times n$ matrices whose t-th columns are $x_{i(t)}$ and $y_{i(t)}$, respectively, $t = 1,\dots,n$, and the sum is over all indices $i(1),\dots,i(n)$ such that $1 \leq i(1) < i(2) < \cdots < i(n) \leq m$. For $n > m \geq 1$, the determinant of the left hand side of (32) is identically zero.

Proof. Let $X \in C^{n\times m}$ and $Y \in C^{n\times m}$ denote the matrices whose tth columns are $\lambda_t x_t$ and y_t, respectively, $t = 1,\dots,m$. Then

$$\det\left(\sum_{i=1}^{m} \lambda_i x_i y_i^T\right) = \det(XY^T).$$

The Binet-Cauchy formula [13, pp. 8–10] for the determinant of the product of two rectangular matrices has two cases. For $n > m \geq 1$, it states that $\det(XY^T) = 0$. For $m \geq n \geq 1$, in the present notation, it gives $\det(XY^T) = \sum \det[\lambda_{i(1)}x_{i(1)},\dots,\lambda_{i(n)}x_{i(n)}]\det[y_{i(1)}^T,\dots,y_{i(n)}^T]$, with the sum ranging over all indices with $1 \leq i(1) < i(2) < \cdots < i(n) \leq m$. The identity (32) follows immediately by factoring $\lambda_{i(t)}$ out of column t in the first determinant, and noting that the determinant of a matrix and its transpose are equal. This concludes the proof. ∎

If $y_i = \bar{x}_i$ for all i, then

$$\det\left(\sum_{i=1}^{m} \lambda_i x_i x_i^H\right) = \sum\left(\prod_{t=1}^{n} \lambda_{i(t)}\right) \left|\det[x_{i(1)},\dots,x_{i(n)}]\right|^2. \quad (33)$$

It is an important fact, implicitly used in the next lemma, that the *coefficient* of each product $\lambda_{i(1)} \cdots \lambda_{i(n)}$ in (33) is nonnegative. Note also that (33) is nonnegative when all $\lambda_i \geq 0$.

LEMMA 3. *Let $n \geq 1$. For $i = 1, \ldots, n+1$, let $\lambda_i \in R$, $x_i \in C^n$, $y_i \in C^{n+1}$. If every subset of n of the vectors $x_1, \ldots, x_{n+1}$ is linearly independent, then the ratio*

$$\frac{\det\left(\sum_{i=1}^{n+1} \lambda_i y_i y_i^H\right)}{\det\left(\sum_{i=1}^{n+1} \lambda_i x_i x_i^H\right)} \tag{34}$$

attains its maximum over all $\lambda \in \Lambda$ for which the denominator is nonzero. A maximizing vector is

$$\lambda_i^* = \frac{|\det S_i|}{\sum_{t=1}^{n+1} |\det S_t|}, \qquad i = 1, \ldots, n+1, \tag{35}$$

where S_i denotes the $n \times n$ matrix $[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n+1}]$, $i = 1, \ldots, n+1$. The maximum value of (34) *is*

$$\left\{ \frac{|\det[y_1, \ldots, y_{n+1}]|}{|\det S_1| + \cdots + |\det S_{n+1}|} \right\}^2. \tag{36}$$

The vector λ^* is unique if and only if $\det[y_1, \cdots, y_{n+1}] \neq 0$.

Proof. From (33), since each $x_i \in C^n$ and $m = n+1$, then

$$\det\left(\sum_{i=1}^{n+1} \lambda_i x_i x_i^H\right) = \sum_{t=1}^{n+1} \left(\prod_{\substack{r=1 \\ r \neq t}}^{n+1} \lambda_r\right) |\det S_t|^2, \tag{37}$$

where S_t is as in the lemma statement. Because the vectors $\{x_i,\ i = 1, \ldots, n,\ i \neq t\}$ are linearly independent, $\det S_t \neq 0$. Consequently, the vector λ^* given by (35) is a well-defined element of Λ, and the determinant (37) does not

vanish for $\lambda = \lambda^*$. From (33), since each $y_t \in C^{n+1}$ and $m = n+1$, then

$$\det\left(\sum_{t=1}^{n+1} \lambda_t y_t y_t^H\right) = \left(\prod_{t=1}^{n} \lambda_t\right) |\det[y_1, \ldots, y_{n+1}]|^2. \tag{38}$$

If $\det[y_1, \ldots, y_{n+1}]$ vanishes then λ^* may as well be selected as the maximizing vector. Suppose now that $\det[y_1, \ldots, y_{n+1}] \neq 0$. The maximum ratio of the determinants in (34) is therefore positive. Hence, from (38), we may restrict attention to vectors $\lambda > 0$. The ratio (34) can now be written as $|\det[y_1, \ldots, y_{n+1}]|^2 / f(\lambda)$, where

$$f(\lambda) = \sum_{t=1}^{n+1} \frac{|\det S_t|^2}{\lambda_t}.$$

Maximizing the ratio is equivalent to minimizing $f(\lambda)$ subject to $\lambda \in \Lambda$. A minimizing vector is necessarily positive in this case, so we form the Lagrangian

$$\mathcal{L}(\lambda, \alpha) = f(\lambda) + \alpha(\lambda_1 + \cdots + \lambda_{n+1} - 1).$$

Stationary points of $\mathcal{L}$ satisfy

$$\frac{\partial}{\partial \lambda_t} \mathcal{L}(\lambda, \alpha) = -\frac{|\det S_t|^2}{\lambda_t^2} + \alpha = 0, \qquad t = 1, \ldots, n+1,$$

$$\frac{\partial}{\partial \alpha} \mathcal{L}(\lambda, \alpha) = \lambda_1 + \cdots + \lambda_{n+1} - 1 = 0.$$

These equations imply that $\alpha > 0$, that

$$\lambda_t^* = \frac{|\det S_t|}{\sqrt{\alpha}}, \qquad t = 1, \ldots, n+1,$$

that $\sqrt{\alpha} = |\det S_1| + \cdots + |\det S_{n+1}|$, and that λ_t^* is the only stationary point. It is obvious from the definition of $f(\lambda)$ that this stationary point is a minimizing point for f. The minimum value of f is

$$f(\lambda^*) = \sum_{t=1}^{n+1} \frac{|\det S_t|^2}{\lambda_t^*} = \sum_{t=1}^{n+1} \frac{|\det S_t|^2}{|\det S_t| / \sqrt{\alpha}} = \alpha,$$

so the maximum value of the ratio of determinants is as claimed. This completes the proof. ∎

The proof of de la Vallée Poussin's result (31) is now easy. From (7) and (8), since $A \in C^{(n+1)\times n}$, the matrix

$$L(\lambda) = \sum_{i=1}^{n+1} \lambda_i R_i^H R_i \in C^{n\times n},$$

where R_i denotes the ith *row* of the matrix A. Similarly, from (9)–(11),

$$M(\lambda) = \sum_{i=1}^{n+1} \lambda_i \tilde{R}_i^H \tilde{R}_i \in C^{(n+1)\times(n+1)},$$

where $\tilde{R}_i$ denotes the ith *row* of the augmented matrix $[A \;\; b] \in C^{(n+1)\times(n+1)}$. From Lemma 3,

$$\max_{\lambda\in\Lambda}\left[\frac{\det M(\lambda)}{\det L(\lambda)}\right]^{1/2} = \frac{\left|\det\left[\tilde{R}_1^H,\ldots,\tilde{R}_{n+1}^H\right]\right|}{\sum_{i=1}^{n+1}\left|\det\left[R_1^H,\ldots,R_{i-1}^H,R_{i+1}^H,\ldots,R_{n+1}^H\right]\right|}. \quad (39)$$

Since the determinant of the conjugate transpose of a matrix is the conjugate of its determinant, we have

$$\left|\det\left[R_1^H,\ldots,R_{i-1}^H,R_{i+1}^H,\ldots,R_{n+1}^H\right]\right| = |\det A_i|,$$

where A_i is defined as in (31). Expanding $\det[\tilde{R}_1^H,\ldots,\tilde{R}_{n+1}^H] = \det[A \;\; b]$ along its last column shows that the right hand side of (39) is identically the right hand side of (31). That the left hand sides are also equal follows from Theorem 3.

The unique $\lambda^* \in \Lambda$ for which the maximum (39) occurs is

$$\lambda_i^* = \frac{|\det A_i|}{\sum_{t=1}^{n+1}|\det A_t|}, \qquad i = 1,\ldots,n+1.$$

Consequently, the Chebyshev solution z^* of $Az = b$ is the (unique) solution of the λ^* weighted least squares problem for $Az = b$.

VI. CONCLUDING REMARKS

The general Chebyshev problem (2) can be posed as a convex optimization problem in the following way.

PROBLEM. Minimize

$$\{e: e \in R,\ z \in C^n\}$$

subject to

$$\left| b_i - \sum_{j=1}^{n} a_{ij} z_j \right|^2 \leqslant e, \qquad i = 1, \ldots, m.$$

Its convex dual can be developed in a manner similar to, but more general than, that pursued in [12] for Sylvester's problem. This approach is based on Wolfe's dual and the Kuhn-Tucker conditions. It yields Theorem 3 after somewhat tedious, but insightful, algebraic manipulations. In particular, the gradient equation of the Kuhn-Tucker conditions turns out to be the system of normal equations of the weighted least squares problem.

Theorem 2 can be extended to "weighted Chebyshev" solutions of $Az = b$. Let $w = (w_i) \in R^m$ be any nonnegative vector, and define

$$\phi_p(z, \lambda; w) = \left\{ \sum_{i=1}^{m} \lambda_i w_i \left| b_i - \sum_{j=1}^{n} a_{ij} z_j \right|^p \right\}^{1/p}, \qquad p \geqslant 1.$$

The proof of Theorem 1 can be trivially modified to show that $\phi_p(z, \lambda; w)$ has a saddle point (z_w^*, λ_w^*) on $C^n \times \Lambda$ for every nonnegative weight vector w. Consequently, as in the proof of Theorem 2, z_w^* is a w weighted Chebyshev solution of $Az = b$ and also a λ_w^* weighted least pth power solution of $Az = b$. Further generalization of Theorem 2 to nonlinear systems may be possible by replacing, in the definition (1) of ϕ_p, the functions $f_i(z) = |b_i - \Sigma a_{ij} z_j|$ with more general real valued convex functions of z.

REFERENCES

1 T. S. Motzkin and J. L. Walsh, Polynomials of best approximation on an interval, *Proc. Nat. Acad. Sci. U.S.A.* 45:1523–1528 (1959).

2 T. S. Motzkin and J. L. Walsh, Polynomials of best approximation on a real finite point set I, *Trans. Amer. Math. Soc.* 91:231–245 (1959).
3 C. L. Lawson, Contributions to the theory of linear least maximum approximations, Ph.D. Thesis, UCLA, 1961.
4 J. R. Rice, *The Approximation of Functions*, Vol. 2, Addison-Wesley, 1969.
5 J. R. Rice and K. H. Usow, The Lawson algorithm and extensions, *Math. Comp.* 22:118–127 (1968).
6 C. J. de la Vallée Poussin, Sur la méthode de l'approximation minimum, *Ann. Soc. Sci. Bruxelles, Seconde Partie, Memoires* 35:1–16 (1911).
7 T. J. Rivlin and H. S. Shapiro, A unified approach to certain problems of approximation and minimization, *J. Soc. Indust. Appl. Math.* 9:670–699 (1961).
8 A. K. Cline, Rate of convergence of Lawson's algorithm, *Math. Comp.* 26:167–176 (1972).
9 P. J. Davis, *Interpolation and Approximation*, Dover, 1975.
10 H. Kneser, Sur un théorème fondamental de la theorie des jeux, *C. R. Acad. Sci. Paris* 234:2418–2420 (1952).
11 E. G. Gol'stein, *Theory of Convex Programming*, Translations of Mathematical Monographs, Vol. 36, Amer. Math. Soc., Providence, R.I., 1972.
12 D. J. Elzinga and D. W. Hearn, The minimal covering sphere problem, *Management Sci.* 19:96–104 (1972).
13 F. R. Gantmacher, *The Theory of Matrices*, Vol. I, Chelsea, 1960.
14 M. Avriel, *Nonlinear Programming*, Prentice-Hall, 1976.

Received 17 August 1983; revised 6 March 1984

A Note On The Semi-Infinite Programming Approach To Complex Approximation

R. L. Streit and A. H. Nuttall

MATHEMATICS OF COMPUTATION
VOLUME 40, NUMBER 162
APRIL 1983, PAGES 599–605

A Note on the Semi-Infinite Programming Approach to Complex Approximation

By Roy L. Streit and Albert H. Nuttall

Abstract. Several observations are made about a recently proposed semi-infinite programming (SIP) method for computation of linear Chebyshev approximations to complex-valued functions. A particular discretization of the SIP problem is shown to be equivalent to replacing the usual absolute value of a complex number with related estimates, resulting in a class of quasi-norms on the complex number field $\mathbf{C}$, and consequently a class of quasi-norms on the space $C(Q)$ consisting of all continuous functions defined on $Q \subset \mathbf{C}$, Q compact. These quasi-norms on $C(Q)$ are estimates of the L_∞ norm on $C(Q)$ and are useful because the best approximation problem in each quasi-norm can be solved by solving (i) an ordinary linear program if Q is finite or (ii) a simplified SIP if Q is not finite.

Glashoff and Roleff [1] solve a semi-infinite program (SIP) which is shown to be equivalent to the linear approximation problem for functions in $C(Q)$, where $C(Q)$ is the space of complex-valued continuous functions on a compact (and not necessarily finite) subset Q of the complex plane $\mathbf{C}$ and is equipped with the uniform (L_∞) norm

$$\|f\|_\infty = \max_{z \in Q} |f(z)| . \tag{1}$$

Their method is a two-step procedure: the first step applies the usual simplex method of linear programming to solve a discrete approximation of the SIP; the second step uses the end result of the first step as the initial starting point in a Newton-Raphson iteration to solve a certain system of nonlinear algebraic equations whose solution (if feasible) is a solution to the linear approximation problem (Problem 1 below). The purpose of this note is to make some observations about the linear program of their discrete first step, which closely connects its solution with the solution of the approximation problem. A knowledge of the SIP definition and solution method is not needed to understand the results presented here. The interested reader is referred to [1], [2], [3], and to their bibliographies. We point out that Theorems 1 and 2 were first proved in [4], where a method identical to the first step of Glashoff-Roleff's procedure for finite Q was discovered independently of knowledge of [1] and of semi-infinite programming. Readers interested in practical examples and an engineering application of linear complex approximation are also referred to [4].

Received February 24, 1982.
1980 *Mathematics Subject Classification.* Primary 65D15, 65E05, 65K05; Secondary 30C30, 41A50, 30A82.

Let $h_1(z),\ldots,h_n(z)$ and $f(z)$ be given functions in $C(Q)$. For any set of complex parameters $a = \{a_1,\ldots,a_n\}$, define

$$L(a; z) = \sum_{k=1}^{n} a_k h_k(z). \tag{2}$$

Problem 1. Compute a set of complex parameters $a^* = \{a_1^*,\ldots,a_n^*\}$ such that, for all parameter sets a,

$$\| f - L(a^*; z)\|_\infty \leq \| f - L(a; z)\|_\infty. \tag{3}$$

We set

$$E_n(f) = \| f - L(a^*; z)\|_\infty. \tag{4}$$

Let $p \geq 2$ be a positive integer. Define the angles

$$\theta_j = \pi(j-1)/p; \qquad j = 1,2,\ldots,2p,$$

and let $S_p = \{\theta_j\}$. Define, for any complex number z,

$$|z|_p = \max_{1\leq j\leq 2p} \{\mathrm{Re}(z)\cos\theta_j + \mathrm{Im}(z)\sin\theta_j\}. \tag{5}$$

It may be readily verified that:

(i) $|z|_p \geq 0$ and $|z|_p = 0$ if and only if $z = 0$.

(ii) $|z + w|_p \leq |z|_p + |w|_p$ for all complex z and w.

(iii) Given α complex, $|\alpha z|_p = |\alpha| \cdot |z|_p$ for all z if and only if $\arg \alpha \in S_p$.

(iv) For α and α_n complex, $|-z|_p = |z|_p$, $\lim_{\alpha_n\to 0} |\alpha_n z|_p = 0$, and $\lim_{|z_n|_p\to 0} |\alpha z_n|_p = 0$.

Thus $|z|_p$ is not a norm on $\mathbf{C}$ because (iii) is not sufficiently strong; however, it is a quasi-norm because of (i), (ii), and (iv). See [5, pp. 30–32]. From the well-known identity

$$|z| = \max_{0\leq\theta<2\pi} \{\mathrm{Re}(z)\cos\theta + \mathrm{Im}(z)\sin\theta\}, \tag{6}$$

it follows that $|z|_p \leq |z|$. In addition it can be shown that

$$|z|_p \leq |z| \leq |z|_p \sec\left(\frac{\pi}{2p}\right), \qquad p \geq 2, \tag{7}$$

for all complex z. To see (7), it is helpful to visualize the set of all z in $\mathbf{C}$ such that $|z|_p = 1$ as an equilateral polygon of $2p$ sides whose inscribed circle is the unit circle $|z| = 1$.

It is easy to verify that

$$\| f \|_p \equiv \max_{z\in Q} |f(z)|_p \tag{8}$$

is a quasi-norm on $C(Q)$ for each integer $p \geq 2$. Further, from (7),

$$\| f \|_p \leq \| f \|_\infty \leq \| f \|_p \sec\left(\frac{\pi}{2p}\right). \tag{9}$$

We now define a new (partially) discretized version of Problem 1.

Problem 2. Fix $p \geq 2$. Compute a set of complex parameters $a^{**} = \{a_1^{**},\ldots,a_n^{**}\}$ such that, for all parameter sets a,

$$\| f - L(a^{**}; z)\|_p \leq \| f - L(a; z)\|_p. \tag{10}$$

We set

$$E_{np}(f) = \|f - L(a^{**}; z)\|_p. \tag{11}$$

THEOREM 1. $E_{np}(f) \leq E_n(f) \leq E_{np}(f)\sec(\frac{\pi}{2p})$.

Proof. We have

$$E_{np}(f) = \|f - L(a^*; z)\|_p \leq \|f - L(a^{**}; z)\|_p \leq \|f - L(a^*; z)\|_\infty = E_n(f)$$

$$\leq \|f - L(a^{**}; z)\|_\infty \leq \|f - L(a^{**}, z)\|_p \sec\left(\frac{\pi}{2p}\right) = E_{np}(f)\sec\left(\frac{\pi}{2p}\right).$$

THEOREM 2. $E_n(f) \leq \|f - L(a^{**}; z)\|_\infty \leq E_n(f)\sec(\frac{\pi}{2p})$.

Proof.

$$E_n(f) = \|f - L(a^*; z)\|_\infty \leq \|f - L(a^{**}; z)\|_\infty \leq \|f - L(a^{**}; z)\|_p \sec\left(\frac{\pi}{2p}\right)$$

$$\leq \|f - L(a^*; z)\|_p \sec\left(\frac{\pi}{2p}\right) \leq E_n(f)\sec\left(\frac{\pi}{2p}\right).$$

COROLLARY 1. $E_{np}(f) \leq \|f - L(a^{**}; z)\|_\infty \leq E_{np}(f)\sec(\frac{\pi}{2p})$.

COROLLARY 2. *For each* $p \geq 2$, $E_n(f) = 0$ *if and only if* $E_{np}(f) = 0$.

COROLLARY 3. *If* $E_n(f) \neq 0$,

$$0 \leq \frac{\|f - L(a^{**}; z)\|_\infty - E_n(f)}{E_n(f)} \leq \frac{\pi^2}{8p^2} + O\left(\frac{1}{p^4}\right), \qquad p \to \infty, \tag{12}$$

and the upper bound is independent of the compact set Q, n, f, *and the functions* $h_1, \ldots, h_n$.

Proof. The indicated ratio is bounded above by the constant $-1 + \sec(\frac{\pi}{2p})$.

It is not necessary that the domain of approximation Q be a subset of the complex plane C. All that is required is that f and $h_1, \ldots, h_n$ be defined on a common domain Q and that a solution to Problem 1 exists.

If the point set Q is not finite, then both Problems 1 and 2 can be readily transformed into linear SIP's with linear objective functions and infinitely many linear constraints and then can be solved in the manner of Glashoff and Roleff [1]. The difference is that, for Problem 1, there is one constraint for each element of the Cartesian product $S \times Q$, where $S \equiv \{\eta \text{ in } \mathbf{C}: |\eta| = 1\}$; whereas for Problem 2, there is only one constraint for each element of $S_p \times Q$, where $S_p \equiv \{\eta \text{ in } \mathbf{C}: \eta^{2p} = 1\}$. It can happen in certain applications that the bounds proved above show that Problem 2 is adequate for some fixed $p \geq 2$. The numerical solution procedures of Glashoff and Roleff may then be appropriately, and potentially significantly, simplified.

On the other hand, if Q is finite, Problem 2 becomes an ordinary linear program, although Problem 1 remains an SIP. The finite Q case is precisely the first step of the Glashoff and Roleff method for solving Problem 1. It is not hard to see that, for $Q = \{z, \ldots, z_m\} \subset \mathbf{C}$, Problem 2 may be reformulated as solving an overdetermined system of mp real linear algebraic equations in $2n$ real unknowns in the usual Chebyshev (l_∞) norm. Full details for setting up the linear equations can be found in [4]. (This formulation works for any choice of $T \equiv \{\theta_k\}$ provided only that $\theta_k \in T$ if

and only if $\theta_k + \pi \in T$.) This real system may be written in the following block-partitioned form:

$$\left[\begin{array}{c|c} R\cos\theta_1 + S\sin\theta_1 & R\sin\theta_1 - S\cos\theta_1 \\ R\cos\theta_2 + S\sin\theta_2 & R\sin\theta_2 - S\cos\theta_2 \\ \vdots & \vdots \\ R\cos\theta_p + S\sin\theta_p & R\sin\theta_p - S\cos\theta_p \end{array}\right]\left[\begin{array}{c} x \\ y \end{array}\right] = \left[\begin{array}{c} u\cos\theta_1 + v\sin\theta_1 \\ u\cos\theta_2 + v\sin\theta_2 \\ \vdots \\ u\cos\theta_p + v\sin\theta_p \end{array}\right]. \tag{13}$$

where we define

$$x = [\mathrm{Re}(a_k)] \in R^n, \qquad y = [\mathrm{Im}(a_k)] \in R^n,$$
$$u = [\mathrm{Re}(f(z_k))] \in R^m, \qquad v = [\mathrm{Im}(f(z_k))] \in R^m,$$

and the two $m \times n$ matrices

$$R = [r_{jk}] = [\mathrm{Re}(h_k(z_j))], \qquad S = [s_{jk}] = [\mathrm{Im}(h_k(z_j))].$$

Computer CPU time and storage requirements may present severe practical limitations on the numerical solution of (13) in certain problems of genuine interest. See Streit and Nuttall [4] for an antenna array example with $n = 44$, $p = 8$, and $m = 501$ which required 1262 simplex iterations and 179 minutes on the DEC VAX 11/780 to solve (13) using the general purpose algorithm [6]. If, however, the special structure of (13) is exploited, very significant reductions in both time and storage requirements are possible; see [7].

At least two situations might arise where the effective use of the structure of (13) in its solution would be important. First, the Glashoff-Roleff method for any given Q requires the solution (by Newton-Raphson or any other workable iterative method) of a nonlinear system of algebraic equations. If the initial point is not sufficiently good, then this procedure either does not converge or it converges to a nonfeasible (hence, incorrect) point. Since initial points are constructed by solving (13), it is conceivable that very large systems may have to be solved (even for small n) to get a sufficiently good initial point. The other reason for studying the special structure of (13) is simply that n may be very large to begin with. In the kind of applications mentioned in [4], it would not be at all unreasonable to find $n \geq 100$. Even for small p, the system (13) is then very large. Either case presents an interesting problem with a large 100% dense linear program having special structure, instead of the more typical situation of a large sparse linear program having relatively little special structure other than sparsity.

Solving the overdetermined system (13), while requiring nonnegative residuals, can have interesting geometrical interpretations. For example, take $p = 2$ so that $\theta_1 = 0$ and $\theta_2 = \pi/2$. Thus, the $2m$ components of the residual vector of (13) are precisely the real and imaginary parts of the complex error $e(z) = f(z) - L(a; z)$ evaluated at all m data points. Requiring nonnegative residuals means that we have forced the error curve $e(z)$ to lie entirely in the first quadrant of the complex plane. Furthermore, it is easy to see that we may force $e(z)$ to lie in any convex wedge-shaped sector $\mathfrak{W}$ of the complex plane by making appropriate alternative choices of the two

angles θ_1 and θ_2. Further exploration of this idea shows that upper and lower bounds for the error $W_n(f)$ defined by

$$W_n(f) \equiv \min_{a \in \mathbf{C}^n} \max_{z \in Q} |f(z) - L(a; z)|$$
$$\text{subject to: } f(z) - L(a; z) \in \mathfrak{W}, z \in Q,$$

can be obtained in terms of $W_{np}(f)$ defined by

$$W_{np}(f) \equiv \min_{a \in \mathbf{C}^n} \max_{z \in Q} |f(z) - L(a; z)|_p$$
$$\text{subject to: } f(z) - L(a; z) \in \mathfrak{W}, z \in Q.$$

This technique requires an appropriately modified set of angles $\theta_1, \ldots, \theta_{2p}$. A solution of $W_{np}(f)$ can then be found numerically by computing the l_∞ solution of an overdetermined system of the form (13) with the additional requirement of nonnegative residuals.

LEMMA. *Let Q be finite. The $2n$ columns of the coefficient matrix in* (13) *are linearly dependent (over the real number field) if and only if the n functions $\{h_1, \ldots, h_n\}$ are linearly dependent on Q (over the complex number field).*

Proof. There exist complex numbers $a_k = x_k + iy_k$, $1 \leq k \leq n$, not all zero, satisfying $\|\Sigma_k a_k h_k\|_\infty = 0$ if and only if $\|\Sigma_k a_k h_k\|_p = 0$. This latter equation is true if and only if

$$\max_{z \in Q} \left| \mathrm{Re}\left(\sum_{k=1}^{n} a_k h_k(z) \right) + i \,\mathrm{Im}\left(\sum_{k=1}^{n} a_k h_k(z) \right) \right|_p = 0,$$

which holds if and only if, for each $z_t \in Q$ and $\theta_j \in S_p$ $(1 \leq j \leq p)$,

$$\left| \left(\sum_{k=1}^{n} x_k \,\mathrm{Re}\, h_k(z_t) - y_k \,\mathrm{Im}\, h_k(z_t) \right) \cos\theta_j + \left(\sum_{k=1}^{n} x_k \,\mathrm{Im}\, h_k(z_t) + y_k \,\mathrm{Re}\, h_k(z_t) \right) \sin\theta_j \right| = 0.$$

Rearranging and using the notation of (13) gives

$$(R\cos\theta_j + S\sin\theta_j)x + (R\sin\theta_j - S\cos\theta_j)y = 0, \qquad j = 1, \ldots, p,$$

which means that the columns of the coefficient matrix in (13) are linearly dependent. This completes the proof.

THEOREM 3. *Let Q contain $m \leq 2n$ distinct points, let the functions $\{h_1, \ldots, h_n\}$ be linearly independent on Q, and let a^{**} satisfy* (10) *where $p \geq 2$. Then*

$$\|f - L(a^{**}, z)\|_\infty = E_{np}(f) \sec\left(\frac{\pi}{2p}\right). \tag{14}$$

Proof. If f is linearly dependent on $h_1, \ldots, h_n$, then $E_n(f) = 0$ and, from Corollary 2, $E_{np}(f) = 0$ and (14) is trivially true. Suppose then that f is linearly independent of $h_1, \ldots, h_n$. Let a^{**} satisfy (10). Then a^{**} is a Chebyshev (l_∞) solution of the system (13), and the maximum residual has magnitude $E_{np}(f) > 0$. From the

preceding Lemma, the rank of the coefficient matrix in (13) is $2n$. Hence there exists [8, p. 29] another solution $\hat{a}$ of (13) such that

$$\| f - L(\hat{a}; z) \|_p = E_{np}(f),$$

and a subset of at least $2n + 1$ of the mp equations (13) has residuals equal in magnitude to $E_{np}(f)$. (We cannot take $\hat{a} = a^{**}$ in general, because we have not assumed that the coefficient matrix in (13) satisfies the Haar condition for matrices.) Now these $2n + 1$ extremal equations must be distributed among the $m \leq 2n$ points of Q. Therefore, at least one point z in Q is assigned at least two equations.

Claim. No point in Q can be assigned more than two extremal equations. Note first that the residuals of the p equations in (13) corresponding to a given point z in Q are precisely

$$r_j \equiv A \cos \theta_j + B \sin \theta_j, \qquad j = 1, \ldots, p,$$

where A and B are the real and imaginary parts of $f(z) - L(\hat{a}; z)$, respectively. Let $K(z)$ denote the set of indices j of the extremal equations assigned to the point z. If $K(z)$ is not empty, then the equations

$$|A \cos \theta_j + B \sin \theta_j| = E_{np}(f), \qquad j \in K(z), \tag{15}$$

must hold simultaneously. Since $E_{np}(f) > 0$, it is clear that, if $K(z)$ contains more than two indices, the system (15) is inconsistent. This proves our claim.

Thus, let z be a point in Q which is assigned two extremal equations. Let $K(z) = \{j, k\}$ with $j \neq k$. Then the equations (15) imply

$$|A + iB| = E_{np}(f) \sec(\phi/2),$$

where ϕ is the smallest angle measured between the four angles $\{\theta_j, \theta_k, \theta_j + \pi, \theta_k + \pi\}$. Since Theorem 1 cannot be violated, we must have $\phi = \pi/p$. This concludes the proof.

If the coefficient matrix in (13) satisfies the Haar condition, then the norm (14) is attained for at least $t = \min\{2n + 1 - m, m\}$ distinct points z in Q. In this case, $\hat{a} = a^{**}$, so every point having two of the $2n + 1$ extremal equations has the residual (14). There must be at least t such points, considering the claim established in the proof of Theorem 3.

The first author is employed by the Naval Underwater Systems Center, New London Laboratory, New London, CT 06320. This paper was written during his stay as a Visiting Scholar in the Department of Operations Research, Stanford University, Stanford, CA 94305. His work was supported jointly by the Office of Naval Research Project RR014-07-01 and by The Independent Research Program of the Naval Underwater Systems Center.

Naval Underwater Systems Center
New London, Connecticut 06320

1. K. GLASHOFF & K. ROLEFF, "A new method for Chebyshev approximation of complex-valued functions," *Math. Comp.*, v. 36, 1981, pp. 233–239.

2. A. CHARNES, W. W. COOPER & K. O. KORTANEK, "Duality, Haar programs and finite sequence spaces," *Proc. Nat. Acad. Sci. U.S.A.*, v. 48, 1962, pp. 783–786.

3. S.-A. GUSTAFSON, "On semi-infinite programming in numerical analysis," in *Semi-Infinite Programming* (R. Hettich, Ed.), Lecture Notes in Control and Information Sciences, Vol. 15, Springer-Verlag, Berlin and New York, 1979, pp. 137–153.

4. R. L. STREIT & A. H. NUTTALL, "A general Chebyshev complex function approximation procedure and an application to beamforming," *J. Acoust. Soc. Amer.*, v. 72, 1982, pp. 181–190; Also NUSC Technical Report 6403, 26 February 1981. (Naval Underwater Systems Center, New London, Connecticut, U.S.A.)

5. K. YOSIDA, *Functional Analysis*, 2nd ed., Springer-Verlag, Berlin and New York, 1968.

6. I. BARRODALE & C. PHILLIPS, "Solution of an overdetermined system of linear equations in the Chebyshev norm," Algorithm 495, *ACM Trans. Math. Software*, v. 1, 1975, pp. 264–270.

7. R. L. STREIT, *Numerical Solutions of Systems of Complex Linear Equations with Constraints on the Unknowns*, Stanford University Department of Operations Research SOL Report. (To appear.)

8. G. A. WATSON, *Approximation Theory and Numerical Methods*, Wiley, New York, 1980.

Solution Of Systems Of Complex Linear Equations In The l_∞ Norm With Constraints On The Unknowns

R. L. Streit

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

SOLUTION OF SYSTEMS OF COMPLEX LINEAR EQUATIONS IN THE l_∞ NORM WITH CONSTRAINTS ON THE UNKNOWNS*

ROY L. STREIT†

Abstract. An algorithm for the numerical solution of general systems of complex linear equations in the l_∞, or Chebyshev, norm is presented. The objective is to find complex values for the unknowns so that the maximum magnitude residual of the system is a minimum. The unknowns are required to satisfy certain convex constraints; in particular, bounds on the magnitudes of the unknowns are imposed. In the algorithm presented here, this problem is replaced by a linear program generated in such a way that the relative error between its solution and a solution of the original problem can be estimated. The maximum relative error can easily be made as small as desired by selecting an appropriate linear program. Order of magnitude improvements in both computation time and computer storage requirements in an implementation of the simplex algorithm to this linear program are presented. Three numerical examples are included, one of which is a complex function approximation problem.

Key words. complex linear equations, Chebyshev solution, convex constraints, complex approximation, semi-infinite programming, l_∞ norm

1. Introduction. The numerical solution of general systems of complex linear equations in the l_∞, or Chebyshev, norm is a mathematical problem that arises in several applications. The objective is to find complex values for the unknowns so that the maximum magnitude residual of the system of equations is minimized. The unknowns are not allowed to assume any complex value whatever; instead, they are required to satisfy convex constraints of the form that can occur in the applications.

Let n, m, and r be positive integers. Let the matrices $A \in C^{n\times m}$, $B \in C^{n\times r}$, and the *row* vectors $f \in C^m$, $g \in C^r$, $a \in C^n$, $d \in R^n$, and $c \in R^r$ be given. The vector of unknowns, $z \in C^n$, is also taken to be a *row* vector. (Row instead of column vectors are used for notational convenience in § 2.) The problem is stated as follows.

Problem.

$$\min_{z\in C^n} \|zA - f\|_\infty \tag{1}$$

subject to:

$$|z - a| \leqq d, \tag{2}$$

$$|zB - g| \leqq c, \tag{3}$$

where the Chebyshev norm $\|\cdot\|_\infty$ of a vector is the maximum modulus of its components, and where the modulus $|\cdot|$ of a vector is defined to be the vector consisting of the moduli of its components. The simple constraints (2) are essential to the solution algorithm presented in this paper, but the more general constraints (3) are optional.

It is assumed that $c > 0$ and $d > 0$. Zero components of c and d are equivalent to equality constraints of the form $zH = e$. If H has rank $q \leqq n$, then q of the unknowns can be solved for explicitly in terms of the remaining unknowns and substituted out of the problem. The reduced problem has $n - q$ unknowns and the same mathematical form as (1)-(3).

In this paper the problem (1)-(3) is replaced by a discretized problem. The discretized problem is a linear program which is generated in such a way that the

* Received by the editors July 5, 1983, and in revised form October 1, 1984. This work was supported by the Office of Naval Research Project RR014-07-01 and by the Independent Research Program of the Naval Underwater Systems Center. This paper was written during the author's stay as a visiting scholar in the Department of Operations Research, Stanford University, Stanford, CA 94305.

† Naval Underwater Systems Center, New London, Connecticut 06320.

relative error between its solution and a solution of the problem (1)-(3) can be estimated without knowing the solution of either. Furthermore, the maximum relative error can easily be made as small as desired by selecting an appropriate discretized problem. See Theorem 1 below.

The starting point for the discretized problem is the following simple observation. Let u be a complex number whose real and imaginary parts are u^R and u^I, respectively. It is easy to show that

$$|u| = \max_{0 \leqq \theta < 2\pi} (u^R \cos\theta + u^I \sin\theta). \tag{4}$$

Let p be a positive integer, and let $D = \{\theta_1, \cdots, \theta_p\}$ be a subset of the interval $[0, 2\pi)$. The discretized absolute value is defined by

$$|u|_D = \max_{\theta \in D} (u^R \cos\theta + u^I \sin\theta). \tag{5}$$

Although the set D can be arbitrary, it is convenient to assume that D consists of the pth roots of unity, that is,

$$\theta_k = (k-1)2\pi/p, \qquad k = 1, 2, \cdots, p, \tag{6}$$

and to assume that $p = 2^K$, $K \geqq 2$. It follows that

$$|u|_D \leqq |u| \leqq |u|_D \sec(\pi/p). \tag{7}$$

No other choice of D can give a tighter upper bound in (7). The requirement that p be a power of 2 facilitates computational efficiencies in solving the optimization problem (5) and is discussed in § 2. With these two assumptions, a relative accuracy of 5 significant digits in (7) requires that $p \geqq 1024$. Other properties of the discretized absolute value are given in [13].

The discretized version of (1)-(3) is developed by first transforming it into an optimization problem and then replacing all absolute values with discretized absolute values. The discretized problem can be written in the following manner.

Discretized problem.

$$\min_{\varepsilon \in R, z \in C^n} \varepsilon \tag{8}$$

subject to:

$$|zA_j - f_j|_D \leqq \varepsilon, \qquad j = 1, \cdots, m, \tag{9}$$

$$|zB_j - g_j|_D \leqq c_j, \qquad j = 1, \cdots, r, \tag{10}$$

$$|z_j - a_j|_D \leqq d_j, \qquad j = 1, \cdots, n, \tag{11}$$

where A_j and B_j denote the jth columns of the matrices A and B, respectively. It is shown in § 2 that the discretized problem is a linear program in $2n+1$ unknowns with $(m+n+r)p$ inequalities. This linear program cannot be assumed to be sparse since the matrices A and B are completely dense in many applications.

The discretized problem is most easily solved by solving its dual. The revised simplex method applied in a straightforward manner to the dual problem requires $O((m+n+r)np)$ storage locations and $O((m+r)np)$ multiplications per simplex iteration. It is shown in this paper that the factor of p can be eliminated from these estimates by successfully exploiting the special structure of the dual. These economies leave unaltered the sequence of basic feasible solutions (vertices) which the simplex method generates enroute to the solution of the dual. Thus the impact of the parameter p is limited to its effect on the total number of simplex iterations required to reach the solution. As will be seen, p affects the number of columns in the dual constraints

and not the number of rows, so the growth of total computational effort as a function of p is not great.

A Fortran program for solving the discretized problem has been written and documented [11]. This program does not implement all of the economies which are possible because of practical considerations discussed in § 2. The program as written requires $O((m+r)n)+O(p)$ storage locations and $O((m+r)n)+O((m+n+r)\log_2 p)$ multiplications per simplex iteration. Also, for reasons stated in § 3, the solution of the discretized problem for large values of p is approached via smaller values of p. The discretized problem for $p=4$ is first solved and its solution used as an advanced start for the $p=8$ discretized problem. The program continues doubling p at each stage until a specified value is attained. This program is practical for modest values of m, n, and r for large values of p.

The following theorem proves that a solution of the discretized problem is an approximate solution of the original problem. It also proves that the maximum relative error in this approximate solution can be made as small as desired by appropriate choice of p. Similar results for the unconstrained problem are given in [12], [13], and [9].

THEOREM 1. *Let $z^* \in C^n$ solve problem* (1)-(3), *and let $\varepsilon^{**} \in R$ and $z^{**} \in C^n$ solve the discretized problem* (8)-(11). *Then*

$$\varepsilon^{**} \leqq \|z^*A - f\|_x \leqq \|z^{**}A - f\|_\infty \leqq \varepsilon^{**} \sec(\pi/p) \tag{12}$$

$$|z^{**}B - g| \leqq c \sec(\pi/p), \tag{13}$$

$$|z^{**} - a| \leqq d \sec(\pi/p). \tag{14}$$

Proof. Since $|z_j^{**} - a_j|_D \leqq d_j$ for each j, it follows from (7) that

$$|z_j^{**} - a_j| \leqq |z_j^{**} - a_j|_D \sec(\pi/p) \leqq d_j \sec(\pi/p).$$

This proves (14), and (13) is proved the same way. The following sequence of inequalities establishes (12):

$$\begin{aligned}
\varepsilon^{**} &= \max |z^{**}A_j - f_j|_D \\
&\leqq \max |z^*A_j - f_j|_D \\
&\leqq \max |z^*A_j - f_j| = \|z^*A - f\|_\infty \\
&\leqq \max |z^{**}A_j - f_j| = \|z^{**}A - f\|_\infty \\
&\leqq \max |z^{**}A_j - f_j|_D \sec(\pi/p) \\
&= \varepsilon^{**} \sec(\pi/p)
\end{aligned}$$

where the max in all cases is over $j = 1, \cdots, m$. This concludes the proof.

There is one hazard in replacing the original problem with the discretized problem. The constraints of the discretized problem have a larger feasible region than the original constraints, so it is possible that the discretized problem has solutions when the original problem is infeasible. The feasible region of the original problem is approximated more and more closely as p is increased, so the discretized problem ultimately fails to have a solution for sufficiently large p when the original problem is infeasible. If the original problem is in some sense "nearly" feasible, but in reality is infeasible, the discretized problem may possess solutions for very large values of p. Thus one may be deceived in certain problems. An alternative viewpoint is that any false solution obtained in this manner to infeasible problems actually represents a "reasonable"

solution to a poorly defined problem. Whether or not this view is sensible depends on the application. An example is given in § 5.

The problem (1)-(3) has a mathematically straightforward solution when all the quantities are real valued instead of complex. The real valued problem is exactly equivalent to a linear program in $n+1$ variables with $2(m+n+r)$ inequality constraints and can therefore be solved in a finite number of steps. The complex valued problem is less simple. Eliminating complex arithmetic by substituting in the real and imaginary parts of all complex quantities yields, after squaring the constraints, a mathematical programming problem in $2n+1$ variables having a linear objective function and $m+n+r$ quadratic constraints. No method is available for solving problems of this kind in a finite number of steps. Since it is a convex programming problem and the functions involved have easily obtained derivatives of all orders, many different algorithms are potentially applicable for its approximate solution. The only reference [14] known to the author which explicitly studies the constrained complex problem (1)-(3) uses a feasible directions method. At each step, a linear program is solved to determine the steepest feasible descent direction, a line search determines the step length, and special precautions are taken to prevent zigzagging, or jamming. A convergence proof is supplied.

The problem (1)-(3) can be viewed as a semiinfinite program (SIP). The SIP formulation of the unconstrained problem, that is, the problem consisting of only the objective function (1), has been studied elsewhere [12], [13], [4] in the context of complex function approximation and it is not difficult to extend that formulation to the constrained problem (1)-(3). None of these references, however, show that the special structure of the discretized problem can be used to significantly reduce the computational effort in its solution. Theorems 3, 4 and 5 of the next section are also new and are unique to the complex valued problem. The relationship between SIP and real valued approximation is presented in [3].

2. Solution of the discretized problem. An algorithm for solving the discretized problem for fixed p is discussed in this section. Attention is directed to special structures of the discretized problem which permit order of magnitude reductions in both storage requirements and multiplications per simplex iteration. Several useful theoretical results are interspersed.

It is first established that the discretized problem (8)-(11) is a linear program. Denote the real and imaginary parts of any quantity u by u^R and u^I, respectively, whether u be a number, a row or column vector, or a matrix. By definition (5),

$$|zA_j - f_j|_D = \max_{\theta \in D} [(zA_j - f_j)^R \cos\theta + (zA_j - f_j)^I \sin\theta], \tag{15}$$

so the m inequalities (9) are equivalent to the system of mp inequalities

$$(zA_j - f_j)^R \cos\theta + (zA_j - f_j)^I \sin\theta \leqq \varepsilon, \quad \theta \in D, \quad j = 1, \cdots, m. \tag{16}$$

Since

$$(zA_j - f_j)^R = z^R A_j^R - z^I A_j^I - f_j^R, \qquad (zA_j - f_j)^I = z^R A_j^I + z^I A_j^R - f_j^I,$$

it is convenient to write (16) in the form

$$[z^R z^I \varepsilon] \begin{bmatrix} A^R \cos\theta + A^I \sin\theta \\ A^R \sin\theta - A^I \cos\theta \\ -1_m \end{bmatrix} \leqq [f^R \cos\theta + f^I \sin\theta], \qquad \theta \in D, \tag{17}$$

where $1_m \in R^m$ is a row vector whose components all equal one. The inequalities (10)

and (11) are treated similarly, so the discretized problem is a linear program in $2n+1$ variables and $(m+n+r)p$ inequalities. The linear program can be written explicitly as follows.

Primal problem.

$$\min_{[z^R z^I \varepsilon]\in R^{2n+1}} [z^R \quad z^I \quad \varepsilon][0_n \quad 0_n \quad 1]^T \tag{18}$$

subject to: $\varepsilon \geqq 0$ and, for each $\theta \in D$,

$$[z^R \quad z^I \quad \varepsilon]\begin{bmatrix} A^R\cos\theta + A^I\sin\theta & B^R\cos\theta + B^I\sin\theta & I\cos\theta \\ A^R\sin\theta - A^I\cos\theta & B^R\sin\theta - B^I\cos\theta & I\sin\theta \\ -1_m & 0_r & 0_n \end{bmatrix} \tag{19}$$

$$\leqq [f^R\cos\theta + f^I\sin\theta \quad c + g^R\cos\theta + g^I\sin\theta \quad d + a^R\cos\theta + a^I\sin\theta],$$

where I denotes the $n\times n$ identity matrix and 0_k denotes a zero row (or column, depending on context) of length $k \geqq 1$.

The primal problem is solved by solving its dual using the revised simplex method. The simplex (Lagrange) multipliers for an optimal basic solution of the dual solve the primal, assuming the primal to be feasible. The dual can be written in one of the standard linear programming formats by explicitly adding a slack variable, denoted Q, which arises naturally in this problem.

Dual problem.

$$\min_{\substack{S\in R^{m\times p}, T\in R^{r\times p} \\ W\in R^{n\times p}, Q\in R}} \sum_{k=1}^{p} \begin{Bmatrix} (f^R\cos\theta_k + f^I\sin\theta_k)S_k \\ +(c + g^R\cos\theta_k + g^I\sin\theta_k)T_k \\ +(d + a^R\cos\theta_k + a^I\sin\theta_k)W_k \end{Bmatrix} \tag{20}$$

subject to: $S \geqq 0$, $T \geqq 0$, $W \geqq 0$, $Q \geqq 0$, and

(21)

$$\sum_{k=1}^{p} \begin{bmatrix} A^R\cos\theta_k + A^I\sin\theta_k & B^R\cos\theta_k + B^I\sin\theta_k & I\cos\theta_k \\ A^R\sin\theta_k - A^I\cos\theta_k & B^R\sin\theta_k - B^I\cos\theta_k & I\sin\theta_k \\ 1_m & 0_r & 0_n \end{bmatrix} \begin{bmatrix} S_k \\ T_k \\ W_k \end{bmatrix} + \begin{bmatrix} 0_n \\ 0_n \\ 1 \end{bmatrix} Q = \begin{bmatrix} 0_n \\ 0_n \\ 1 \end{bmatrix}.$$

An alternative statement of the dual is given at the end of this section.

The slack variable Q plays a special role, as seen in the next result.

THEOREM 2. *Let the matrices* $S^{**} \geqq 0$, $W^{**} \geqq 0$, $T^{**} \geqq 0$, *and the real number* $Q^{**} \geqq 0$ *denote an optimal basic feasible solution of the dual problem* (20)-(21). *If* $Q^{**} > 0$, *then the optimal value of the objective function in the primal problem* (18)-(19) *is zero.*

Proof. Let $[z^{**R} z^{**I} \varepsilon^{**}] \in R^{2n+1}$ denote the simplex multipliers of the optimal basic solution S^{**}, W^{**}, T^{**}, Q^{**}. Applying the complementary slackness theorem [8, p. 77], $Q^{**} > 0$ implies $\varepsilon^{**} = 0$ as claimed.

Except for the slack variable Q, every basic variable of the dual is uniquely identified by specifying the matrix to which it belongs together with its location (row and column number) in this matrix. The matrix names S, T, and W correspond to the inequality systems (9), (10), and (11), respectively. The row number of a basic variable identifies the particular constraint which gives rise to it. For example, all the dual variables in row q of matrix T are eliminated from the dual problem if the qth inequality in (10) is deleted from the discretized problem. Similarly, the column number of a basic variable identifies the angle in the set D to which it corresponds.

The revised simplex algorithm, as applied to the dual, is defined in general terms as follows:

Step 1. Determine an initial basic feasible solution of the dual problem.
Step 2. Compute the simplex multipliers corresponding to the current basic feasible solution.
Step 3. Determine the incoming variable by selecting the variable having the most negative reduced cost coefficient; terminate if all reduced cost coefficients are nonnegative—the primal problem is solved by the current simplex multipliers.
Step 4. Compute the column of the incoming variable in terms of the current basis.
Step 5. Determine the outgoing basic variable by a ratio test; terminate if the dual objective function is unbounded below—the primal problem is infeasible.
Step 6. Update the basis inverse and current basic feasible solution by pivoting, and return to Step 2.

The special structure of the dual problem has its strongest influence on Steps 1, 3, and 4. These effects are outlined next. More detailed aspects of the algorithm are postponed to § 4.

The dual problem is already in canonical form for initiating the second phase of the simplex algorithm. In other words, Step 1 is trivial because an identity matrix of order $2n+1$ can be assembled from the columns of the coefficient matrix of (21). One readily available column is the column corresponding to the slack variable Q. The remaining $2n$ columns correspond to dual variables which are the components of two particular W columns. From (6), $\theta_1 = 0$ so that $\cos\theta_1 = 1$ and $\sin\theta_1 = 0$. Hence one of the W columns can be taken to be W_1. Similarly, the other is $W_{1+p/4}$ since $\theta_{1+p/4} = \pi/2$. The initial basic feasible solution is therefore

$$W_1 = W_{1+p/4} = 0_m, \qquad Q = 1. \tag{22}$$

The simplex multipliers corresponding to (22) are derived in a special way later in this section.

The initial basic feasible solution (22) is highly degenerate. As discussed in [2], it is in problems of this general kind that cycling in the simplex algorithm is occasionally observed in practice. Such cycling was observed in an example given in this paper. However, a modification of the tie-breaking rule in the ratio test for the outgoing basic variable, together with "preferential treatment" of certain incoming variables, seems to avoid the difficulty. Further discussion of cycling in the dual problem is postponed to § 4.

The cost coefficients and the columns of any dual variable can be found by inspecting (20)-(21). They are given in a complex arithmetic format in Table 1. Explicitly computing and storing all $(m+n+r)p$ columns of the dual problem is unnecessary (and impractical) since the column of any dual variable can be constructed directly from the matrices A and B. Not counting the necessary sine and cosine, this requires only n complex multiplications and reduces the storage from $(2n+1)(m+n+r)p$ words to only $2n(m+n+r)$ words. The columns of the dual variables W_{jk} are merely columns of the identity matrix I, which need not be explicitly stored. Therefore the total storage necessary for constructing the column of any dual variable is only $2n(m+r)$ words. In practice, it is convenient to compute the cosines and sines once and for all to reduce the computational overhead. If this is done, as it is in [11], the storage requirements are $2n(m+r)+2p$.

TABLE 1
Dual variable cost coefficients and columns in complex format.

dual variable	cost coefficient	column, in R^{2n+1}
S_{jk}	$(f_j e^{-i\theta_k})^R$	$[(A_j e^{-i\theta_k})^R \quad -(A_j e^{-i\theta_k})^I \quad 1]^T$
T_{jk}	$(c_j + g_j e^{-i\theta_k})^R$	$[(B_j e^{-i\theta_k})^R \quad -(B_j e^{-i\theta_k})^I \quad 0]^T$
W_{jk}	$(d_j + a_j e^{-i\theta_k})^R$	$[(I_j e^{-i\theta_k})^R \quad -(I_j e^{-i\theta_k})^I \quad 0]^T$

An efficient method of computing the smallest reduced cost coefficient in Step 3 of the revised simplex algorithm is now discussed. This method is particularly interesting because the columns of the dual variables are not explicitly needed. The only data required are the original complex matrices A and B and the sines and cosines of the angles in D. Let λ be any real row vector of simplex multipliers for the dual problem; thus, λ is of length $2n+1$. The vector λ defines a complex row vector $z \in C^n$ and a real number ε by the identification

$$\lambda = [z^R \; z^I \; -\varepsilon] \in R^{2n+1}. \tag{23}$$

The reduced cost of the dual variable S_{jk} is the cost coefficient of S_{jk} minus the product of λ with the column of S_{jk}. Using (23) and Table 1 gives

$$\begin{aligned} C_S^{jk} &= (f_j e^{-i\theta_k})^R - [z^R \quad z^I \quad -\varepsilon][(A_j e^{-i\theta_k})^R \quad -(A_j e^{-i\theta_k})^I \quad 1]^T \\ &= \varepsilon - [(zA_j - f_j)\, e^{-i\theta_k}]^R, \end{aligned} \tag{24}$$

so the minimum reduced cost coefficient of the p variables in row j of S is

$$C_S^j = \min_{1 \le k \le p} C_S^{jk} = \varepsilon - |zA_j - f_j|_D, \qquad j = 1, 2, \cdots, m. \tag{25}$$

The smallest reduced cost coefficient of all the dual variables of S is then

$$C_S = \min_{1 \le j \le m} C_S^j = \varepsilon - \max_{1 \le j \le m} |zA_j - f_j|_D. \tag{26}$$

Similarly, the minimum reduced cost coefficients over all the dual variables of T and W are

$$C_T = \min_{1 \le j \le r} (c_j - |zB_j - g_j|_D) \tag{27}$$

and

$$C_W = \min_{1 \le j \le n} (d_j - |z_j - a_j|_D), \tag{28}$$

respectively. The smallest reduced cost of all the variables of the dual problem is $\min\{C_S, C_W, C_T\}$.

The smallest of the three quantities C_S, C_W, and C_T and the index j for which the minimum value is attained determine the row number and the correct matrix name of the incoming dual variable. The column number is determined by the angle $\theta_k \in D$ giving the largest projection (i.e., the discretized absolute value) at the minimal index j. The angle θ_k may not be unique because of possible ties in (5), so a tie-breaking rule called the minimal clockwise index (MCI) rule is used to determine unambiguously the incoming dual variable.

The MCI rule is defined for all $u \in C$. Let u_D be the set of those angles $\theta \in D$ for which the maximum in (5) is attained. There are three cases. First, if u_D has precisely one element, the MCI of u is defined to be the index of that element. Second, if u_D has precisely two elements, say θ_k and θ_j, and neither k or j equals p, then the MCI of u is defined to be $\min\{k, j\}$; on the other hand, if either $k = p$ or $j = p$, then the

MCI of u is taken to be p. Third, if u_D has more than two elements, then it must be that $u = 0$ and $u_D = D$, so the MCI of u is defined to be 1.

The computation of the discretized absolute value and corresponding MCI must be undertaken for $m+n+r$ complex numbers to compute (26)-(28) during each iteration of the simplex algorithm in Step 3. A brute force approach using the definition (5) requires $2p$ real multiplications for each complex number. Such an approach is inefficient and does not exploit the special form of the set D. For $p = 4$, it is clear that comparison tests alone suffice to solve this subproblem. For $p \geqq 8$, comparison tests and at most $2 \log_2 p - 5$ real multiplications are sufficient. To see this, first determine the quadrant of the complex plane in which the given number lies, and determine whether it lies above or below the 45° line bisecting the quadrant. This can be done using comparison tests only. Now that the "half-quadrant" in which the number lies is known, its projections onto the bounding rays of this half-quadrant can be computed in this special case using only one multiplication. If $p = 8$, a final comparison test ends the problem. If $p \geqq 16$, then the larger of the two projections reveals the "quarter-quadrant" in which the number must lie. The projection onto one of the bounding rays of this quarter-quadrant is already known; so it is only necessary to compute the projection onto the other bounding ray. This requires 2 real multiplications. If $p = 16$, a final comparison test ends the problem. If $p \geqq 32$, we continue as before. Counting the total possible number of steps proves the claim. This bisection method works because of the special form of the set D.

In principle the discretized absolute value and corresponding MCI can be found with computational effort independent of p. The argument (phase) of the given complex number can be computed, essentially as an inverse tangent, and from it the MCI can be found using comparison tests. Whenever the inverse tangent computation requires fewer than $2 \log_2 p - 5$ multiplications, it is more efficient than the bisection method described above. For $p \leqq 1024$ the bisection method is more efficient, and it is used in [11].

The number of real multiplications required to complete Step 3 using these methods is significantly less than that required in the usual approach. The straightforward method requires the computation of $(m+n+r)p - (2n+1)$ real inner products of length $2n$. Taking account of the simple form of the W columns gives a total of approximately

$$(2p-4)[n(m+r)+1]+4n(m+r-n-1/2) \tag{29}$$

real multiplications. The special methods discussed above require $m+n+r$ complex inner products of length n followed by the computation of the discretized absolute value and corresponding MCI for each inner product. Counting one complex multiplication as four real multiplications and considering the special form of the W columns gives a total of

$$4n(m+r)+(m+n+r)N_D \tag{30}$$

real multiplications, where N_D is the number of multiplications needed to compute one discretized absolute value and corresponding MCI. If the inverse tangent method is used, N_D is a constant independent of p. If the bisection method is used $N_D = 2 \log_2 p - 5$ for $p \geqq 8$, $N_D = 0$ for $p = 4$. The special methods are clearly better when $p \geqq 4$ and $m > n$. In the derivation of both (29) and (30) it was assumed that the last row of (21) in the dual problem was specially treated to avoid multiplications by 1 and 0.

The simplex multipliers $\lambda^{(0)} \in R^{2n+1}$ corresponding to the initial basic feasible solution (22) are now derived. Multiplying the initial basis inverse on the left by the

row vector containing the cost coefficients of the initial basic variables gives the row vector $\lambda^{(0)}$. The initial basis inverse is the identity matrix, the cost coefficients of the basic W variables (22) are given in Table 1, and the cost coefficient of the slack variable Q is 0. Consequently,

$$\lambda^{(0)} = [d+(a)^R \quad d+(-ia)^R \quad 0] = [d+a^R \quad d+a^I \quad 0] \in R^{2n+1}.$$

The definition (23) thus gives

$$z^{(0)} = a + d\,e^{i\pi/4}\sqrt{2}, \qquad \varepsilon^{(0)} = 0. \tag{31}$$

From the proof of Theorem 2 it can be seen that $\varepsilon = 0$ for as long as the slack variable remains in the basis and is positive.

The matrices S, W, and T are sparse because basic feasible solutions of the dual consist of only $2n+1$ nonnegative variables. Furthermore, no row of S, W, and T can contain more than two basic variables as the next theorem shows.

THEOREM 3. *No basic feasible solution of the dual problem* (20)-(21) *can have more than two basic variables in any one row of W or T. If a basic feasible solution of the dual problem has corresponding simplex multipliers with $\varepsilon > 0$, then S cannot have more than two basic variables in any one row.*

Proof. The first statement is proved for the matrix T; the proof for W is a special case. Consider the jth row of T. Suppose a basic feasible solution has three basic variables $T_{j\alpha}$, $T_{j\beta}$, and $T_{j\gamma}$ with α, β, and γ being distinct. Then the reduced costs for all three variables must be zero. A result analogous to (24) was used to prove (27); using it here gives

$$C_T^{jq} = 0 = c_j - [(zB_j - g_j)\,e^{-i\theta_q}]^R, \qquad q = \alpha, \beta, \gamma. \tag{32}$$

Thus the single complex number $zB_j - g_j$ has the same projection, namely c_j, in three distinct directions. This is impossible unless $zB_j - g_j = c_j = 0$, in contradiction to the assumption that $c_j > 0$. This establishes the first statement. The second statement is proved in the same way, by using (24) itself.

The following theorem relates knowledge of an optimal basis of the dual to "observable" quantities in the primal problem. The results of the theorem depend on the *names*, but not the actual numerical values, of the optimal dual basis. In addition it seems to indicate that the upper bound (12) in Theorem 1 will often be attained in practice.

THEOREM 4. *Let $\varepsilon^{**} \in R$ and $z^{**} \in C^n$ denote the simplex multipliers in the form* (23) *of an optimal basis for the dual problem* (20)-(21), *and suppose that $\varepsilon^{**} > 0$. If the jth row of one of the matrices S, W, or T contains two optimal basic variables in columns α and β with $p \geqq \alpha > \beta \geqq 1$, then either $\alpha - \beta = 1$ or $\alpha - \beta = p - 1$. If $\alpha - \beta = 1$, then*

$$z^{**}A_j - f_j = \varepsilon^{**} \sec(\pi/p) \exp[i(2\beta - 1)\pi/p], \tag{33}$$

or

$$z^{**}B_j - g_j = c_j \sec(\pi/p) \exp[i(2\beta - 1)\pi/p], \tag{34}$$

or

$$z_j^{**} - a_j = d_j \sec(\pi/p) \exp[i(2\beta - 1)\pi/p], \tag{35}$$

according to whether the jth row is a row of S, T, or W, respectively. Replacing β with p in (33)-(35) *gives the equations corresponding to the alternative case $\alpha - \beta = p - 1$.*

Proof. Only the S matrix case is treated since the other two cases are similar. The two basic variables involved are $S_{j\alpha}$ and $S_{j\beta}$. Assume that $p \geqq \alpha > \beta \geqq 1$. The reduced

costs $C_S^{j\alpha}$ and $C_S^{j\beta}$ must be 0, so (24) gives the two equations

$$\varepsilon^{**} = [(z^{**}A_j - f_j)\, e^{-i\theta_\alpha}]^R, \qquad \varepsilon^{**} = [(z^{**}A_j - f_j)\, e^{-i\theta_\beta}]^R. \tag{36}$$

Any complex number having identical projections in two directions is uniquely defined in both magnitude and phase. If θ_α differs from θ_β by π radians, the system (36) implies that $\varepsilon^{**} = 0$, contrary to assumption. Thus (36) implies that $z^{**}A_j - f_j = \varepsilon^{**} \sec(\phi/2) > 0$, where $\phi = \min\{\theta_\alpha - \theta_\beta, 2\pi - \theta_\alpha + \theta_\beta\}$. By Theorem 1, $\phi = \pi/p$, so that either $\theta_\alpha - \theta_\beta = \pi/p$ or $\theta_\alpha - \theta_\beta = \pi(2p-1)/p$. From (6), either $\alpha - \beta = 1$ or $\alpha - \beta = p - 1$. For $\alpha - \beta = 1$, solving the system (36) for the phase of $z^{**}A_j - f_j$ gives (33). The case $\alpha - \beta = p - 1$ is handled in the same way. This completes the proof.

Theorem 4 is useful in practice. Computed optimal dual solutions can be inspected to verify that optimal basic variables occurring in the same row are in fact "paired" in the manner described. If they are not, then numerical round-off errors have adversely affected the computed solution.

THEOREM 5. *Let ε^{**} and z^{**} be as in Theorem 4. If the jth row of one of the matrices S, W, or T contains an optimal basic variable in column α, $1 \leqq \alpha \leqq p$, then*

$$\varepsilon^{**} \leqq |z^{**}A_j - f_j| \leqq \varepsilon^{**} \sec \pi/p, \qquad \theta_\alpha - \pi/p \leqq \arg(z^{**}A_j - f_j) \leqq \theta_\alpha + \pi/p,$$

or

$$c_j \leqq |z^{**}B_j - g_j| \leqq c_j \sec \pi/p, \qquad \theta_\alpha - \pi/p \leqq \arg(z^{**}B_j - g_j) \leqq \theta_\alpha + \pi/p,$$

or

$$d_j \leqq |z_j^{**} - a_j| \leqq d_j \sec \pi/p, \qquad \theta_\alpha - \pi/p \leqq \arg(z_j^{**} - a_j) \leqq \theta_\alpha + \pi/p,$$

according to whether the jth row is a row of S, T, or W, respectively.

Proof. The proof is closely related to the method of proof of Theorem 4 and is not presented.

This section is concluded with a concise statement of the dual problem using complex arithmetic notation.

Dual problem: complex format.

$$\min_{\substack{S,T \\ W,Q}} [(fS + gT + aW)\, e^{-iD}]^R + \sum_{j=1}^{p} (CT_j + DW_j)$$

subject to: $S \geqq 0$, $T \geqq 0$, $W \geqq 0$, $Q \geqq 0$, and

$$(AS + BT + W)\, e^{-iD} = 0 \in C^n, \qquad Q + \sum_{j=1}^{m} \sum_{k=1}^{p} S_{jk} = 1.$$

We have used e^{-iD} to denote a complex column vector of length n whose kth component is $\exp(-i\theta_k)$; other notation is unchanged from (20)-(21).

3. Solution of the discretized problem for large p. One reason to solve large p discretized problems is that applications requiring 5 or more significant digits of relative accuracy in the optimal value of the objective function and/or in constraint satisfaction need to take $p \geqq 1024$; see Theorem 1. Another reason to solve large p problems is that their solutions furnish starting points for other methods which potentially provide greater accuracy. For instance, the problem (1)-(3) can be rewritten as a semiinfinite program, or SIP, and an interesting algorithm [5], [6] for solving a class of SIP's can be utilized. This method sets up an appropriate nonlinear system of algebraic equations which are solved using the Newton-Raphson method (or other iterative method); a

feasible solution of the nonlinear system is a solution of the SIP. The starting point of the Newton-Raphson iteration is taken to be the solution of a discretized problem. Large p discretized problems will have to be solved whenever very good starting points are needed to ensure convergence of the Newton-Raphson iteration.

There is, however, a practical limit to how large p may be taken in many problems. A discretized problem is numerically unstable for sufficiently large p if its optimal solution has, for every p, two basic dual variables in at least one row or S, W, or T. The columns of two such basic dual variables are less distinguishable numerically as p increases (see Table 1). Consequently, the basis matrix is more ill-conditioned for large p. Only those problems which never, for any p, have more than one optimal basic variable per row of S, W, or T can escape numerical ill-conditioning from this cause. Such problems seem to be uncommon.

The algorithm we suggest for solving the discretized problem for large p begins by solving the smallest dual problem, that is, the dual problem with $p=4$. Next, the $p=8$ dual problem is solved using the optimal basis for the $p=4$ dual to start the simplex algorithm. The $p=16$ dual is then solved starting at the optimal basis for the $p=8$ dual, and so forth. The algorithm is always well-defined because basic feasible dual solutions for a given p are also basic feasible dual solutions for all larger values of p because the sets D are nested for $p=4, 8, 16, 32, \cdots$. By doubling p at each stage beginning with $p=4$, this algorithm avoids bases associated with numerical instability from the discretization process until p becomes very large. Difficulties caused by ill-conditioning in the complex equations themselves cannot, of course, be avoided.

One advantage of this algorithm is that the optimal basis for each intermediate value of p can be easily inspected using Theorems 4 and 5 to determine if numerical round-off errors are significant. If sufficient error is present, the algorithm can be terminated early, or alternatively, the basis can be reinverted before continuing to the next value of p.

The primary drawback of the algorithm is that more simplex iterations are usually required to reach the final optimal dual basis by proceeding via smaller values of p than by solving the full dual problem all at once. This difficulty does not seem to be significant in practice and, in any event, can be partially overcome by skipping more rapidly through the available values of p. It is also possible to begin the algorithm with a larger initial value of p; that is, $p>4$.

Optimal solutions of the primal discretized problem converge only linearly with increasing p, while the optimal values ε^{**} converge quadratically. It would be useful to be able to extrapolate the primal solutions to obtain a better solution of the original problem (1)-(3). Richardson extrapolation (see, e.g., [7], [10]) worked very well for Examples 1-3 in § 5 for sufficiently large p, but failed in other problems. It is apparently successful only when (a) the row numbers of the optimal dual basic variables of the discretized problems identify the optimal active constraints of the original problem, and (b) the optimal values of the discretized problems equal the optimum value of the original problem. The first requirement can be met by taking p sufficiently large. The second requirement imposes more severe limitations on the practical utility of Richardson extrapolation.

4. Details of the revised simplex algorithm. Computer codes which treat complex matrices and vectors by separating them into their real and imaginary parts cause thrashing on virtual memory systems. Therefore the solution vector z of the primal problem is best stored as a complex vector and the simplex multipliers reordered to reflect the storage of z. The rows of the dual problem should also be reordered. The

computer code therefore visualizes the dual problem rows in the following order: $\{1, n+1, 2, n+2, \cdots, n-1, 2n-1, n, 2n, 2n+1\}$. These numbers denote the row numbers in the original system (21). The reordered system is much easier to work with in FORTRAN than the original system. With the rows of the dual problem in this order the reduced cost calculations can be coded in FORTRAN just as they are written in (26)-(28), provided the initial data of the problem are typed COMPLEX.

The name of a dual variable is a triplet $i/j/k$ of positive integers, where:

$i = 1, 2$, or 3 according to whether it is an S, W, or T variable,
j = constraint number, from (9)-(11),
k = projection number of the angle in the set D, $1 \leqq k \leqq p$.

The middle name j has different ranges depending on the value of the first name i. These triplets are ordered lexicographically.

The most negative reduced cost determines the entering basic variable in the simplex algorithm. Ties for the most negative reduced cost are broken by choosing the variable with the least lexicographically ordered name. Because the highly degenerate initial starting point (22) can cause cycling in the simplex algorithm, there is one exception to the least name rule in case of ties for the entering variable. As long as the slack variable Q remains in the basis, the only entering variables permitted are S variables with negative reduced costs. If S variables with negative reduced costs do not exist, then the entering variable is permitted to be a W or a T variable and ties are resolved by the least name rule. Thus, S variables are given priority for entering the basis only for as long as the slack Q is in the basis. Once Q is removed from the basis it never enters again, and exceptions to the tie breaking rule cease.

The outgoing basic variable is determined by the usual ratio test. If the least ratio is attained by more than one variable, the variable having the largest magnitude pivot leaves the basis. If more than one variable has the same magnitude pivot, then the variable with least index is selected. Because of degeneracy and cycling, there is one exception to this tie-breaking rule for the exiting variable. So long as the slack Q remains in the basis, only W variables are permitted to exit. This rule makes sense only when a W variable is involved in the tie; if no such W variable exists, the exception is not invoked. If more than one W variable is involved in the tie, then the one having the largest magnitude pivot with the least index is selected to exit. Just as for the entering variable, this exception ceases once the slack Q leaves the basis.

Cycling in the simplex algorithm has not been observed with these modifications to the usual tie breaking rules for entering and exiting variables. However, if these modifications are not used, cycling may well occur. Example 3 of § 5 below cycled (with a cycle of length 19) without these modifications. It is possible that cycling in this particular example is an artifact of finite precision arithmetic.

A nonzero tolerance is necessary when testing for the most negative reduced cost and for possible divisors in the ratio test. This number must not be too small and it must somehow be dependent on the scale of the problem data. The number used in [11] is the product of the unit round-off error of the host computer with the sum of the absolute values of the incoming column (i.e., its l_1 norm). This number is used for both reduced cost and pivot tolerance tests.

Besides the usual termination criteria in the simplex algorithm, the pricing method implicit in (26)-(28) yields a novel way to terminate the algorithm. The pricing method computes the most negative reduced cost by indirectly examining *all* reduced costs, not just the reduced costs of the nonbasic variables. Hence it can happen that the entering and the exiting variables are identical because of numerical round-off errors.

This event seems to signal that no further improvement in the solution is numerically possible. Solutions returned by terminating the algorithm whenever this "self-cycling" occurs appear to be satisfactory.

The Fortran code [11] was developed to test the methods described for solving the dual problem. It holds an explicit basis inverse and performs pivoting to update the inverse in each simplex iteration. Pivoting is known to be numerically unstable, but easily programmed. To forestall numerical difficulties the inverse is held in double precision, although a double precision inverse is not a satisfactory substitute for a numerically stable technique. Updating the QR factorization of the basis is preferable. Nonetheless the explicit inverse code gives good performance in many problems.

5. Examples. Example 1 is taken directly from [14, p. 249]. Let $n=2$, $m=5$, $r=2$, and define the matrices

$$A=\begin{bmatrix} 1 & 1 & 1 & .5 & 2 \\ -2 & 0 & 3 & -1 & 1 \end{bmatrix}, \qquad B=\begin{bmatrix} 2 & 2 \\ 2 & -4 \end{bmatrix}, \tag{37}$$

$$a=g=[0,0], \quad c=[\sqrt{2},\sqrt{2}], \quad d=[10,10],$$

$$f=[-1+i,\ -1+i,\ .5i,\ 0,\ -1+i].$$

Only the vector f is complex. The exact solution is $z_1=(-1+i)/2$, $z_2=0$, and $\varepsilon=\sqrt{2}/2$. The constraints of type (2) are not part of the original problem given in [14]. They have been added because their discretizations provide the initial dual basis.

Table 2 gives the solutions of the discretized primal problem for selected values of p. The optimal value of ε for $p=8$ is the optimal value of ε for all $p\geqq 8$. For $p\geqq 8$, the accuracy of the primal solutions depends solely upon the discretization errors since the optimal ε does not change. Table 3 gives the optimal basic solutions of the dual problems for the same values of p. The active constraints do not change for $p\geqq 8$, except for their θ names. Hence the active constraints at the optimum of the original nonlinear problem (1)-(3) have been identified. The fourth and fifth basic variables are "paired" in an obvious way; this behavior is explained by Theorem 4.

All optimal dual solutions are degenerate, or very nearly so. It turns out that the "degenerate parts" of the optimal dual solutions approximately doubled as p is doubled, especially when $p\geqq 64$. Assuming the trend continues indefinitely, the optimal dual solution will eventually look nondegenerate. This trend is probably an artifact of the numerical ill-conditioning inherent in the discretization process.

The conditions mentioned at the end of § 3 for success using Richardson extrapolation seem to be met for $p\geqq 8$. Since convergence of the z vectors is linear, multiply the $p=32$ vector by two and subtract the $p=16$ vector to get

$$[z_1^R, z_1^I, z_2^R, z_2^I]=[-.500964,\ .499036,\ .40\times 10^{-10},\ .36\times 10^{-10}].$$

One step of this extrapolation gives values nearly as accurate as the values corresponding to $p=2048$.

Numerical computations for this and the next two examples were performed on a DEC 10. It has a double precision unit round-off error of approximately 2×10^{-19}.

Example 1 can be made infeasible by adjoining one constraint of type (3). Replace B, g, and c in (37) with

$$\tilde{B}=\begin{bmatrix} 2 & 2 & 1 \\ 2 & -4 & 1 \end{bmatrix}, \qquad \tilde{g}=[0,0,7-4i], \qquad \tilde{c}=[\sqrt{2},\sqrt{2},\ 29/4].$$

The discretized primal problem is feasible for $p=4$ and 8; for $p\geqq 16$, it is infeasible.

TABLE 2
Solutions of the primal problem, Example 1.

p	4	8	16	32	2048
z_1^R	$-.588760$	$-.292893$	$-.400544$	$-.450754$	$-.499233$
z_1^I	.588760	.707107	.599456	.549246	.500767
z_2^R	$.59\times10^{-1}$	$-.82\times10^{-9}$	$-.28\times10^{-9}$	$-.12\times10^{-9}$	$-.15\times10^{-11}$
z_2^I	$-.59\times10^{-1}$	$-.82\times10^{-9}$	$-.12\times10^{-9}$	$-.78\times10^{-10}$	$-.15\times10^{-11}$
ϵ	.4112399	.7071068	.7071068	.7071068	.7071068
total iterations	10	14	17	20	38

TABLE 3
Solutions of the dual problem, Example 1.

p	4	8	16	32	2048
	1/2/1	1/1/8	1/1/15	1/1/29	1/1/1793
	1/2/4	1/2/1	1/2/16	1/2/30	1/2/1794
basis names	1/3/3	3/1/4	3/1/6	3/1/12	3/1/768
	3/2/2	3/2/3	3/2/6	3/2/12	3/2/768
	3/2/3	3/2/4	3/2/7	3/2/13	3/2/769
	.714286	1.000000	1.000000	1.000000	1.000000
	.000000	$.50\times10^{-19}$	$.66\times10^{-19}$	$.11\times10^{-18}$	$.58\times10^{-17}$
basis values	.285714	$-.50\times10^{-37}$	$.91\times10^{-19}$	$.24\times10^{-19}$	$.19\times10^{-17}$
	.000000	$.11\times10^{-18}$	$.22\times10^{-18}$	$.43\times10^{-18}$	$.27\times10^{-16}$
	.214286	.500000	.500000	.500000	.500000

This illustrates the remark made in § 1 that some discretized problems have feasible solutions when the original problem is actually infeasible.

Example 2 is the same as Example 1, except that constraints of type (2) are tightened so that they are active at the solution. Replace the vector d in (37) with $\tilde{d}=[.4,.4]$. The exact solution of this problem is $\epsilon=\sqrt{2}-.4$, $z_1=(-1+i)\sqrt{2}/5$, and

$$z_2^I=\frac{317(\sqrt{2}-1)-(431902-190320\sqrt{2})^{1/2}}{300-1200\sqrt{2}}\cong -.208846903,$$

$$z_2^R=-\frac{\sqrt{2}}{10}+\left[\frac{1}{8}-\left(z_2^I-\frac{\sqrt{2}}{10}\right)^2\right]^{1/2}\cong -.093336568.$$

Tables 4 and 5 give, respectively, the solutions of the primal and dual discretized problem for selected values of p. The obvious "pair" of basic variables in Table 5 is explained by Theorem 4. The conditions for success using Richardson extrapolation seem to be met for $p\geqq 32$. Extrapolation of the $p=32$ and $p=64$ vectors in Table 4 performed as in Example 1 gives

$$[z_1^R, z_1^I, z_2^R, z_2^I]=[-.282776,\ .282911,\ -.092665,\ -.209334],$$

which is comparable to the values corresponding to $p=2048$.

Example 3 is taken from [12] and is an unconstrained complex function approximation problem; that is, constraints of type (2)-(3) are absent. The 101 columns of the

TABLE 4
Solutions of the primal problem, Example 2.

p	4	8	16	32	64	2048
z_1^R	−.400000	−.345442	−.293794	−.310700	−.296738	−.283277
z_1^I	.400000	.220244	.271891	.254985	.268948	.282409
z_2^R	.153553	−.026274	−.076571	−.061857	−.077261	−.092805
z_2^I	−.153553	−.243431	−.217608	−.214390	−.211862	−.208960
ε	.600000	1.014214	1.014214	1.014214	1.014214	1.014214
total iterations	7	11	13	16	18	33

TABLE 5
Solutions of the dual problem, Example 2.

p	4	8	16	32	64	2048
basis names	1/2/1	1/2/8	1/2/15	1/2/29	1/2/57	1/2/1793
	1/2/4	1/3/6	1/3/12	1/3/22	1/3/43	1/3/1346
	2/1/3	2/1/4	2/1/7	2/1/13	2/1/25	2/1/769
	3/2/2	3/2/3	3/2/5	2/1/14	2/1/26	2/1/770
	3/2/3	3/2/4	3/2/6	3/2/10	3/2/19	3/2/558
basis values	1.	1.	1.	1.	1.	1.
	0.	0.	0.	0.	0.	0.
	1.	1.	1.	1.	1.	1.
	0.	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.	0.

matrix $A \in C^{3\times 101}$ are

$$A_j = [1 \quad \exp(i(j-1)\pi/400) \quad \exp(i2(j-1)\pi/400)]^T, \qquad j = 1, 2, \cdots, 101,$$

while the components of $f \in C^{101}$ are $f_j = \exp(i3(j-1)\pi/400)$, $j = 1, 2, \cdots, 101$. In other words, the complex valued function e^{i3x} is approximated by complex linear combinations of the three functions 1, e^{ix}, and e^{i2x} over 101 equispaced points on the x-interval $[0, \pi/4]$. Bounds of type (2) must be specified, so we take $a = [0, 0, 0]$, $d = [10, 10, 10]$. These constraints are not active at the optimal solution.

It can be verified that the exact solution of Example 3 is $z_1 = \alpha_1 \exp(i3\pi/8)$, $z_2 = \alpha_2 \exp(i5\pi/4)$, $z_3 = \alpha_3 \exp(i\pi/8)$, where

$$\alpha_1 = a \cong .96157056080646,$$

$$\alpha_2 = b - 2(b-a^2)/(1-a^2) \cong 2.8122548927058,$$

$$\alpha_3 = a(1-2b+a^2)/(1-a^2) \cong 2.8477590650226,$$

$$a = \lambda \cos(\pi/16) + (1-\lambda)\cos(\pi/8),$$

$$b = \lambda \cos(\pi/8) + (1-\lambda)\cos(\pi/4),$$

$$c = \lambda \cos(3\pi/16) + (1-\lambda)\cos(3\pi/8),$$

$$\lambda = \sin(\pi/8)/(\sin(\pi/16) + \sin(\pi/8)),$$

$$\varepsilon = (1 - c\alpha_1 + b\alpha_2 - a\alpha_3)^{1/2} \cong .014706309694449.$$

Tables 6 and 7 give, respectively, the solutions of the primal and dual discretized problems for selected values of p. The obvious "pairing" of the basic variables in Table 7 is explained by Theorem 4. Note also that the row numbers of the optimal dual basic variables are different for $p = 1024$ and $p = 64$ (probably because the dual does not have a unique solution). Nonetheless, Richardson extrapolation works when applied to the cases $p = 32$ and $p = 64$. As in the previous two examples, one extrapolation step gives

$$[z_1^R, z_1^I, z_2^R, z_2^I, z_3^R, z_3^I] = [.367954, .888319, -1.988481, -1.988481, 2.630930, 1.089767]$$

which, in turn, gives the values $\alpha_1 = .96151$, $\alpha_2 = 2.81214$, $\alpha_3 = 2.84770$. The case $p = 1024$ used directly gives the values $\alpha_1 = .96236$, $\alpha_2 = 2.81376$, $\alpha_3 = 2.84855$, which are clearly inferior to the extrapolated values.

TABLE 6
Solutions of the primal problem, Example 3.

p	8	16	32	64	1024
z_1^R	.378265	.377950	.377718	.372836	.368281
z_1^I	.913212	.912452	.911891	.900105	.889108
z_2^R	−2.026895	−2.024346	−2.022845	−2.005663	−1.989632
z_2^I	−2.026895	−2.024346	−2.022845	−2.005663	−1.989632
z_3^R	2.654494	2.654624	2.654502	2.642716	2.631719
z_3^I	1.099528	1.099581	1.099531	1.094649	1.090094
ϵ	.0141560	.0145244	.0147063	.0147063	.0147063
total iterations	20	25	33	36	48

TABLE 7
Solutions of the dual problem, Example 3.

p	8	16	32	64	1024
basis names	1/1/8	1/1/14	1/1/28	1/1/56	1/1/896
	1/25/4	1/1/15	1/1/29	1/1/57	1/1/897
	1/28/5	1/26/7	1/26/13	1/26/26	1/26/417
	1/74/8	1/27/8	1/26/14	1/26/27	1/76/993
	1/77/1	1/75/16	1/76/32	1/76/63	1/76/994
	1/101/5	1/76/1	1/101/17	1/101/33	1/101/513
	1/101/6	1/101/9	1/101/18	1/101/34	1/101/514
basis values	.163234	.004365	.000000	.000000	.000000
	.244029	.160548	.168829	.168829	.168829
	.091325	.170912	.000000	.000000	.331171
	.070677	.164573	.331171	.331171	.331170
	.263126	.173184	.331171	.331171	.000000
	.157491	.162060	.168829	.168829	.168829
	.010118	.164358	.000000	.000000	.000000

Another unconstrained complex function approximation problem in [12] is moderately large and completely dense. The motivating background and engineering application of this problem are fully discussed in [12]. The 501 columns of the matrix $A \in C^{44\times 501}$ are

$$A_j = [\exp(ik_1x_j) \quad \exp(ik_2x_j) \quad \cdots \quad \exp(ik_{44}x_j)]^T$$
$$-\exp(ik_{45}x_j)[1 \quad 1 \quad \cdots \quad 1]^T, \qquad j = 1, 2, \cdots, 501$$

where $1 = k_1 < k_2 < \cdots < k_{44} < k_{45} = 49$ are the distinct integers between 1 and 49, excluding the integers 7, 17, 21, and 29, and where $x_j = u_0 + (j-1)(1-u_0)/250$, $j = 1, 2, \cdots, 501$ with $u_0 = .0538117$. The components of $f \in C^{501}$ are $f_j = \exp(ik_{45}x_j)$, $j = 1, \cdots, 501$. This example lacks constraints of type (2)-(3). The discretized problem for $p = 16$ was solved on a DEC VAX 11/780 in 1350 simplex iterations. Total CPU time was 25 minutes and .7 million page faults were incurred. Only 80,000 words of storage were needed. In contrast, the algorithm proposed in [12] (which utilizes the algorithm [1] as a subroutine) solved this problem on the same VAX in 1270 simplex iterations, requiring 179 minutes of CPU time and incurring 11 million page faults. Over 360,000 words of storage were needed. The difference in the number of simplex iterations is explained as follows. The algorithm [12] solves the full problem for $p = 16$, while the algorithm developed in this paper solves the $p = 4$ problem and the $p = 8$ problem before solving the $p = 16$ problem. This indirect route to the full problem solution is less efficient in this example than solving the $p = 16$ problem immediately.

6. Concluding remarks. A solution of the discretized problem for sufficiently large p identifies the constraints active at a solution of the original problem (1)-(3). Deleting inactive constraints from the original problem yields an equality constrained nonlinear optimization problem. Lagrange's method gives rise to a nonlinear system of algebraic equations in the optimum value ϵ, the solution vector z, and the multipliers λ. Iterative methods for the solution of this system can be started from an initial point (ϵ, z, λ) provided by a discretized problem solution. Safeguarded Newton-Raphson iteration may be highly effective for solving this system, especially if advantage is taken of the system's special form (i.e., for λ given, the vector z can be found by solving a system of linear equations). A possible limitation of this approach is that very large values of p might be necessary in order to identify the right active set. The examples of the previous section, however, indicate that the optimal active set is found for relatively small values of p. Specifically, in Examples 1, 2, and 3, the correct active sets (determined from the optimal dual basis names in Tables 3, 5, and 7) first appear when p is 8, 8, and 32, respectively.

Certain kinds of domain and range constraints can be adjoined to the discretized problem (8)-(11) with only minor extension of the algorithm proposed here. Let the matrix $H \in C^{n \times q}$, and the row vectors $e \in C^q$, $\phi \in R^q$, and $h \in R^q$ be given. Then the constraints

$$((zH_j - e_j)\exp(-i\phi_j))^R \leqq h_j, \qquad j = 1, \cdots, q \tag{38}$$

are linear in z^R and z^I, and so can be added to the discretized problem. The constraints (10) and (11) are instances of (38); however, (38) can impose constraints not possible with (10) and (11). For instance, if $q = 1$, the constraint that the complex number $zH_1 - e_1$ must lie in the right half complex plane is equivalent to $((zH_1 - e_1)\exp(-i\pi))^R \leqq 0$. Furthermore, if $q \geqq 1$ and the columns H and e are identical to their first columns, then the number $zH_1 - e_1$ can be confined to any closed convex polygonal region (bounded or unbounded) in the complex plane by appropriate choices of ϕ, h, and q.

When complex function approximation on an arc or domain boundary in the complex plane gives rise to the problem (1)-(3), then an implicit natural ordering of the columns of the matrix A exists. The ordering is inherited from the ordering of the discrete points along the arc, and it makes possible clever strategies of both multiple and partial pricing which may significantly reduce overall computation time when m and n are large. Effective partial pricing schemes require far fewer evaluations of the

vector-matrix products zA_j in (26) without significantly increasing the total number of iterations. Effective multiple pricing schemes decrease the number of iterations by increasing the change in ϵ in each iteration. Both multiple and partial pricing can be implemented simultaneously.

One particularly interesting problem is complex function approximation on the mth roots of unity. When $m \geqq n$ and when m is a power of 2, the fast Fourier transform (FFT) algorithm can be used to compute the m products zA_j in $2m \log_2 m$ operations. The straightforward products zA_j require mn operations. Therefore, the FFT method is more efficient whenever $2 \log_2 m \leqq n \leqq m$.

It has been assumed throughout this paper that the unknown vector z must lie in C^n. In some applications it is necessary to restrict z to R^n, while still retaining complex matrices A and B in original problem (1)-(3). Setting $z^I = 0$ in the discretized problem is equivalent to eliminating n of the $2n+1$ rows of the dual problem constraints (21). The techniques developed for the dual problem simplify when applied to this modified problem. Consequently the modified dual problem is smaller and easier to solve. Examples and a Fortran program for this problem are given in [11].

REFERENCES

[1] I. BARRODALE AND C. PHILLIPS, *Solution of an overdetermined system of linear equations in the Chebyshev norm*, ACM Algorithm 495, ACM Trans. Math. Software, 1 (1975), pp. 264-270.

[2] S. I. GASS, *Comments on the possibility of cycling with the simplex method*, Letter to The Editor, Oper. Res., 27 (1979), pp. 848-852.

[3] K. GLASHOFF AND S.-A. GUSTAFSON, *Linear Optimization and Approximation*, Springer-Verlag, New York, 1983.

[4] K. GLASHOFF AND K. ROLEFF, *A new method for Chebyshev approximation of complex-valued functions*, Math. Comp., 36 (1981), pp. 233-239.

[5] S.-A. GUSTAFSON, *Nonlinear systems in semi-infinite programming*, in Numerical Solution of Nonlinear Algebraic Systems, G. B. Byrnes and C. A. Hall, eds., Academic Press, New York, 1973, pp. 63-99.

[6] S.-A. GUSTAFSON AND K. KORTANEK, *Numerical treatment of a class of semiinfinite programming problems*, Naval Research Log. Quart., 20 (1973), pp. 477-504.

[7] D. C. JOYCE, *Survey of extrapolation processes in numerical analysis*, SIAM Rev., 13 (1971), pp. 435-488.

[8] D. G. LUENBERGER, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973.

[9] G. OPFER, *Solving complex approximation problems by semiinfinite-finite optimization techniques: a study on convergence*, Numer. Math., 39 (1982), pp. 411-420.

[10] A. RALSTON, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1965.

[11] R. L. STREIT, *An algorithm for the solution of systems of complex linear equations in the l_∞ norm with constraints on the unknowns*, ACM Trans. Math. Software, 11, 3 (1985).

[12] R. L. STREIT AND A. H. NUTTALL, *Linear Chebyshev complex function approximation and an application to beamforming*, J. Acoust. Soc. Amer., 72(1) (1982), pp. 181-190. (Also in Naval Underwater Systems Center Report 6403, 26 February 1981.)

[13] R. L. STREIT AND A. H. NUTTALL, *A note on the semi-infinite programming approach to complex approximation*, Math. Comp., 40(1983), pp. 599-605.

[14] S. I. ZUKHOVITSKIY AND L. I. AVDEYEVA, *Linear and Convex Programming*, W. B. Saunders, Philadelphia, 1966. (Original Russian edition, Moscow, 1964.)

ALGORITHM 635:
An Algorithm For The Solution Of Systems Of Complex Linear Equations In The l_∞ Norm With Constraints On The Unknowns

R. L. Streit

ALGORITHM 635
An Algorithm for the Solution of Systems of Complex Linear Equations in the l_∞ Norm with Constraints on the Unknowns

ROY L. STREIT
Naval Underwater Systems Center

Categories and Subject Descriptors: G.1.2 [Numerical Analysis]: Approximation—*minimax approximation and algorithms*, G.1.3 [Numerical Analysis]: Numerical Linear Algebra—*linear systems (direct and iterative methods)*, G.1.6 [Numerical Analysis]: Optimization—*linear programming*

General Terms: Algorithms, Complex Systems

Additional Key Words and Phrases: complex linear equations, Chebyshev solution, complex approximation, constraints, semi-infinite programming

1. DESCRIPTION

The set of FORTRAN subroutines given here is an implementation of the algorithm [1] for computing $\mathcal{L}_\infty$, or Chebyshev, solutions to complex systems of equations with constraints on the unknowns.

Problem

$$\min_{\epsilon \in R, z \in C^m} \epsilon \tag{1}$$

subject to the approximation constraints

$$|zA_j - f_j| \le \epsilon, \qquad j = 1, \ldots, m, \tag{2}$$

general bound constraints

$$|zB_j - g_j| \le c_j, \qquad j = 1, \ldots, \ell, \tag{3}$$

and the simple bound constraints

$$|z_j - h_j| \le d_j, \qquad j = 1, \ldots, n. \tag{4}$$

It is assumed that the matrices $A \in C^{n \times m}$, $B \in C^{n \times \ell}$, and the row vectors $f \in C^m$,

This work was supported by the Office of Naval Research Project PR014-07-01 and by the Independent Research Program of the Naval Underwater Systems Center.
Author's address: Naval Underwater Systems Center, New London, CT 06320.
1985 ACM 0098-3500/85/0900-0242

$g \in C^\ell$, $h \in C^n$, $d \in R^n$, and $c \in R^\ell$ are all given. It is also assumed that $c_j > 0$ and $d_j > 0$ for all indices j. The vector of unknowns, z, is taken to be a row vector for reasons of notational convenience. Also, the jth columns of matrices A and B are denoted A_j and B_j, respectively. Note that m is allowed to be either greater than, less than, or equal to n. The simple bounds (4) are always assumed to be in the problem statement; however, the more general bounds (3) are allowed to be nonexistent. A different set of subroutines is given to solve this problem when the solution vector z is required to be real valued.

The algorithm is a very efficient implementation of the simplex method of linear programming applied to a discretized version of this problem.

Discretized Problem

$$\min_{\epsilon \in R, z \in C^n} \epsilon \tag{5}$$

subject to:

$$|zA_j - f_j|_D \le \epsilon, \qquad j = 1, \ldots, m, \tag{6}$$

$$|zB_j - g_j|_D \le c_j, \qquad j = 1, \ldots, \ell, \tag{7}$$

$$|z_j - h_j|_D \le d_j, \qquad j = 1, \ldots, n. \tag{8}$$

where, for any complex number $u \in C$, we defined the "discretized absolute value"

$$|u|_D = \max_{1 \le k \le p} |(\mathrm{Re}\ u) \cos \theta_k + (\mathrm{Im}\ u) \sin \theta_k|, \tag{9}$$

where $D = \{\theta_1, \ldots, \theta_p\}$ with

$$\theta_k = (k - 1)\ 2\pi/p, \qquad k = 1, 2, \ldots, p \tag{10}$$

and p is a positive integer controlling the degree of discretization. In this implementation of the algorithm, we have required that $p = 2^{**}\mathrm{LOGP}$, where LOGP is greater than or equal to one. From [1, Eq. (12)] we have

$$|u|_D \le |u| \le |u|_D \sec\left(\frac{\pi}{p}\right). \tag{11}$$

Thus, to attain a relative accuracy of five significant decimal digits (i.e., a relative error less than 0.5×10^{-5}) in the discretized absolute value requires that $p \ge 1024$. Other properties of the discretized absolute value are given in [1] and [2]. Also in [2] is a discussion of problem (1)–(2), without the constraints (3)–(4), as a semi-infinite program (SIP).

The error incurred by solving the Discretized Problem (5)–(8) instead of the original Problem (1)–(4) is given in [1, Theorem 2], which is repeated here for the sake of completeness.

THEOREM 2. *Let $\epsilon^* \in R$ and $z^* \in C^n$ solve Problem* (1)–(4), *and let $\epsilon^{**} \in R$ and $z^{**} \in C^n$ solve the Discretized Problem* (5)–(8). *Then*

$$\epsilon^{**} \le \epsilon^* \le \epsilon^{**} \sec\left(\frac{\pi}{p}\right), \tag{12}$$

and

$$|z^{**}A_j - f_j| \leq \epsilon^{**} \sec\left(\frac{\pi}{p}\right), \qquad j = 1, \ldots, m,$$

$$|z^{**}B_j - g_j| \leq c_j \sec\left(\frac{\pi}{p}\right), \qquad j = 1, \ldots, \ell,$$

$$|z_j^{**} - h_j| \leq d_j \sec\left(\frac{\pi}{p}\right), \qquad j = 1, \ldots, n.$$

It is clear from this result that the optimal ϵ in the Discretized Problem converges to the optimal ϵ in the original Problem quadratically as $p \to \infty$; however, the optimal z vectors need converge only linearly as $p \to \infty$. For a simple example, see [3].

The Discretized Problem is a dense linear program in $2n + 1$ real variables and $(m + n + \ell)p$ inequalities. It is solved numerically by solving its dual using the revised simplex method with explicitly held inverse. Even for modest values of m, n, ℓ, and p the dual is a very large linear program. Fortunately, it also has special structure which can be used very effectively to greatly reduce total computational effort. Instead of requiring the $(2n + 1)(m + n + \ell)p$ storage locations that would be necessary in a straightforward analysis, this implementation requires only $2n(m + \ell) + 2p$ locations. Moreover, a straightforward approach would require $O((m + \ell)np)$ real multiplications to determine the most negative reduced cost (and hence the entering basic variable) in each simplex iteration. This implementation requires only $O((m + \ell)n) + O((m + n + \ell)\log_2 p)$ real multiplications for the same purpose. In other words the discretization parameter p does not significantly affect the computational effort of a single simplex iteration. The size of p impacts primarily only the total number of iterations necessary to reach the optimal solution. The details are given in [1].

The revised simplex method with pivoting to update the basis inverse is known to be numerically unstable. Should a stable version become necessary, one can update the QR factors of the basis instead. The cost is a bit more computational effort in each simplex iteration. In practice, however, fewer iterations may be necessary with QR updating because of its stability. Consequently, total CPU time may not be significantly affected.

As was just described, the growth of computer storage as a function of p is precisely $2p$. This is quite satisfactory for all but the most demanding of applications. It is possible, however, to make the algorithm's storage requirements independent of p with slightly more computational effort per simplex iteration. Similarly, as a function of p, the multiplication count per simplex iteration grows as $\log_2 p$, but it is possible to alter the algorithm so that this growth is independent of p. Reprogramming the code given here to effect this modification should not be too difficult, if it ever becomes desirable to do so. Theoretically, then, the Discretized Problem can be solved by an algorithm whose storage requirements

and multiplication count per iteration is independent of the discretization parameter p; only the total number of iterations need remain dependent on p.

There are four subroutines in the package.

CAPROX	This is the main routine that implements the revised simplex method to solve the dual of the Discretized Problem.
CPAIRS	This subroutine prints the optimal basis names (if requested) of the Discretized Problem so that natural pairings (see [1]) in the optimal basis are immediately apparent.
CEND	This subroutine stores the best computed solution in the proper location prior to exit from CAPROX.
PABS	This is a subroutine that solves the optimization subproblem (9) for a given complex number u; that is, it computes the maximum in (9) and also the minimal clockwise angle θ_k for which this maximum occurs.

These four routines must be used together, but only CAPROX need be called by users of the algorithm. They have been tested on the VAX 11/780, and they have all been verified by the PFORT verifier [4] for portability.

In general, p cannot be taken equal to 2 without losing the desirable approximation properties of the Discretized Problem. In some special cases letting $p = 2$ will work, for example, when the problem is entirely real valued. From (10), for $p = 2$ we have $\theta_1 = 0$ and $\theta_2 = \pi$, so that $|u|_D = |u|$ when u is real—as it always will be in real valued problems. For this reason the implementation allows LOGP = 1 as a legal input. Most problems, however, will require that LOGP ≥ 2 for successful convergence to a desirable solution.

For those applications in which the solution vector z must be real valued even though the matrices A and B and the vectors h, g, and f are all complex, a different but highly similar set of FORTRAN subroutines has been provided. The four subroutines in this package are KAPROX, KPAIRS, KEND, and PABS. The routine PABS is the same one referred to above. All four routines must be used together, all have been tested on the VAX 11/780, and all have been verified by the PFORT verifier. These routines require less storage and are significantly faster than the more general problem allowing complex solution vectors.

2. EXAMPLE

The following numerical example (not included in [1]) is a constrained complex function approximation problem on a disconnected domain. We approximate the constant function 1 by polynomials of degree n, $n \geq 1$, which have zero constant terms. The domain is the union of a circle with center at $2i$ and radius 1 and a square with center at $-2i$ and sides of length 2. In addition, bounds are placed on the magnitudes of the coefficients of the approximating polynomial as well as on the magnitude of its first two derivatives evaluated at the point 1.

To pose this problem in the form (1)–(4), we must first discretize the domain boundary. Rather arbitrarily, we take 125 data points equispaced around the

circle and 160 data points equispaced around the square. This gives about the same spacing (as measured by arc length) on both the circle and the square. Explicitly, for precision's sake, the data points on the circle are

$$u_j = i\left[2 + \exp\left(\frac{(j-1)2\pi i}{125}\right)\right], \qquad j = 1, \ldots, 125,$$

and the data points on the square are

$$u_{125+j} = \left[1 - \frac{(j-1)}{20}\right] - i, \qquad j = 1, \ldots, 40,$$

$$u_{165+j} = -1 + \left[-1 - \frac{(j-1)}{20}\right]i, \qquad j = 1, \ldots, 40,$$

$$u_{205+j} = \left[-1 + \frac{(j-1)}{20}\right] - 3i, \qquad j = 1, \ldots, 40,$$

$$u_{245+j} = 1 + \left[-3 + \frac{(j-1)}{20}\right]i, \qquad j = 1, \ldots, 40.$$

Note that both the continuous domain and the discrete domain are symmetric about the imaginary axis.

The components of the solution vector z of (1)–(4) represent the coefficients of the approximating polynomial in this problem. Hence, the inequalities (2) are written simply

$$f_j = 1, \qquad j = 1, \ldots, 285,$$

$$A_j = \begin{bmatrix} u_j \\ u_j^2 \\ u_j^3 \\ \vdots \\ u_j^n \end{bmatrix}, \qquad j = 1, \ldots, 285.$$

The general bounds (3) express the derivative constraints by defining

$$B_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ n \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 0 \\ 2 \\ 6 \\ \vdots \\ n(n-1) \end{bmatrix}$$

$$g_1 = g_2 = 0, \qquad c_1 = c_2 = \tfrac{3}{4}.$$

The coefficient bounds are expressed by the inequalities (4); for illustrative purposes, we take

$$\left.\begin{aligned} h_j &= 0 \\ d_j &= \tfrac{1}{5} \end{aligned}\right\} \quad j = 1, \ldots, n.$$

Finally, we set $p = 1024$ and solve the Discretized Problems for ϵ^{**} and z^{**}. See Table I. For $1 \le n \le 4$, the problems might as well lack constraints of type (3) and (4) since these constraints are inactive at the optimal solution. For $5 \le n \le 8$, exactly one constraint of type (3) and one of type (4) are active at the solution. Optimal vectors z^{**} for $n = 4$ and $n = 8$ are given in Table II.

The discretized complex u-domain is symmetric about the imaginary axis. Hence, from [5, pp. 26–27], if general bounds (3) and simple bounds (4) are made so loose that they are never active at optimal solutions, this problem must have solution vectors z with alternately pure real and pure imaginary components. As Table II clearly shows, this effect need not occur when constraints of type (3) and (4) *are* active at optimality.

Under the additional requirement that the solution vector z be real valued, the same problem was solved using subroutine KAPROX. The results are summarized in Tables III and IV. We note that simple bounds (4) are not active for $1 \le n \le 8$ and the general bounds (3) are not active for $1 \le n \le 7$. In this problem, when the bounds (3) and (4) are not active at optimality, every real solution vector must have odd numbered components which are zero. (This follows easily from symmetry properties in the underlying u-domain.) Clearly, from Table IV, when a general bound (3) is active, the odd numbered components need not be zero.

The coefficients for $n = 2$ in Table IV deserves explanation. From Table III it is apparent that the error in the best (real) approximation is 1, so it must be the case that both coefficients are zero. So why is the second coefficient, z_2, equal to -0.001534? This is an effect of the discretization process and the fact that coefficients need to converge only linearly as p goes to infinity. Closer inspection of the problem solution shows that the active constraint of type (1) for $j = 126$ is a point where the upper bound (12) is attained. Since $u_{126} = 1 - i$, $f_{126} = 1$, $z_1 = 0$, $\epsilon^{**} = 1$, and $p = 1024$, we have

$$|zA_{126} - f_{126}| = \epsilon^{**} \sec \frac{\pi}{p} \tag{13}$$

or

$$|1 - z_2(1 - i)^2| = \sec \frac{\pi}{1024}.$$

Solving for z_2 gives

$$z_2 = \frac{-1}{2} \tan \frac{\pi}{1024} \doteq -0.00153398.$$

It would appear that z_2 satisfies (13) for all p; if so, it will never equal zero precisely and converge to zero only linearly.

Table I. Optimal Complex Solutions Using Subroutine CAPROX

Order	ϵ^{**} = optimal ϵ	Iterations	Time in seconds (VAX 11/780)
1	1.000000	3	1
2	0.973568	41	4
3	0.951666	84	8
4	0.905695	169	18
5	0.848662	219	22
6	0.848541	312	33
7	0.827420	708	87
8	0.825552	753	108

Table II. Optimal Complex Solution Vectors z Using Subroutine CAPROX

z^{**} component	$n = 4$	$n = 8$
1	.000000 + .087000 i	.040618 + .055840 i
2	−.195483 + .000000 i	−.199848 − .007808 i
3	.000000 + .026738 i	.020761 + .073041 i
4	−.014866 + .000000 i	−.020794 − .006209 i
5		.003313 + .014802 i
6		.001217 − .001490 i
7		.000184 + .000917 i
8		.000186 − .000105 i

Table III. Optimal Real Solutions Using Subroutine KAPROX

Order	ϵ^{**} = optimal ϵ	Iterations	Time in seconds (VAX 11/780)
1	1.000000	2	1
2	1.000000	12	2
3	1.000000	5	1
4	0.977278	49	6
5	0.977278	44	5
6	0.948679	74	9
7	0.948679	71	9
8	0.877424	157	18

Table IV. Optimal Real Solution Vectors z Using Subroutine KAPROX

z^{**} component	$n = 2$	$n = 4$	$n = 6$	$n = 8$
1	0.000000	0.000000	0.000000	0.073535
2	−0.001534	−0.105599	−0.155179	−0.193224
3		0.000000	0.000000	0.051395
4		−0.011453	−0.025029	−0.051975
5			0.000000	0.008079
6			−0.001251	−0.006946
7				0.000462
8				−0.000372

ACKNOWLEDGMENT

The author would like to thank Marvin J. Goldstein of the Naval Underwater Systems Center for making available software [6] that significantly eased the burden of producing readable FORTRAN code. Without the use of this software package, bringing the initial versions of subroutines CAPROX and KAPROX into compliance with certain ACM publication standards would have been extremely tedious.

REFERENCES

1. STREIT, R. L. Solution of systems of complex linear equations in the L_∞ norm with contraints on the unknowns. *SIAM J. Sci. Stat. Comp.*, to be published in Jan. 1986. (Also in Tech. Rep. SOL 83-3, Department of Operations Research, Stanford University, Stanford, CA, March, 1983.)
2. STREIT, R. L. AND NUTTALL, A. H. A Note on the semi-infinite programming approach to complex approximation. *Math. Comp. 40* (1983), 599–605.
3. OPFER, G. Solving complex approximation problems by semi-infinite optimization techniques. *Numer. Math. 39* (1982), 411–420.
4. RYDER, B. G. The PFORT verifier. *Softw. Pract. Exper. 4* (1974), 359–377
5. MEINARDUS, G. *Approximation of Functions: Theory and Numerical Methods*, Springer-Verlag, 1967.
6. GOLDSTEIN, M. J., AND LAWSON, J. R. JR. A new program aid in producing structured FORTRAN programs. NUSC Tech. Memo. 821162, Naval Underwater Systems Center, New London, CT, 9 Nov. 1982.

Received August 1983; accepted May 1985

Polynomial Iteration For Nonsymmetric Indefinite Linear Systems

H. C. Elman and R. L. Streit

Polynomial Iteration for Nonsymmetric Indefinite Linear Systems

Howard C. Elman
Yale University
Department of Computer Science
New Haven, CT

Roy L. Streit
Naval Underwater Systems Center
New London, CT

Abstract

We examine iterative methods for solving sparse nonsymmetric indefinite systems of linear equations. Methods considered include a new adaptive method based on polynomials that satisfy an optimality condition in the Chebyshev norm, the conjugate gradient-like method GMRES, and the conjugate gradient method applied to the normal equations. Numerical experiments on several non-self-adjoint indefinite elliptic boundary value problems suggest that none of these methods is dramatically superior to the others. Their performance in solving moderately difficult problems is satisfactory, but for harder problems their convergence is slow.

1. Introduction

In recent years there has been significant progress in the development of iterative methods for solving sparse real linear systems of the form

$$Au = b, \tag{1.1}$$

where A is a nonsymmetric matrix of order N. One key to this progress has been the derivation of polynomial based methods, i.e. methods whose m-th approximate solution iterate has the form

$$u_m = u_0 + q_{m-1}(A)r_0, \tag{1.2}$$

where u_0 is an initial guess for the solution, $r_0 = b - Au_0$, and q_{m-1} is a real polynomial of degree $m-1$. The residual $r_m = b - Au_m$ satisfies

$$r_m = [I - Aq_{m-1}(A)]r_0 = p_m(A)r_0, \tag{1.3}$$

where p_m is a real polynomial of degree m such that $p_m(0) = 1$. Applying any norm to (1.3) gives

$$\|r_m\| \leq \|p_m(A)\|\|r_0\|.$$

Moreover, if A is diagonalizable as $A = U\Lambda U^{-1}$, then

$$\|p_m(A)\| = \|Up_m(\Lambda)U^{-1}\| \leq \|U\|\|U^{-1}\| \max_{\lambda \in \sigma(A)} |p_m(\lambda)|,$$

The work presented in this paper was supported by the U. S. Office of Naval Research under contract N00014-82-K-0814, by the U. S. Army Research Office under contract DAAG-83-0177 and by the Naval Underwater Systems Center Independent Research Project A70209.

so that

$$\|r_m\| \leq \|U\|\|U^{-1}\| \max_{\lambda \in \sigma(A)} |p_m(\lambda)| \, \|r_0\|. \tag{1.4}$$

Thus any polynomial p_m that is sufficiently small on the eigenvalues of A is a good candidate for generating an iterative method.

The conjugate gradient and Chebyshev methods are well-known polynomial-based methods for solving symmetric positive-definite systems for which the residual polynomials $\{p_m\}$ have desirable optimality properties [8]. Generalizations of these techniques have been developed for solving both symmetric indefinite systems (see e.g. [3, 4, 17, 18]), and nonsymmetric systems with definite symmetric part $(A + A^T)/2$ (see e.g. [5, 8, 14] and references therein). In the latter case, all of the eigenvalues of A lie in either the right half or the left half of the complex plane. Sparse linear systems that both are nonsymmetric and have indefinite symmetric part arise in numerous settings. Examples include the discretization of the Helmholtz equations for modelling acoustic phenomena [1] and the discretization of the coupled partial differential equations arising in numerical semiconductor device simulation [12]. Gradient methods that have been proposed as solvers for such problems include the conjugate gradient method applied to the normal equations (CGN) [9], the biconjugate gradient method [7], the restarted generalized minimum residual method (GMRES) [20], and new methods presented in [11, 26]. Smolarski and Saylor [22] and Saad [19] have proposed adaptive polynomial iteration methods of the form (1.2) using polynomials that are optimal with respect a weighted least squares norm. In this paper, we introduce a polynomial-based method, PSUP, that computes a polynomial that is nearly optimal with respect to the Chebyshev norm on a region containing the eigenvalue estimates and then uses this polynomial in (1.2). We compare its performance with the two gradient methods CGN and GMRES.

In Section 2, we give a brief description of the gradient methods CGN and GMRES. In Section 3, we describe the new PSUP method and several heuristics developed to improve its performance. In Section 4, we describe numerical experiments in which these three methods are used to solve some non-self-adjoint indefinite elliptic problems, and in Section 5 we draw conclusions based on the numerical tests.

2. Gradient Methods

In this section we briefly review two conjugate gradient-like methods for solving nonsymmetric indefinite systems. The conjugate gradient method [9] is applicable only to symmetric positive definite linear systems. For nonsymmetric systems, it can be used to solve the normal equations $A^T Ax = A^T b$. The scaled residuals $\{A^T r_m\}$ satisfy

$$A^T r_m = p_m(A^T A) A^T r_0,$$

where p_m is the unique polynomial of degree m such that $p_m(0) = 1$ and $\|r_m\|_2$ is minimum. As is well known, the condition number of $A^T A$ is the square of that of A. Moreover, the standard implementation of CGN requires two matrix-vector products at each iteration, one by A and one by A^T, plus 5N additional operations. The storage requirement is 4N words. The dependence of CGN on $A^T A$ has led to efforts to find alternatives that are more rapidly convergent and less expensive per step. For nonsymmetric systems with positive definite symmetric part, several methods have been shown to be superior to CGN [5].

GMRES is a method proposed for solving nonsymmetric indefinite systems that avoids the use of the normal equations [20]. Given an initial guess, u_0, for the solution, with residual r_0, this method generates an orthogonal basis $\{v_1, \ldots, v_m\}$ for the Krylov space

$$K_m = \text{span}\{r_0, Ar_0, \ldots, A^{m-1} r_0\}$$

using Arnoldi's method. Let $v_1 = r_0/\|r_0\|_2$. The Arnoldi process computes for $j = 1, \ldots, m$

$$h_{ij} = (Av_j, v_i), \qquad i = 1, \ldots, j,$$
$$\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i,$$
$$h_{j+1,j} = \|\hat{v}_{j+1}\|_2,$$
$$v_{j+1} = \hat{v}_{j+1}/h_{j+1,i}.$$

GMRES then computes an approximate solution

$$u_m = u_0 + \sum_{j=1}^{m} \alpha_j v_j, \tag{2.1}$$

where the scalars $\{\alpha_j\}_{j=1}^m$ are chosen so that $\|r_m\|_2$ is minimum. These scalars can be computed by solving the upper Hessenberg least squares problem

$$\min_{\alpha} \left\| \|r_0\|_2 e_1 - \hat{H}_m \alpha \right\|_2,$$

where $e_1 = (1, 0, \ldots, 0)^T \in \mathbf{R}^{m+1}$ and $\hat{H}_m$ is the Hessenberg matrix of size $(m+1) \times m$ whose (i,j)-entry is h_{ij} [20]. By the choice of basis and the minimization property, $r_m = p_m(A) r_0$ where p_m is the real polynomial of degree m such that $p_m(0) = 1$ and p_m is optimal with respect to the residual norm $\|r_m\|_2$ (c.f. [8] for other formulations of this optimal iteration).

In a practical implementation, the dimension m of the Krylov space is fixed, and the GMRES iteration is restarted with u_m in place of u_0. This is the GMRES(m) method. Defining one "step" to be the average of the m-fold iteration divided by m, the cost per step is $(m + 3 + 1/m)N$ operations plus one matrix-vector product. It requires $(m+2)N$ words of storage.

We remark that the Arnoldi process was originally developed as a technique for computing eigenvalues [27]. Let V_m denote the matrix whose columns are the m vectors generated by the Arnoldi step in GMRES(m), and let H_m denote the square upper Hessenberg matrix consisting of the first m rows of $\hat{H}_m$. Then V_m is an orthonormal matrix of order $N \times m$ that satisfies

$$V_m^T A V_m = H_m. \tag{2.2}$$

Relation (2.2) resembles a similarity transformation, and Arnoldi's method consists of using the eigenvalues of H_m as estimates for (some of) the eigenvalues of A. Suppose $A = U \Lambda U^{-1}$ for diagonal Λ and r_0 is dominated by m eigenvectors $\{u_j\}_{j=1}^m$, with corresponding eigenvalues $\{\lambda_j\}_{j=1}^m$. Then the residual after m GMRES steps satisfies [6]

$$\|r_m\|_2 \leq \|U\|_2 \|U^{-1}\|_2 \; c_m \; \|e\|_2$$

where

$$c_m = \max_{k>m} \prod_{j=1}^{m} |\lambda_k - \lambda_j| / |\lambda_j|$$

and e is orthogonal to $\{u_j\}_{j=1}^m$. Loosely speaking, GMRES(m) damps out from the residual the eigenvectors whose eigenvalues are computed by Arnoldi's method.

3. The PSUP Method

The gradient methods just described compute iterates and residuals that satisfy (1.2) and (1.3) (for CGN, with respect to $A^T A$) in which the polynomials are built up recursively

without explicit computation of their coefficients. In this section, we describe an alternative iteration that computes explicitly the coefficients of a polynomial $q_{m-1}(z)$ for which $p_m(z) = 1 - zq_{m-1}(z)$ is small on the spectrum $\sigma(A)$. In the following, we will refer to the polynomial $q_{m-1}(z)$ of (1.2) as the "iteration polynomial" and to the polynomial $p_m(z) = 1 - zq_{m-1}(z)$ of (1.3) as the "residual polynomial."

Suppose a compact region $D \subset \mathbf{C}$ contains $\sigma(A)$. Let p_m be a polynomial of degree m that satisfies

$$p_m(0) = 1, \qquad \|p_m\| = \max_{z \in D} |p_m(z)| = \epsilon < 1.$$

As is evident from (1.4), an iteration having p_m as its residual polynomial will result in a decrease of the residual norm if ϵ is small enough. The best possible iteration polynomial with respect to this norm (the Chebyshev norm) is the solution to the minimax problem

$$\epsilon = \min_{q_{m-1}} \max_{z \in D} |1 - zq_{m-1}(z)|. \tag{3.1}$$

Let $q_{m-1}(z) = \sum_{j=0}^{m-1} a_j z^j$. The solution to (3.1) is also the Chebyshev solution to the infinite system of equations

$$\sum_{j=0}^{m-1} z^{j+1} a_j = 1, \qquad z \in \partial D. \tag{3.2}$$

Only the boundary ∂D need be considered because of the maximum modulus principle.

The PSUP method uses an iteration polynomial obtained from an approximate solution to (3.1). We briefly summarize the technique used; details can be found in [24]. First, (3.2) is replaced by a finite dimensional problem

$$\sum_{j=0}^{m-1} z^{j+1} a_j = 1, \qquad z \in \partial D_M, \tag{3.3}$$

where ∂D_M is a finite subset of ∂D containing M points, $M > m$. Equation (3.3) is an overdetermined system of M equations in the m unknowns $\{a_j\}_{j=0}^{m-1}$. The Chebyshev problem for (3.3) is given by

$$\min_{\{a_j\}} \max_{z \in \partial D_M} \left| \sum_{j=0}^{m-1} z^{j+1} a_j - 1 \right|. \tag{3.4}$$

Second, equation (3.4) is solved approximately using a semi-infinite linear programming approach to complex approximation, which is based on the identity $|w| = \max_{0 \le \theta < 2\pi} Re(we^{-i\theta})$, $w \in \mathbf{C}$. Let $\Theta = \{\theta_1, \ldots, \theta_p\} \subset [0, 2\pi)$, and define the *discretized absolute value*

$$|w|_\Theta = \max_{\theta \in \Theta} Re(we^{-i\theta}).$$

Consider the discretized problem

$$\min_{\{a_j\}} \max_{z \in \partial D_M} \left| \sum_{j=0}^{m-1} z^{j+1} a_j - 1 \right|_\Theta, \tag{3.5}$$

where the absolute value in (3.4) is replaced by the discretized absolute value. This gives rise to a linear program for $\{a_j\}_{j=0}^{m-1}$. Let ϵ^* denote the minimax value of $|\sum_{j=0}^{m-1} z^{j+1} a_j - 1|$ at the solution to (3.4), and let ϵ_p^* denote the minimax value for (3.5). It can be shown that

$$|w|_\Theta \le |w| \le |w|_\Theta \sec(\alpha/2)$$

for all $w \in \mathbf{C}$, and consequently that

$$\epsilon_p^* \le \epsilon^* \le \epsilon_p^* \sec(\alpha/2),$$

where α is the smallest difference (mod 2π) between two neighboring angles in Θ. The upper bounds are sharpest for given p when Θ consists of the p-th roots of unity, so that $\alpha = 2\pi/p$. We use this choice of Θ in the following, with $p = 256$ so that $\sec(\alpha/2) = 1.000075$.

The dual of the LP (3.5) can be written in the form

$$\min_{S \in \mathbf{R}^{M \times p},\ Q \in \mathbf{R}} Re[e_M^T S e^{-i\Theta}]$$

$$\text{subject to: } S \ge 0, \quad Q \ge 0, \quad Z^T S e^{-i\Theta} = 0 \in \mathbf{C}^m$$

$$\text{and} \quad Q + \sum_{j=1}^{M} \sum_{k=1}^{p} S_{jk} = 1,$$

where $e_M \in \mathbf{C}^M$ is the vector whose components are all 1, $Z \in \mathbf{C}^{M \times m}$ is the coefficient matrix of (3.4), and $e^{-i\Theta} \in \mathbf{C}^p$ denotes the vector whose j-th component is $e^{-i\theta_j}$. Q is a slack variable which must be 0 if $\epsilon_p^* > 0$. A straightforward application of the simplex method to the dual requires $O(Mmp)$ multiplications per simplex iteration and $O(Mmp)$ storage locations. In [24], it is shown that the factor p can be eliminated from these estimates by exploiting the special structure of the dual. These economies leave unaltered the sequence of basic feasible solutions that the simplex method generates en route to the solution. Moreover, they simplify further if the coefficients $\{a_j\}$ are required to be real. In practice the number of simplex iterations has been observed to be $O(m)$ so that the computational effort to compute $\{a_j\}$ using the algorithm in [23] is $O(Mm^2)$. In the experiments discussed below, both M and m are significantly smaller than the order N of the linear system so that construction of the coefficients of the iteration polynomial is a low order cost of the solution process.

Given u_0 and r_0, the basic PSUP iteration consists of repeated application of the iteration polynomial q_{m-1}, as follows:

Algorithm 1: The PSUP iteration.
For $k = 1, 2, \ldots$ Do
$u_{km} = u_{(k-1)m} + q_{m-1}(A) r_{(k-1)m}$
$r_{km} = b - A u_{(k-1)m}$.

The actual computation $w \leftarrow q_{m-1}(A) r$ is performed using Horner's rule:

$w \leftarrow a_{m-1} r$
For $j = 1$ to $m - 1$ Do
$v \leftarrow Aw$
$w \leftarrow a_{m-1-j} r + v$.

The m-fold PSUP iteration requires m matrix-vector products and m scalar vector products, so that the "average" cost is one matrix-vector product and one scalar-vector product. PSUP requires $4N$ storage, for u, r, v and w.

In practice, the PSUP iteration needs estimates of the eigenvalues of A in order to obtain the set D. Several adaptive techniques have been developed for combining an eigenvalue estimation procedure with polynomial iteration [6, 13, 19]. We will use the hybrid technique developed in [6, 19], which uses Arnoldi's method for eigenvalue estimates.

First, the Arnoldi process is used to compute some number k_i of eigenvalue estimates prior to execution of the PSUP iteration. Given these estimates, a set D is constructed that contains them, from which the PSUP iteration polynomial q_{m-1} is computed. (We discuss our choice for D below.) One possible strategy is to perform the PSUP iteration with q_{m-1}

until the iteration converges. However, there is no guarantee that all the extreme eigenvalues of A are computed by the Arnoldi procedure. The set D is contained in the lemniscate region [10] $L_m = \{z \in \mathbf{C} \mid |p_m(z)| \leq \epsilon\}$, where ϵ and $p_m = 1 - zq_{m-1}(z)$ solve (3.1). Moreover, the modulus of p_m is greater than ϵ outside L_m and tends to grow rapidly outside L_m, at least in some directions. If an eigenvalue λ lies outside L_m and $|p_m(\lambda)|$ is large enough, then the PSUP method will diverge.

One way to avoid this behavior is to invoke the adaptive procedure: if PSUP diverges then k_a additional Arnoldi steps are performed to compute k_a new eigenvalue estimates. These estimates are then used to construct a new enclosing set D and a new iteration polynomial q_{m-1}, with which the PSUP iteration is resumed. A good choice for a starting vector v_1 is the last residual from the previous PSUP iteration (normalized to have unit norm). For if PSUP diverges, then the residual will tend to be dominated by the eigenvectors whose eigenvalues are not being damped out by the PSUP polynomial. Moreover, this technique can be improved using GMRES. Once the k_a Arnoldi vectors are available, the GMRES(k_a) iteration (2.1) can be performed at relatively little extra expense. This has the effect of damping out from the residual the eigenvector components that were being enhanced by the previous PSUP iteration.

Rather than use the PSUP iteration alone, we consider a hybrid PSUP-GMRES method that makes use of these observations. This method consists of repeated iteration of some number s of PSUP steps, followed by a smaller number k_a of Arnoldi-GMRES steps. The initial eigenvalue estimates are provided by k_i Arnoldi-GMRES steps, where k_i may differ from k_a. In addition, the adaptive procedure is invoked immediately if the residual norm of the PSUP iteration increases by some tolerance τ relative to the smallest residual previously encountered. The following is a modification of the hybrid method developed in [6] that uses the PSUP iteration:

Algorithm 2: The hybrid GMRES-PSUP method.
Choose u_0. Compute $r_0 = b - Au_0$.
Until Convergence Do
Adaptive (Initialization) Steps: Set v_1 = the current normalized residual,
perform k_a (or k_i) Arnoldi/GMRES steps, and use the new eigenvalue estimates to update (or initialize) the PSUP coefficients.
PSUP Steps: While $(\|r_j\|/\|r_{min}\| \leq \tau)$)
Perform s steps of the PSUP iteration (Algorithm 1) to update the approximate solution u_j and residual r_j.

For the enclosing set D we take the union of the four sets D_j, where D_j is the convex hull of the set of eigenvalue estimates in the j-th quadrant of the complex plane. With this choice, if the extreme eigenvalues of each quadrant have been computed, then all the eigenvalues are contained in D. If all the eigenvalue estimates in either half plane are real, then the part of D containing these estimates is taken to be the line segment between the leftmost and rightmost estimates in the half plane.

There is no guarantee that the eigenvalue estimates computed by Arnoldi's method are accurate. Moreover, since the PSUP residual polynomial has the value 1 at the origin, if D contains points with both positive and negative real parts that are near the origin, then the Chebyshev norm of the residual polynomial will be very close to 1. (See Section 4 for an example.) We consider one heuristic designed to improve the performance of the hybrid PSUP method on problems with eigenvalues very near the origin: we successively remove the points closest to the origin from the set of eigenvalue estimates (and generate a smaller D) until the norm of the PSUP polynomial is smaller than some predetermined value η, and use that polynomial for the PSUP iteration.

There are two possible effects of this heuristic. If the deleted points are not accurate as eigenvalue estimates, then the resulting PSUP iteration will be just as robust and more rapidly convergent than if the deleted points had been included. On the other hand, if the deleted

points are good estimates, then the PSUP polynomial will probably be large on the deleted points, and the iteration will not damp out the residual in the direction of the corresponding eigenvectors. However, if the dimension of this eigenspace is small (say, 2 or 3), then the iteration should damp out the residual in all other components, so that the residual should be dominated by a small number of components. In this situation, a small number of GMRES steps should damp out these dominant components. We will refer to the hybrid PSUP method with this heuristic added as the GMRES/Reduced-PSUP scheme.

We note that with the methods of [24], (3.5) can be also solved with the constraint

$$\max_{z \in E} \left| \sum_{j=0}^{m-1} z^{j+1} a_j - 1 \right|_\Theta \leq 1,$$

where E is some finite set. In particular, if E is the set of deleted eigenvalue estimates in the GMRES/Reduced-PSUP scheme, then the PSUP polynomial on the reduced set D can be forced to be bounded in modulus by one on the deleted points. In experiments with this version of the GMRES/Reduced-PSUP iteration, we found its performance to be essentially the same as that of the unconstrained version described above.

4. Numerical Experiments

In this section, we compare the performance of CGN, GMRES(m), GMRES/PSUP and GMRES/Reduced-PSUP in solving several linear systems arising from a finite difference discretization of the differential equation

$$-\Delta u + 2P_1 u_x + 2P_2 u_y - P_3 u = f, \quad u \in \Omega, \tag{4.1}$$

$$u = g, \quad u \in \partial\Omega,$$

where Ω is the unit square $\{0 \leq x, y \leq 1\}$, and P_1, P_2 and P_3 are positive parameters. We use $f = g \equiv 0$, so that the solution to (4.1) is $u = 0$.

We discretize (4.1) by finite differences on a uniform $n \times n$ grid, using centered differences for the Laplacian and the first derivatives. Let $h = 1/(n+1)$. After scaling by h^2, the matrix equation has the form (1.1) in which the typical equation for the unknown $u_{ij} \approx u(ih, jh)$ is

$$(4-\sigma)u_{ij} - (1+\beta)u_{i-1,j} + (-1+\beta)u_{i+1,j} - (1+\gamma)u_{i,j-1} + (-1+\gamma)u_{i,j+1} = h^2 f_{ij},$$

where $\beta = P_1 h$, $\gamma = P_2 h$, $\sigma = P_3 h^2$ and $f_{ij} = f(ih, jh)$. The eigenvalues of A are given by [21]

$$4 - \sigma + 2\sqrt{1-\beta^2}\, cos\frac{s\pi}{n+1} + 2\sqrt{1-\gamma^2}\, cos\frac{t\pi}{n+1}, \qquad 1 \leq s, t \leq n.$$

The eigenvalues of the symmetric part are

$$4 - \sigma + 2cos\frac{s\pi}{n+1} + 2cos\frac{t\pi}{n+1}, \qquad 1 \leq s, t \leq n.$$

The leftmost eigenvalue of the symmetric part, corresponding to $s = t = n$, is given by

$$(2\pi^2 - P_3)h^2 + O(h^4),$$

so that for small enough h the symmetric part is indefinite when $P_3 > 2\pi^2$.

Six test problems corresponding to six choices of the parameter set $\{P_1, P_2, P_3\}$ are considered. We use the three values $P_3 = 30$, 80, and 250 together with each of the pairs of

values $\{P_1 = 1,\ P_2 = 2\}$ and $\{P_1 = 25,\ P_2 = 50\}$. For all tests, $n = 31$, so that the order $N = n^2$ is 969. For all six test problems, the coefficient matrix A is indefinite, and the number of negative eigenvalues of $(A + A^T)/2$ is increasing as P_3 grows. For the first choice of the (P_1, P_2) pair, A is mildly nonsymmetric and its eigenvalues are real, and for the second choice, A is more highly nonsymmetric and has complex eigenvalues.

Although it is not our intention here to examine preconditioners for indefinite systems, preconditioning has been shown to be a critical factor in the performance of iterative methods [3, 5, 15]. In our tests, we precondition (1.1) by the finite difference discretization of the Laplacian. That is, the iterative methods being considered are applied to the *preconditioned problem*

$$AQ^{-1}\hat{x} = b, \quad x = Q^{-1}\hat{x},$$

where Q is the discrete Laplacian. (See [2] for an asymptotic analysis of this preconditioner for finite element discretizations.) The preconditioned matrix-vector product then consists of a preconditioning solve of the form $Q^{-1}v$ and a matrix multiply of the form Av. Since Ω is a square domain, the preconditioning is implemented using the block cyclic reduction method at a cost of $3n^2\log_2 n$ operations [25]. We have confirmed numerically that the preconditioned matrix AQ^{-1} in all six problems has indefinite symmetric part.

We use the following parameters for the hybrid GMRES-PSUP iteration. In an effort to obtain the dominant and subdominant eigenvalues of each quadrant at the outset, the initialization step consists of eight GMRES steps ($k_i = 8$) giving eight eigenvalue estimates. All subsequent calls to the adaptive procedure consist of four GMRES steps ($k_a = 4$). For all tests with PSUP, we use a residual polynomial of degree four ($m = 4$), and allow at most $s = 32$ PSUP steps (or eight successive applications of the PSUP polynomial). The adaptive procedure is invoked if the residual norm increases during a PSUP step ($r = 1$), or after s steps are performed. We use $M = 100$ points for the discretized enclosing set ∂D_M, and allocate them so that the number of points in each quadrant is approximately proportional to the circumference of the convex hull in that quadrant. For subsets of D that overlap on quadrant boundaries (e.g. if a line segment on the real line is shared by regions in the first and fourth quadrants), the shared boundary is discretized twice. For the GMRES/Reduced-PSUP scheme, in which eigenvalue estimates closest to the origin are deleted until the minimax norm is less than some tolerance η, we examine $\eta = .5$ and .3. For this scheme, we take k_a to be two plus the number of eigenvalue estimates deleted. We use the notation GMRES-PSUP(m) (with $m = 4$) for the "unreduced" scheme, and GMRES-PSUP(m, η) for the reduced version.

We examine GMRES(m) for $m = 5$ and $m = 20$. Recall that the latter version generates a higher degree optimal polynomial at the expense of a larger average cost per step.

All numerical tests were run on a VAX 11-780 in double precision (55 bit mantissa). The initial guess in all runs was a vector u_0 of random numbers between -1 and 1. Figures 1 - 6 show the performance of the methods measured in terms of multiplication counts, for the six problems (also numbered 1 - 6). Note that the horizontal scale of Figure 1 is wider than the others, and the scales in Figures 5 and 6 are slightly narrower. Table 1 shows the iteration counts needed to satisfy the stopping criterion of

$$\frac{\|r_j\|_2}{\|r_0\|_2} \leq 10^{-6}.$$

A maximum of 100, 150, and 200 iterations were permitted for the CGN, GMRES and PSUP methods, respectively. (For these iteration counts, CGN, GMRES(20) and GMRES-PSUP(4) performed roughly the same number of operations.) Our main observations on this data are:

1. Problems 1 and 3 are solved efficiently by nearly all the methods, but for the other four problems convergence is slow.

2. In general, the hybrid GMRES-PSUP(m) scheme is weakest. The plateaus in Figures 3, 5 and 6 for this method correspond to the PSUP step, for which convergence is very slow. The "reduction" heuristic improves the performance, but the improvement is due largely to increased effectiveness of the GMRES part of the iteration (e.g. in the steep drops of Figures 2 - 4), and the improved performance is not better than that of GMRES alone.
3. On the whole, GMRES(20) and CGN are the most effective methods for these problems, but they are not dramatically superior to the others. GMRES(20) converges more rapidly than GMRES(5).

Excluding storage for the matrix and right hand side, the storage requirements for the methods considered are

CGN:	$4N$
GMRES(5):	$7N$
GMRES(20):	$22N$
All PSUP variants:	$10N$

The high cost of the PSUP methods is due to the eight initializing GMRES steps.

Although the GMRES/Reduced-PSUP (PSUP(m, η)) scheme is not as fast as pure GMRES, the reduction heuristic does have its intended effect of improving upon the hybrid scheme. We briefly examine the effect of the heuristic on Problem 3, focusing on two curve segments of Figure 3: the plateau of curve D (GMRES-PSUP(4)) between multiplication counts 200000 and 300000, and the last plateau in curve E (GMRES-PSUP(4,.5)). For curve D, on return from the adaptive step at about multiplication count 200000, the real parts of the eigenvalue estimates lie in the intervals [-3,-.33] and [0.4,.98], the Chebyshev norm of the residual polynomial is .98, and convergence is slow. For curve E, on return from the adaptive step prior to the last plateau of the curve, the real parts of the eigenvalue estimates lie in the intervals [-3,-.56] and [.05,.97], and the Chebyshev norm is .96. The effect of deletion of points is shown in Table 2. The Chebyshev norm is very large when there are points near the origin, and it declines as these points are deleted. The deletion of points does not significantly hurt the PSUP part of the iteration and it strongly enhances the effect of the GMRES steps.

Problem #	1	2	3	4	5	6
CGN	13	>100	28	>100	>100	>100
GMRES(5)	13	>150	46	>150	>150	>150
GMRES(20)	10	111	17	119	>150	>150
GMRES-PSUP	16	>200	199	>200	>200	>200
PSUP(4,.5)	16	>200	62	>200	>200	>200
PSUP(4,.3)	16	>200	70	>200	>200	>200

Table 1: Iteration counts.

Deleted Points	Intervals Containing Real Parts	Chebyshev Norm
-	[-3, -.56], [.05,.97]	.96
.05	[-3, -.56], [.34,.97]	.76
.34	[-3, -.56], [.61,.97]	.55
-.56	[-3,-1.46], [.61,.97]	.33

Table 2: Effect of point deletion on GMRES/Reduced-PSUP(4,.5) for Problem 3.

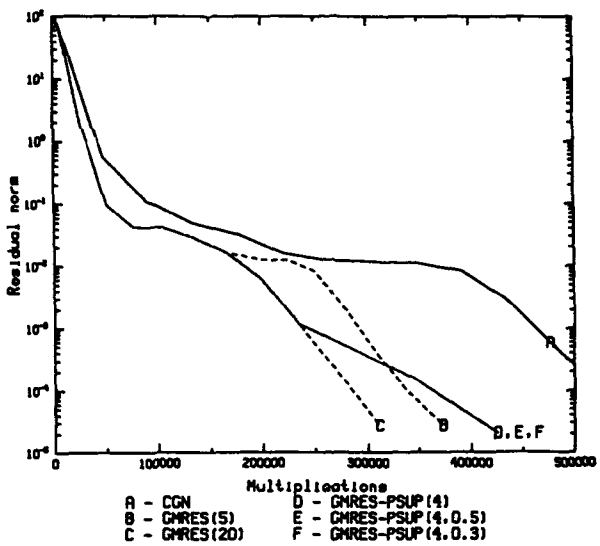


Figure 1: $P_1 = 1$, $P_2 = 2$, $P_3 = 30$

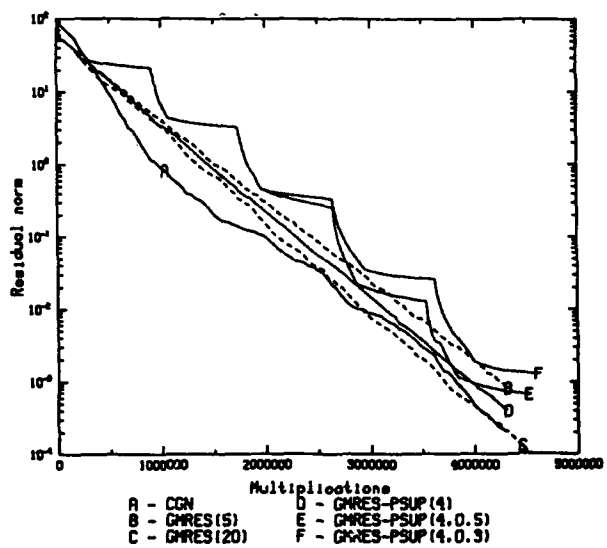


Figure 2: $P_1 = 25$, $P_2 = 50$, $P_3 = 30$

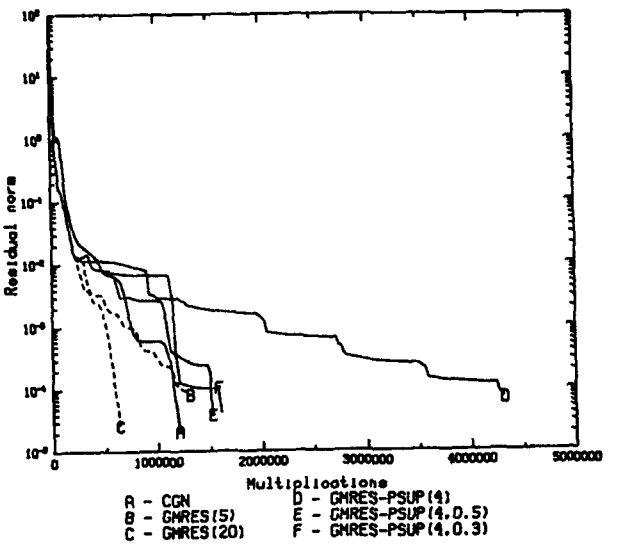


Figure 3: $P_1 = 1$, $P_2 = 2$, $P_3 = 80$

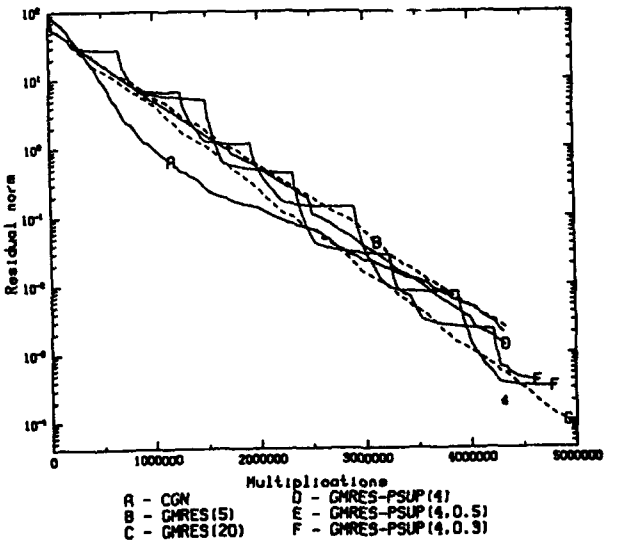


Figure 4: $P_1 = 25$, $P_2 = 50$, $P_3 = 80$

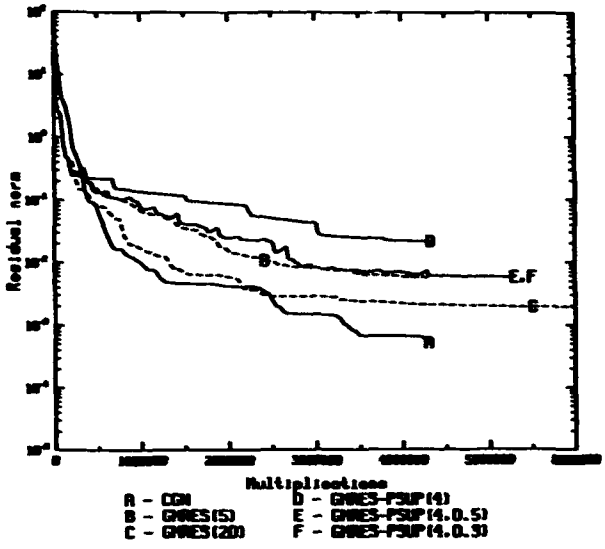


Figure 5: $P_1 = 1$, $P_2 = 2$, $P_3 = 250$

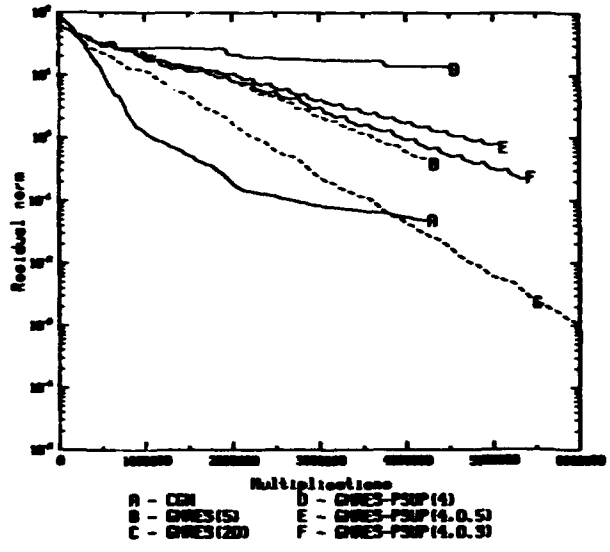


Figure 6: $P_1 = 25$, $P_2 = 50$, $P_3 = 250$

We remark that we also considered other variants of the PSUP iteration. In experiments with degrees $m = 6$ and 10 the performance of PSUP was essentially the same.* Moreover, as we noted in Section 3, a variant of the GMRES/Reduced-PSUP in which the PSUP polynomial is constrained to be bounded in modulus by one on the set of deleted eigenvalue estimates displayed about the same behavior as the unconstrained version. Similarly, we tested LSQR [16], a stabilized version of CGN, and found that its performance was nearly identical to CGN.

5. Conclusions

The GMRES and PSUP methods are iterative methods that are optimal in the class of polynomial-based methods with respect to the Euclidean or l_∞ norms respectively, for arbitrary nonsingular linear systems. For linear systems in which the coefficient matrix is either symmetric or definite (or both), these types of methods are effective solution techniques [3, 5]. In particular, they are superior to solving the normal equations by the conjugate gradient method. In the results of Section 4, the methods based on polynomials in the coefficient matrix are not dramatically superior to CGN, especially for systems that are both highly nonsymmetric and highly indefinite. GMRES appears to be a more effective method than PSUP.

We note that the best results for other classes of problems depend strongly on preconditioning. We used the discrete Laplacian as a preconditioner in our experiments, and the large iteration/work counts in the results show that this is not a good choice for the given mesh size when the coefficients in the differential operator are large. We believe that improvements in preconditioners are needed to handle this class of problems.

*In some tests with degree 16, we were unable to generate the polynomial coefficients. We believe the choice of the powers of z as basis functions makes (3.5) ill conditioned for large m; see [19]. In addition, the implementation based on Horner's rule may suffer from instability for large m.

References

[1] A. Bayliss, C. I. Goldstein and E. Turkel, *An iterative method for the Helmholtz equation,* Journal of Computational Physics, 49 (1983), pp. 443–457.

[2] J. H. Bramble and J. E. Pasciak, Preconditioned iterative methods for nonselfadjoint or indefinite elliptic boundary value problems, H. Kardestuncer ed., *Unification of Finite Element Methods,* Elsevier Science Publishers, New York, 1984, pp. 167–184.

[3] R. Chandra, *Conjugate Gradient Methods for Partial Differential Equations,* Ph.D. Thesis, Department of Computer Science, Yale University, 1978. Also available as Technical Report 129.

[4] C. de Boor and J. R. Rice, *Extremal polynomials with application to Richardson iteration for indefinite linear systems,* SIAM J. Sci. Stat. Comput., 3 (1982), pp. 47–57.

[5] H. C. Elman, *Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations,* Ph.D. Thesis, Department of Computer Science, Yale University, 1982. Also available as Technical Report 229.

[6] H. C. Elman, Y. Saad and P. E. Saylor, *A Hybrid Chebyshev Krylov-Subspace Method for Nonsymmetric Systems of Linear Equations,* Technical Report YALEU/DCS/TR-301, Yale University Department of Computer Science, 1984. To appear in SIAM J. Sci. Stat. Comput.

[7] R. Fletcher, Conjugate gradient methods for indefinite systems, G. A. Watson ed., *Numerical Analysis Dundee 1975,* Springer-Verlag, New York, 1976, pp. 73–89.

[8] L. A. Hageman and D. M. Young, *Applied Iterative Methods,* Academic Press, New York, 1981.

[9] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems,* Journal of Research of the National Bureau of Standards, 49 (1952), pp. 409–435.

[10] E. Hille, Volume II: *Analytic Function Theory,* Blaisdell, New York, 1962.

[11] K. Ito, *An Iterative Method for Indefinite Systems of Linear Equations,* Technical Report NAS1-17070, ICASE, April 1984.

[12] T. Kerkhoven, *On the Choice of Coordinates for Semiconductor Simulation,* Technical Report RR-350, Yale University Department of Computer Science, 1984.

[13] T. A. Manteuffel, *Adaptive procedure for estimation of parameters for the nonsymmetric Tchebychev iteration,* Numer. Math., 31 (1978), pp. 187–208.

[14] ———, *The Tchebychev iteration for nonsymmetric linear systems,* Numer. Math., 28 (1977), pp. 307–327.

[15] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix,* Math. Comp., 31 (1977), pp. 148–162.

[16] C. C. Paige and M. A. Sanders, *LSQR: An algorithm for sparse linear equations and sparse least squares,* ACM Trans. on Math. Software, 8 (1982), pp. 43–71.

[17] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations,* SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[18] Y. Saad, *Iterative solution of indefinite symmetric systems by methods using orthogonal polynomials over two disjoint intervals,* SIAM J. Numer. Anal., 20 (1983), pp. 784–811.

[19] ———, *Least squares polynomials in the complex plane with applications to solving sparse nonsymmetric matric problems,* Technical Report 276, Yale University Department of Computer Science, June 1983.

[20] Y. Saad and M. H. Schultz, *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems,* Technical Report 254, Yale University Department of Computer Science, 1983.

[21] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods,* Oxford University Press, New York, 1978.

[22] D. C. Smolarski and P. E. Saylor, *Optimum Parameters for the Solution of Linear Equations by Richardson's Iteration,* May 1982. Unpublished manuscript.

[23] R. L. Streit, *An Algorithm for the Solution of Systems of Complex Linear Equations in the l_∞ Norm with Constraints on the Unknowns,* 1983. Submitted to ACM Trans. on Math. Software.

[24] ———, *Solution of Systems of Complex Linear Equations in the l_∞ Norm with Constraints on the Unknowns,* Technical Report 83-3, Systems Optimization Laboratory, Stanford University Department of Operations Research, 1983. To appear in SIAM J. Sci. Stat. Comput.

[25] P. N. Swarztrauber, *The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle,* SIAM Review, 19 (1977), pp. 490–501.

[26] M. A. Saunders, H. D. Simon, and E. L. Yip, *Two Conjugate-Gradient-Type Methods for Sparse Unsymmetric Linear Equations,* Technical Report ETA-TR-18, Boeing Computer Services, June 1984.

[27] J. H. Wilkinson, *The Algebraic Eigenvalue Problem,* Oxford University Press, London, 1965.

Extremals And Zeros In Markov Systems Are Monotone Functions Of One Endpoint

R. L. Streit

Extremals and Zeros in Markov Systems are Monotone Functions of One Endpoint

ROY L. STREIT

Abstract. The synthesis of optimum field patterns for discrete linear antenna arrays leads naturally to the study of the behavior of generalized Chebyshev polynomials as a function of one endpoint of the interval of definition. Of particular importance in this application is the variation of the zeros and the extreme points of the generalized Chebyshev polynomials as the left-hand endpoint of the interval is shifted to the right. All the zeros and all the extreme points of the classical Chebyshev polynomials defined on the intervals [t, b] are strictly increasing functions of t. In Haar Systems, this property does not hold, but in Markov Systems with unit element it does. An apparently new extremal property in Haar Systems is proved and then used to show that in every Haar System the first zero must be an increasing function of t. If, in addition, the Haar System has a unit element, then the first zero must be strictly increasing.

INTRODUCTION AND MOTIVATION

Visualize any number M of fixed points in the plane, all collinear and spaced symmetrically about the center of the shortest line segment containing all the points. Take the origin to be the center of this smallest line segment. If each of these points is taken as the position of a sensor of an antenna array, then the directional response of this array is directly proportional to the absolute value of

$$P(u) = \sum_{k=1}^{N} a_k \cos \xi_k u \quad , \; 0 \leq u \leq \pi \quad , \tag{1}$$

where $N = \left[\frac{M+1}{2}\right]$, ξ_k is a constant multiple of the distance of the k-th point to the right of the origin, the variable u can be regarded as an angle measured from a normal to the line of points, and the coefficients a_k are any real constants. The coefficients a_k are the design parameters and are chosen to enhance the directionality of the array.

Usually, the direction perpendicular to the line of points is the desirable direction and all other directions are of less interest. Thus, coefficients a_k are chosen so that $P(u)$ has its largest magnitude near or at the point $u = 0$, while keeping $P(u)$ as small as possible in magnitude elsewhere. With this objective in mind, the u domain is split into two parts: the "mainlobe" region $[0, u_0]$ and the "sidelobe" region $[u_0 , \pi]$. The coefficients a_k are defined to be optimal if and only if the ratio of L_∞ norms

$$\| P(u) \|_{[0, u_0]} \div \| P(u) \|_{[u_0, \pi]} \tag{2}$$

is the largest possible for any coefficient set. Theorem 2 of this paper gives conditions under which this ratio is maximized by minimizing the denominator independently of the numerator. That is, if the set of cosines in (1) forms a Haar System, then the optimal coefficients are proportional to the coefficients of the generalized Chebyshev

polynomial for the interval $[u_0, \pi]$. With these optimal coefficients, the peak point of a sidelobe is just an extreme point of a uniform function approximation.

The ratio (2) is therefore maximized as a function of the parameter u_0 dividing the mainlobe and the sidelobe regions. In engineering applications, it is important to know how this ratio changes with u_0 and how this change shows up in the actual field pattern (1). Theorem 3 gives necessary and sufficient conditions for the maximum value of the ratio (2) to be a strictly increasing function of u_0 , while Theorem 1 gives conditions under which all the zeros and all the sidelobes of the optimal field pattern shift strictly to the right with increasing u_0 . Finally, since the first zero of $P(u)$ is sometimes used as a measure of the mainlobe region $[0, u_0]$, Theorem 4 gives weaker conditions under which this measure is also strictly increasing with increasing u_0 .

Pokrovskii [1] studied this problem in detail, but for equispaced sensors only. In the equispaced case, a transformation of variables reduces the problem to a study of ordinary polynomials. An explicit solution to the maximization of the ratio (2) for equispaced sensors was given by Dolph [2]. A generalization of Dolph's method to symmetrically spaced sensors can be found in Streit [3], [4].

THE MATHEMATICS

Let $T_n(u) \equiv \cos [n \cos^{-1} u]$ be the Chebyshev polynomial of degree n on the closed interval $[-1, 1]$. Then the Chebyshev polynomial on the closed interval $[t, b]$ is just

$$T_n\left(\frac{2}{b-t}x - \frac{b+t}{b-t}\right) \quad , \; t \leq x \leq b \quad ,$$

which has the n zeros (counting from left to right)

$$z_k(t) = \frac{t}{2}\left[1 - \cos\left(n - k + \frac{1}{2}\right)\frac{\pi}{n}\right]$$

$$+ \frac{b}{2}\left[1 + \cos\left(n - k + \frac{1}{2}\right)\frac{\pi}{n}\right] \quad ,$$

$$k = 1 \; , \; \ldots \; , \; n \; ,$$

and the $(n + 1)$ extremal points

$$x_k(t) = \frac{t}{2}\left[1 - \cos\,(n - k + 1)\,\frac{\pi}{n}\right]$$

$$+ \frac{b}{2}\left[1 + \cos(n - k + 1)\,\frac{\pi}{n}\right] \quad ,$$

$$k = 1 \; , \; \ldots \; , \; n + 1 \; .$$

By inspection, except for $x_{n+1}(t)$, all the zeros and all the extremals are strictly increasing continuous functions of the left-hand endpoint t . Theorem 1 extends this property to more general spaces.

Let $C[a, b]$ be the linear space of all real valued and continuous functions on the closed finite interval $[a, b]$. Meinardus [5] defines the finite or infinite sequence of linear subspaces V_n , $n = 0 \; , \; 1, \; 2, \; \ldots$ of $C[a, b]$ to

be a Haar System provided it has the three properties:

i. $V_n \subset V_{n+1}$

ii. The dimension of V_n is $n + 1$

iii. V_n satisfies the Haar condition, i.e., $f \in V_n$ and $f \not\equiv 0$ implies that f has at most n zeros in $[a, b]$.

Define $V_n(t)$ to be that linear subspace of $C[t, b]$ obtained by restricting every function in V_n to the interval $[t, b] \subset [a, b]$. Thus, $V_n(t)$ forms a Haar System for each $t \in [a, b)$. For $f \in C[a, b]$, define

$$\| f \|_{\mathcal{A}} = \max_{x \in \mathcal{A}} | f(x) | \quad , \mathcal{A} = \overline{\mathcal{A}} \subset [a, b] \ ,$$

and

$$\rho_n(f; t) = \min_{g \in V_n(t)} \| f - g \|_{[t, b]} \ .$$

A Haar System with unit element is a Haar System for which V_0 contains the constant functions. Zielke [6] proves the following lemma in a more general form.

Lemma 1. If V_n is a Haar System with unit element on the closed interval $[a, b]$, and if $f \in V_{n+1} \setminus V_{n-1}$, then there exist at most $(n + 1)$ points $x_k \in [a, b]$ such that f is strictly monotone on each interval $[x_k, x_{k+1}]$, $k = 1, 2, \ldots, n + 1$.

Define $S_0^{(t)}(x) \equiv 1$ for all x and t. The following lemma, due to Meinardus [7], [5], is an immediate consequence of de la Vallée Poussin's Theorem and Lemma 1.

Lemma 2. For a Haar System V_n with unit element, the functions $S_n^{(t)}$ satisfying the conditions

a. $S_n^{(t)} \in V_n(t)$,

b. $\| S_n^{(t)} \|_{[t,b]} = 1$,

c. $\rho_{n-1}(S_n^{(t)} ; t) = 1$,

d. $S_n^{(t)}(b) = 1$

have the following properties:

1. $S_n^{(t)}$ is uniquely defined for each $n \geq 0$ and $t \in [a, b)$.
2. $S^{(t)}$ possesses precisely n+1 extremal points $x_k(t)$ $(k = 1,2,\ldots, n + 1)$ in the interval $[t, b]$. The points t and b are extremal points, and arranging the points in increasing order

$$t = x_1(t) < x_2(t) < \ldots < x_n(t) < x_{n+1}(t) = b ,$$

we have the alternating property

$$S_n^{(t)}\left(x_k(t)\right) + S_n^{(t)}\left(x_{k+1}(t)\right) = 0 \ (k = 1,2,\ldots,n).$$

3. $S_n^{(t)}$ is strictly monotone (increasing or decreasing) in $x_k(t) \leq x \leq x_{k+1}(t)$ $(k = 1, 2,\ldots,n)$.

Lemma 1 does not imply that V_n' is a Haar System if V_n contains only continuously differentiable functions. Let $V_0 = \{1\}$ and $V_1 = \{1, h\}$ where h is any continuously differentiable strictly increasing function on $[a, b]$ that has, say, 15 inflection points in (a, b). Then V_0, V_1 is a continuously differentiable Haar System, but every function in V_1' has 15 zeros in (a, b).

The system V_n is defined here to be a Markov System with unit element on the closed interval [a, b] if V_n is a Haar System of functions continuous on [a, b], having continuous derivative on the open interval (a, b), and with the property that the spaces V_n' spanned by the derivatives of functions in V_n form a Haar System on (a, b). Therefore, $f \in V_n'$ and $f \not\equiv 0$ implies that f has at most $n - 1$ zeros in (a, b). Let $V_n'(t)$ be the restriction of V_n' to (t, b). We now prove an extension of Lemma 2 to Markov Systems with unit element.

Theorem 1. Let V_n be a Markov System with unit element on the closed interval [a, b]. Then the functions $S_n^{(t)}$ satisfying conditions a. through d. of Lemma 2 also have the following additional properties:

4. Each function $x_k(t)$ $(k = 1, \ldots, n)$ is a continuous and strictly monotonically increasing function of t.
5. The zeros $z_k(t)$ $(k = 1, \ldots, n)$ of $S_n^{(t)}$, arranged in the increasing order

$$t < z_1(t) < z_2(t) < \ldots < z_{n-1}(t) < z_n(t) < b,$$

are all continuous and strictly monotonically increasing functions of t.
6. $S_n^{(t)}(x)$ is strictly monotone (increasing or decreasing) in the interval $a \leq x \leq x_2(t)$.

Proof. To prove property 6, note that Rolle's Theorem implies that $\left[S_n^{(t)}(x)\right]'$ has exactly n-1 zeros in the interval $[x_2(t), z_n(t))$, and so must have none in the interval $[a, x_2(t))$, because any function in V_n' has at most n-1 zeros in [a, b]. Thus, $S_n^{(t)}$ must be monotone in the interval $[a, x_2(t)]$. The continuity of $x_k(t)$ and $z_k(t)$ $(k = 1, 2, \ldots, n)$ follows from a remark in Meinardus

[5, p. 85]. Now, fix $t \in [a, b)$. By continuity, there exists $\delta > 0$ such that for each $0 < \varepsilon < \delta$, the sets $I_k(\varepsilon)$ defined by

$$I_k(\varepsilon) = \begin{cases} \left[x_k(t),\ x_k(t+\varepsilon)\right], & \text{if } x_k(t) < x_k(t+\varepsilon) \\ \{x_k(t)\}, & \text{if } x_k(t) = x_k(t+\varepsilon) \\ \left[x_k(t+\varepsilon),\ x_k(t)\right], & \text{if } x_k(t) > x_k(t+\varepsilon) \end{cases}$$

$$(k = 1, 2, \ldots, n+1)$$

are pairwise disjoint. Put

$$S(x;\varepsilon) = S_n^{(t)}(x) - S_n^{(t+\varepsilon)}(x) \quad .$$

Because of property 6, $S(x;\varepsilon)$ has no zero in $I_1(\varepsilon)$ and precisely one zero in each of the remaining intervals. Thus the intervals $I_2(\varepsilon), \ldots, I_{n+1}(\varepsilon)$ contain all the zeros of $S(x;\varepsilon)$. Because of property 2,

$$x_1(t) = t < t + \varepsilon = x_1(t+\varepsilon) \ .$$

Let $j < n+1$ be the smallest integer such that there exists $\hat{\varepsilon} < \delta$, $\hat{\varepsilon} > 0$, and $x_j(t+\hat{\varepsilon}) \leq x_j(t)$. If strict inequality holds, then the interval $(x_{j-1}(t+\hat{\varepsilon}), x_j(t+\hat{\varepsilon}))$ is disjoint from the intervals $I_2(\hat{\varepsilon}), \ldots, I_{n+1}(\hat{\varepsilon})$ but must contain a zero of $S(x;\hat{\varepsilon})$. On the other hand, if $x_j(t+\hat{\varepsilon}) = x_j(t)$, then $S'(x;\hat{\varepsilon})$ has a zero at $x_j(t)$. But Rolle's Theorem already gives $S'(x;\hat{\varepsilon})$ at least n-1 zeros, one between each consecutive pair of zeros

of $S(x;\hat{\varepsilon})$. Therefore, $S'(x;\hat{\varepsilon})$ contradicts the hypothesis on $V_n'(t)$. Hence, $j = n + 1$ and property 4 follows. Finally, property 5 is an immediate consequence of property 4, for otherwise the function $S(x;\hat{\varepsilon})$ will have too many zeros. This completes the proof.

The next object is to weaken the hypotheses of Theorem 1, specifically, by dropping the unit element and the differentiability conditions. One method of achieving this end leads to Theorem 2. First, define $\{h_0 , h_1 , \dots , h_n\}$ to be a basis for the Haar System V_n if $\{h_0 , h_1 , \dots , h_k\}$ is a basis for V_k , $k = 0, 1, \dots , n$. Also, the functions $S_n^{(t)}$ are defined above only for Haar Systems with unit element. For the general Haar System, define $S_n^{(t)} \equiv c(t)\cdot(h_n + g^*)$, where $g^* \in V_{n-1}(t)$ and the constants $c(t)$ are uniquely defined by

$$[c(t)]^{-1} \equiv \| h_n + g^* \|_{[t,b]} = \min_{g \in V_{n-1}(t)} \| h_n + g \|_{[t,b]}$$

Theorem 2. Let V_n be a Haar System on the closed interval $[a, b]$. Let $\{h_0, h_1, \dots , h_n\}$ be a basis for V_n . For fixed r and s satisfying $a \le r \le s \le t < b$, define

$$M_n(t) = \max_{g \in V_{n-1}} \left\{ \frac{\| h_n(x) + g \|_{[r,s]}}{\| h_n(x) + g \|_{[t,b]}} \right\} .$$

Then

a. the ratio of norms is maximized by the best approximation to h_n on $[t, b]$, namely $g^* \in V_{n-1}$,

b. $M_n(t) = \| S_n^{(t)} \|_{[r,s]}$,

c. $M_n(t)$ is a continuous increasing function of t . If, in addition, V_n has a unit element, then also

b'. $M_n(t) = | S_n^{(t)}(r) |$.

Proof. Suppose $h \in V_n$, $h \not\equiv S_n^{(t)}$, and

$$\frac{\| h \|_{[r,s]}}{\| h \|_{[t,b]}} > \frac{\| S_n^{(t)} \|_{[r,s]}}{\| S_n^{(t)} \|_{[t,b]}} \quad . \tag{3}$$

There exists a constant $k > 0$ such that

$$\| kh \|_{[r,s]} = \| S_n^{(t)} \|_{[r,s]} , \tag{4}$$

so that, because of (3) ,

$$\| kh \|_{[t,b]} < \| S_n^{(t)} \|_{[t,b]} .$$

The alternating properties of $S_n^{(t)}$ (de la Vallée Poussin's Theorem) imply that both of the functions

$$S_n^{(t)}(x) \pm kh(x) \in V_n(t) \tag{5}$$

have n zeros in the open interval (t,b). But because of (4), one of these functions also has a zero in the interval $[r, s]$. Thus, one of the functions (5) violates the Haar condition on V_n . This contradiction establishes that either $h \equiv S_n^{(t)}$ or that (3) is an equality. Either way, $S_n^{(t)}$ maximizes the ratio of norms and part (a) is established. Part(b)

follows easily from part (a). The continuity of $M_n(t)$ follows from part (b). To prove that $M_n(t)$ is increasing, suppose there exist $\beta > \alpha \geq s$ such that $M_n(\beta) < M_n(\alpha)$. Then for some constant $k \neq 0$,

$$\| kS_n^{(\beta)} \|_{[r,s]} = \| S_n^{(\alpha)} \|_{[r,s]}$$

and

$$\| kS_n^{(\beta)} \|_{[\beta,b]} > \| S_n^{(\alpha)} \|_{[\alpha, b]} .$$

Hence, both of the functions

$$kS_n^{(\beta)}(x) \pm S_n^{(\alpha)}(x)$$

have n zeros on the interval $[\beta, b]$. But one of them also has a zero in the interval $[r,s]$, contrary to the assumption that V_n is a Haar System. Thus, part (c) is established. If V_n has a unit element, then Lemma 1 guarantees that $S_n^{(t)}$ is monotone in $[r,s]$, so that (b') must be true. This completes the proof.

Theorem 3. Under the assumptions of Theorem 2, for all $\beta > \alpha \geq s$,

$$M_n(\alpha) = M_n(\beta) \tag{6}$$

if and only if

$$S_n^{(\alpha)}(x) = S_n^{(\beta)}(x) \tag{7}$$

for each $x \in [a,b]$.

Proof. If (7) holds, then (6) follows from Theorem 2. Suppose then that (6) holds, but that (7) does not. Let $x_1(t)$ denote the $(n+1)$st extremal point of $S_n^{(t)}$ counted from the right-hand endpoint to the left. If $h = x_1(\alpha) > \alpha$, then $S_n^{(\alpha)} \equiv S_n^{(t)}$ for all $t \in [\alpha, h]$, so that (7) holds if $\beta \leq h$. Hence without loss of generality, we may take $x_1(\alpha) = \alpha < \beta = x_1(\beta)$. Also, because $x_1(t)$ is continuous and $M_n(t)$ is monotone, it may be assumed without loss of generality that $x_1(\beta)$ is strictly less than the first zero of $S_n^{(\alpha)}$ and that $S_n^{(\beta)}(\beta)$ is of the same sign as $S_n^{(\alpha)}(\beta)$. Define the functions $T_n^{(t)}(x) \equiv S_n^{(t)}(x) \cdot c(t)$. (Thus, the coefficient of h_n is always unity.) Because $T_n^{(\beta)} \not\equiv T_n^{(\alpha)}$, the function $T(x) \equiv T_n^{(\alpha)}(x) - T_n^{(\beta)}(x)$ lies in V_{n-1} and cannot be identically zero. Because $[\beta, b]$ is a subset of $[\alpha, b]$ and $T(x) \not\equiv 0$,

$$\| T_n^{(\beta)} \|_{[\beta, b]} < \| T_n^{(\alpha)} \|_{[\alpha, b]}, \tag{8}$$

and because of (6),

$$\| T_n^{(\beta)} \|_{[r,s]} < \| T_n^{(\alpha)} \|_{[r,s]}. \tag{9}$$

Now if $|T_n^{(\beta)}(\beta)| \leq |T_n^{(\alpha)}(\beta)|$, then $T(x)$ has n zeros in $[\beta, b]$ because of de la Vallee Poussin's Theorem and (8), contradicting $T(x) \in V_{n-1}$ and $T(x) \not\equiv 0$. Therefore, $|T_n^{(\beta)}(\beta)| > |T_n^{(\alpha)}(\beta)|$ and $T(x)$ has only $n-1$ zeros in $[\beta, b]$. However, $T(x)$ has another zero in $[r, \beta)$ because of the Intermediate Value Theorem, (9), and the fact that $T_n^{(\beta)}(\beta)$ and $T_n^{(\alpha)}(\beta)$ are of the same sign. Thus, $T(x) \in V_{n-1}$ and $T(x) \not\equiv 0$ is a contradiction. This completes the proof.

Corollary. Under the assumptions of Theorem 2, if V_n has a unit element, then also

c'. $M_n(t)$ is strictly increasing in t.

Proof. $S_n^{(\alpha)} \equiv S_n^{(\beta)}$ cannot occur for $\alpha \neq \beta$ because α and β must be extremal points of $S_n^{(\alpha)}$ and $S_n^{(\beta)}$, respectively.

The next theorem is the weakened form of Theorem 1 that was sought earlier.

Theorem 4. Let V_n be any Haar System, and let $\{h_0, h_1, \ldots, h_n\}$ be a basis for the system. Let $z_1(t)$ be the smallest zero in the interval $[t, b]$ of $S_n^{(t)}$. Then $z_1(t)$ is a monotonically increasing function of t. Furthermore, if V_n has a unit element, then $z_1(t)$ is strictly monotone.

Proof. Consider first the case for V_n without a unit element. Suppose there exists $\beta > \alpha \geq a$ such that $z_1(\beta) < z_1(\alpha)$. Therefore, $S_n^{(\alpha)} \not\equiv S_n^{(\beta)}$ and by Theorem 3,

$$\frac{\| S_n^{(\beta)} \|_{\{a\}}}{\| S_n^{(\beta)} \|_{[\beta,b]}} > \frac{\| S_n^{(\alpha)} \|_{\{a\}}}{\| S_n^{(\alpha)} \|_{[\alpha,b]}} .$$

Thus, there is a constant $k \neq 0$ such that

$$\| kS_n^{(\beta)} \|_{\{a\}} = \| S_n^{(\alpha)} \|_{\{a\}} \tag{10}$$

and

$$\| kS_n^{(\beta)} \|_{[\beta,b]} < \| S_n^{(\alpha)} \|_{[\alpha,b]} .$$

Because $z_1(\beta) < z_1(\alpha)$, de la Vallée Poussin's Theorem guarantees that both the functions

$$S_n^{(\alpha)} \pm kS_n^{(\beta)} \in V_n \tag{11}$$

have n zeros on $[\beta, b]$, while (10) guarantees that one of them has a zero at $x = a$. Therefore, one of the functions (11) violates the Haar condition of V_n . In the case where V_n has a unit element, $\beta > \alpha$ guarantees by Lemma 2 that $S_n^{(\alpha)} \not\equiv S_n^{(\beta)}$. The supposition that $z_1(\beta) \leq z_1(\alpha)$ leads to a contradiction in the same manner as before. This completes the proof.

Finally, we note that Theorem 2 fails if V_n does not satisfy the Haar condition.

Example. The linear space spanned by the functions $\{1, \sin x\}$ on the interval $[0, \pi]$ is not Haar. For $[r, s] = [0, \pi/2]$,

$$\frac{\| \sin x + a \|_{[0, \pi/2]}}{\| \sin x + a \|_{[\pi/2, \pi]}} = 1$$

for all constants a, including the constant of best approximation to $\sin x$ on the internal $[\pi/2, \pi]$, namely, $a = 1/2$. When the interval $[r, s]$ is replaced by the point $x = \pi/6$, then

$$\frac{\| \sin x \|_{\{\pi/6\}}}{\| \sin x \|_{[\pi/2, \pi]}} = \frac{1}{2} > 0 = \frac{\| \sin x - \frac{1}{2} \|_{\{\pi/6\}}}{\| \sin x - \frac{1}{2} \|_{[\pi/2, \pi]}} .$$

Thus, the results of Theorem 2 do not hold and $M_n(t)$ is not achieved by minimizing the denominator.

Acknowledgment. I would like to thank the referee for suggesting a helpful change in notation, and for pointing out the articles by Zielke and Pokrovskii.

REFERENCES

[1] V. L. Pokrovskii, "On a class of polynomials with extremal properties," AMS Translations, vol. 19, 1962, pp 199-219.

[2] C. L. Dolph, "A current distribution of broadside arrays which optimizes the relationship between beam width and side-lobe level," Proc. IRE Waves Elections, vol. 34, June 1946, pp 335-348.

[3] R. Streit, "Sufficient conditions for the existence of optimum beam patterns for unequally spaced linear arrays," IEEE Trans. Antennas Propag., vol. AP-23, no. 1, Jan 1975. pp 112-115.

[4] R. Streit, "Optimized symmetric discrete line arrays," IEEE Trans. Antennas Propag., (to appear November, 1975).

[5] G. Meinardus, Approximation of Functions: Theory and Numerical Methods, Springer-Verlag, New York, 1967.

[6] R. Zielke, "Alternation properties of Tchebyshev-Systems and the existence of adjoined functions," J. Approximation Theory, vol. 10, 1974, pp 172-184.

[7] G. Meinardus, "Über Tschebyscheffsche Approximationen," Arch. Rat. Mech. Anal., vol. 9, 1962, pp 329-351.

Concertina-Like Movement In The Absence Of A Chebyshev System

J. T. Lewis and R. L. Streit

Note

Concertina-Like Movement in the Absence of a Chebyshev System

JAMES T. LEWIS*

Mathematics Department, University of Rhode Island, Kingston, Rhode Island 02881, USA

AND

ROY L. STREIT

Naval Underwater Systems Center, New London Laboratory, New London, Connecticut 06320, USA

Communicated by Oved Shisha

Received October 30, 1981; revised February 4, 1982

INTRODUCTION

Meinardus [1, p. 29] defined functions $S(x)$ having certain oscillatory and best approximation properties on an interval $[a, b]$. The most notable example is the Chebyshev polynomial of the first kind, $T_n(x)$. In [2], Streit studied the dependence of $S(x)$ on the left endpoint, a, of the interval and discussed an application to the design of linear antenna arrays. The dependence on the endpoint was further investigated by Zielke [3] who obtained stronger results. We will summarize briefly some of the theory and then present an example to settle a certain question.

PROPERTIES OF $S_t(x)$

Let $[a, b]$ be a finite real interval, n a positive integer and $h_1 = 1, h_2, \ldots, h_n$, f real continuous functions on $[a, b]$ such that $\{1, h_2, \ldots, h_n\}$ is a Chebyshev system of degree n on $[a, b]$ (i.e., $\sum_{i=1}^{n} a_i h_i$ has at most $n-1$ zeros in $[a, b]$ unless $a_1 = 0, \ldots, a_n = 0$). Assume also that $\{1, h_2, \ldots, h_n, f\}$ is a Chebyshev system of degree $n+1$ on $[a, b]$. Let $a \leqslant t < b$ and let $p_t(x)$ denote the best

* The work of this author was performed while he was a summer employee of the Naval Underwater Systems Center, New London, Connecticut, U.S.A.

uniform approximation to $f(x)$ on $[t, b]$ by a linear combination of $1, h_2, \ldots, h_n$. Then [1, p. 29], $f - p_t$ has exactly $n+1$ extremals of alternating sign and equal magnitude which include the endpoints a and b, and $f - p_t$ is a strictly monotone function of x between these extremals. Define

$$S_t(x) = \pm [f(x) - p_t(x)] / \max_{t \leqslant x \leqslant b} |f(x) - p_t(x)|$$

where the sign is chosen so that $S_t(b) = +1$.

If $\{1, h_2, \ldots, h_n, f\}$ is $\{1, x, \ldots, x^n\}$ and $[a, b] = [-1, 1]$, then

$$S_t(x) = T_n\left(\frac{2x}{1-t} - \frac{1+t}{1-t}\right).$$

Motivated by results obtained from the application of the shifted Chebyshev polynomials to linear antenna arrays, Streit [2] studied for the general case the movement of the zeros and extremals of S_t as a function of t. In [4] Zielke showed the entire graph of S_t moves to the right as t increases (concertina-like movement) except possibly the extremal points. They, too, must move to the right if the derivatives $\{h_2', \ldots, h_n', f'\}$ form a Chebyshev system of degree n on (a, b). Of course, the right-hand endpoint of the graph stays fixed at $(b, S_t(b)) = (b, 1)$. We summarize the known properties of S_t: For each t such that $a \leqslant t < b$,

(a) S_t is a linear combination of $1, h_2, \ldots, h_n, f$.

(b) $\max_{t \leqslant x \leqslant b} |S_t(x)| = 1$.

(c) The best uniform approximation to S_t on $[t, b]$ by a linear combination of $\{1, h_2, \ldots, h_n\}$ is 0.

(d) $S_t(x)$ has $n+1$ extremals of alternating sign and equal magnitude, which include the endpoints t and b, and $S_t(x)$ is a strictly monotone function of x between the extremals.

(e) $S_t(b) = 1$.

(f) S_t satisfying (a)–(e) is unique.

(g) The graph of S_t moves to the right as t increases (except for the fixed right-hand endpoint); i.e., $a \leqslant t_1 < t_2 < b$, α in $[-1, 1]$, and $1 \leqslant k \leqslant n$ implies that the smallest z such that $S_{t_1}(x) = \alpha$ for k distinct points in $[t_1, z]$ is strictly less than the smallest z such that $S_{t_2}(x) = \alpha$ for k distinct points in $[t_2, z]$.

The Example

Proof of the existence of S_t with the nice properties (a)–(g) relies heavily on the fact that $\{1, h_2, \ldots, h_n, f\}$ is a Chebyshev system. We were curious as

to whether a system could give rise to an S_t satisfying (a)–(g) without being a Chebyshev system. Clearly this is impossible for $\{1, f\}$ since $c_1 f - c_2$ is strictly monotone between the extremals a and b only if f is (and hence $\{1, f\}$ forms a Chebyshev system). However, we did construct an example $\{1, h_2, f\}$ which we now present.

EXAMPLE. Let $h_2(x) = x$, $f(x) = x^3$ and $[a, b] = [-\frac{1}{2}, 1]$. Then $\{1, x, x^3\}$ is not a Chebyshev system on $[-\frac{1}{2}, 1]$ since, for example, $p(x) = x(x^2 - \frac{1}{16})$ has zeros at $-\frac{1}{4}$, 0, $\frac{1}{4}$. We will now show S_t exists such that properties (a)–(g) are satisfied. Letting $-\frac{1}{2} \leqslant t < 1$, $E_t(x) = x^3 - (a_t + b_t x)$ and using t, x_t, 1 as a reference set gives the equations

$$\begin{aligned} E_t(t) &= t^3 - (a_t + b_t t) = d_t, \\ E_t(x_t) &= x_t^3 - (a_t + b_t x_t) = -d_t, \\ E_t(1) &= 1 - (a_t + b_t) = d_t. \end{aligned} \tag{1}$$

Subtracting the third equation from the first equation gives $t^3 - 1 - b_t(t - 1) = 0$, i.e., $b_t = t^2 + t + 1$. Now

$$\frac{d}{dt} E_t(x) = 3x^2 - b_t = 3x^2 - (t^2 + t + 1) = 0, \qquad \text{when} \quad x = x_t.$$

Hence, $x_t = |(t^2 + t + 1)/3|^{1/2}$. Substituting x_t and b_t into Eqs. (1), one could solve uniquely for a_t and d_t in terms of t and observe that $d_t > 0$; we omit the details. Considering dE_t/dx and using $t \geqslant -\frac{1}{2}$ we see $E_t(x)$ is strictly decreasing in $[t, x_t]$ and strictly increasing in $[x_t, 1]$. Hence, the characterization theorem guarantees that $a_t + b_t x$ obtained from solving (1) is the unique best uniform approximation to x^3 on $[t, 1]$.

Then, for $-\frac{1}{2} \leqslant t < 1$, $S_t(x) = (1/d_t)|x^3 - (a_t + b_t x)|$ satisfies (a)–(f). Now, let $-\frac{1}{2} \leqslant t_1 < t_2 < 1$. Since x_t is strictly increasing as a function of t, $x_{t_1} < x_{t_2}$. Clearly $S_{t_1}(x) - S_{t_2}(x)$ has a zero in (x_{t_1}, x_{t_2}) and a zero at $x = 1$. If $S_{t_1} - S_{t_2}$ has no other zeros in $[t_2, 1]$, then (g) will be satisfied. Assume the opposite; then by Rolle's theorem $d|S_{t_1}(x) - S_{t_2}(x)|/dx$ has at least two zeros, say $z_1 < z_2$, in $(t_2, 1)$ with $z_2 > x_{t_1} \geqslant x_{-1/2} = \frac{1}{2}$. Hence, $z_2 \neq -z_1$ which is impossible since $d|S_{t_1}(x) - S_{t_2}(x)|/dx$ has the form $c_1 x^2 + c_2$. This completes the verification of (a)–(g) for the example.

REFERENCES

1. G. MEINARDUS, "Approximation of Functions: Theory and Numerical Methods," Springer-Verlag, New York, 1967.

2. R. L. STREIT, Extremals and zeros in Markov systems are monotone functions of one end point, *in* "Theory of Approximation with Applications (Proc. Conf., Univ. Calgary, 1975)," pp. 387–401, Academic Press, New York, 1976.
3. R. ZIELKE, Concertina-like movements of the error curve in the alternation theorem, *Manuscripta Math.* **22** (1977), 229–234.

Printed by the St. Catherine Press Ltd., Tempelhof 41, Bruges, Belgium

Limits of Chebyshev Polynomials When The Argument Is A Ratio of Cosines

R. L. Streit

Note

Limits of Chebyshev Polynomials When the Argument Is a Ratio of Cosines

ROY L. STREIT

Naval Underwater Systems Center,
New London Laboratory, New London, Connecticut 06320, U.S.A.

Communicated by Oved Shisha

Received January 25, 1983

Two new limits involving Chebyshev polynomials of the first and second kinds are given. These limits are useful in certain engineering applications. The proofs are based on the Mehler–Heine theorem for Jacobi polynomials.

Let $P_n^{(\alpha,\beta)}(x)$ denote the Jacobi polynomials. It is evident from the representation [1, (4.21.2)]

$$P_n^{(\alpha,\beta)}(x)=\frac{1}{n!}\sum_{\nu=0}^{n}\binom{n}{\nu}(n+\alpha+\beta+1)_\nu(\alpha+\nu+1)_{n-\nu}\left(\frac{x-1}{2}\right)^\nu \quad (1)$$

that $P_n^{(\alpha,\beta)}(x)$ is a polynomial of degree n in x and in the parameters α and β. Hence, $P_n^{(\alpha,\beta)}(x)$ can be extended to all complex values of α, β, and x. In this note, α and β are restricted to be real numbers.

For any complex number x, the Mehler–Heine theorem [1, Theorem 8.1.1] states that

$$\lim_{n\to\infty} n^{-\alpha}P_n^{(\alpha,\beta)}\left(\cos\frac{x}{n}\right)=(x/2)^{-\alpha}J_\alpha(x), \quad (2)$$

where $J_\alpha(x)$ is the Bessel function of the first kind of order α [1, (1.71.1)]. Szegö's proof of (2) actually establishes that

$$\lim_{n\to\infty} n^{-\alpha}P_n^{(\alpha,\beta)}\left(1-\frac{x^2}{2n^2}+o(n^{-2})\right)=(x/2)^{-\alpha}J_\alpha(x).$$

Consequently, for all complex x and y,

$$\lim_{n\to\infty} n^{-\alpha} P_n^{(\alpha,\beta)}\left(\frac{\cos(x/n)}{\cos(y/n)}\right) = \lim_{n\to\infty} n^{-\alpha} P_n^{(\alpha,\beta)}\left(1 - \frac{x^2-y^2}{2n^2} + o(n^{-2})\right)$$

$$= \left(\frac{1}{2}\sqrt{x^2-y^2}\right)^{-\alpha} J_\alpha(\sqrt{x^2-y^2}). \tag{3}$$

Like the Mehler–Heine theorem, this result holds uniformly for x and y in every bounded region of the complex plane.

The limit (3) has interesting special forms for the Chebyshev polynomials $T_n(z)$ and $U_n(z)$ of the first and second kinds, respectively. Substituting the identities

$$P_n^{(-1/2,-1/2)}(z) = \frac{(2n)!}{2^{2n}(n!)^2} T_n(z), \qquad n \geqslant 1,$$

and

$$J_{-1/2}(z) = \left(\frac{2}{\pi z}\right)^{1/2} \cos z$$

in (3) and applying Stirling's formula gives

$$\lim_{n\to\infty} T_n\left(\frac{\cos(x/n)}{\cos(y/n)}\right) = \cos\sqrt{x^2-y^2}. \tag{4}$$

Similarly, substituting

$$P_n^{(1/2,1/2)}(z) = \frac{(2n+2)!}{2^{2n+1}((n+1)!)^2} U_n(z), \qquad n \geqslant 0,$$

and

$$J_{1/2}(z) = \left(\frac{2}{\pi z}\right)^{1/2} \sin z$$

in (3) and applying Stirling's formula gives

$$\lim_{n\to\infty} n^{-1} U_n\left(\frac{\cos(x/n)}{\cos(y/n)}\right) = \frac{\sin\sqrt{x^2-y^2}}{\sqrt{x^2-y^2}}. \tag{5}$$

These limiting forms do not seem to be mentioned elsewhere in the literature.

A result similar to (4) is used implicitly in an antenna design application [2]. The result (5) is shown in [3] to be intimately related to the so-called

Kaiser–Bessel window in digital filter design. These applications require knowledge of the cosine transform of the right-hand side of (3), which is provided by a special case of Sonine's second finite integral [4, p. 376] for $\alpha > -\frac{1}{2}$. In particular,

$$\lim_{n\to\infty} n^{-1} U_n\left(\frac{\cos(x/n)}{\cos(y/n)}\right) = \int_0^1 I_0(y\sqrt{1-\zeta^2})\cos x\zeta\, d\zeta, \tag{6}$$

where $I_\nu(z)$ is the modified Bessel function of order ν. Sonine's second finite integral diverges for $\alpha = -\frac{1}{2}$; however, the cosine transform of $\cos((x^2-y^2)^{1/2})$ is known [5, (871.2)], so that

$$\lim_{n\to\infty} T_n\left(\frac{\cos(x/n)}{\cos(y/n)}\right) = \cos x + y\int_0^1 \frac{I_1(y\sqrt{1-\zeta^2})}{\sqrt{1-\zeta^2}}\cos x\zeta\, d\zeta. \tag{7}$$

It is evident from (6) and (7) that the limiting forms have finite support (i.e., are bandlimited) and, thus, are of exponential type.

An extremal property of $\cos((x^2-y^2)^{1/2})$ in the space of functions of exponential type is given in [6]. The proof is based on a theorem in [7]. Whether or not the limit function (3) has extremal properties in this space is not known to the author.

REFERENCES

1. G. SZEGÖ, "Orthogonal Polynomials," 4th ed., Vol. 23, *Amer. Math. Soc. Colloq. Publ.*, 1978.
2. G. J. VAN DER MAAS, A simplified calculation for Dolph–Tchebycheff arrays, *J. Appl. Phys.* **25** (1) (1954).
3. R. L. STREIT, A two parameter family of weights for nonrecursive digital filters and antennas, *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-32**, 108–118.
4. G. N. WATSON, "Theory of Bessel Functions," 2nd ed., Cambridge Univ. Press, London/New York, 1966.
5. G. A. CAMPBELL AND R. M. FOSTER, "Fourier Transforms for Practical Applications," Van Nostrand, Princeton, N. J., 1948.
6. V. BARCILON AND G. C. TEMES, Optimum impulse response and the van der Maas function, *IEEE Trans. Circuit Theory* **CT-19** (4) (1972), 336–342.
7. R. J. DUFFIN AND A. C. SCHAEFFER, Some properties of functions of exponential type, *Bull. Amer. Math. Soc.* **44** (1938), 236–240.

Printed by the St. Catherine Press Ltd., Tempelhof 41, Bruges, Belgium

A Routine For Numerical Solution Of Fredholm Integral Equations

R. L. Streit and A. H. Nuttall

Abstract

A routine is presented for numerical solution of Fredholm integral equations of the first and second kind and for solution of characteristic values and functions. An approximate method for solving integral equations of the first kind by means of integral equations of the second kind is presented and illustrated with several examples. Application to simultaneous determination of several characteristic values and functions of a kernel is also considered and documented by several examples.

INTRODUCTION

The interest here lies in obtaining approximations to the solution g(x) of the Fredholm integral equation

$$\int_a^b dy\, K(x,y)\, g(y) = \lambda g(x) + f(x), \quad a < x < b, \tag{1}$$

where the kernel $K(x,y)$ and function $f(x)$ are known. If λ and $f(x)$ are non-zero, (1) is an integral equation of the second kind. If λ is zero, (1) is an integral equation of the first kind. If $f(x)$ is zero, (1) is a characteristic value problem, possessing solutions only for certain values of λ ; in this case, we express (1) as

$$\int_a^b dy\, K(x,y)\, \phi_n(y) = \lambda_n \phi_n(x), \quad a < x < b, \tag{2}$$

where $\{\lambda_n\}$ are the characteristic values of the kernel and $\{\phi_n(x)\}$ are the characteristic functions (assumed to be of unit energy: $\int_a^b dx\, \phi_n^2(x) = 1$).

PROBLEM SOLUTION

The integral on the left side of (1) is approximated by any of numerous integration rules available, such as the Trapezoidal rule, Simpson's rule, Gauss quadrature, etc. That is,

$$\int_a^b dy\, K(x,y)\, g(y) \cong \sum_{j=1}^{N} w_j\, K(x, \xi_j)\, g(\xi_j), \quad a < x < b, \tag{3}$$

where $\{w_j\}$ are the weights and $\{\xi_j\}$ are the abscissas (points) of the particular integration rule adopted. The approximation to the true solution $g(x)$ of (1) is denoted by $\hat{g}(\xi_j)$ at the point ξ_j . This approximation is obtained by utilizing (3) in (1) and requiring that

$$\sum_{j=1}^{N} w_j\, K(\xi_i, \xi_j)\, \hat{g}(\xi_j) = \lambda\, \hat{g}(\xi_i) + f(\xi_i), \quad 1 \le i \le N. \tag{4}$$

This method is known as the method of collocation. Equation (4) constitutes N equations in the N unknowns $\{\hat{g}(\tau_i)\}_1^N$.

We define four matrices F, G, B, and D as

$$F^T = [f(\tau_1) \cdots f(\tau_N)]$$
$$\hat{G}^T = [\hat{g}(\tau_1) \cdots \hat{g}(\tau_N)]$$
$$B = [K(\tau_i, \tau_j)] \quad (N \times N)$$
$$D = [w_j \delta_{ij}] \quad (N \times N). \tag{5}$$

Equation (4) then takes the form

$$BD\hat{G} = \lambda \hat{G} + F, \tag{6}$$

with solution

$$\hat{G} = (BD - \lambda I)^{-1} F, \tag{7}$$

where I is the identity matrix. (It should be noted that BD is not necessarily symmetric, even if B is symmetric.) If $\{\hat{\lambda}_n\}$ are the characteristic numbers of matrix BD, then

$$|BD - \hat{\lambda}_n I| = 0, \quad 1 \le n \le N. \tag{8}$$

Therefore (7) possesses a solution if $\lambda \neq \hat{\lambda}_n$, $1 \le n \le N$.

The general approximate solution to the integral equation of the second kind is afforded by (7) and is considered in the next section. Solution to the integral equation of the first kind ($\lambda = 0$ in (1)) is considered in the succeeding section. The characteristic value problem ($f(t) = 0$ in (1)) is considered last.

INTEGRAL EQUATIONS OF THE SECOND KIND

The Fredholm Alternative guarantees that if the kernel $K(x,y)$ of (1) is square-integrable, then there exists a unique solution to (1) if λ is not a characteristic value. Since λ is assumed not to be a characteristic value in this section, the existence and uniqueness of solutions to (1) are established and are of no concern here. This is not the case for Fredholm integral equations of the first kind.

The numerical solution (7) of Fredholm integral equations of the second kind is an excellent approximation to the exact solution, if the kernel $K(x,y)$ is smooth enough to permit a good approximation to the integral, as in (3), by some quadrature formula. However, even if the kernel is discontinuous, the approximation to the exact solution $g(x)$ can still be quite good. These rather vague remarks are best explained by examples.

Example 1

$$\int_0^1 dy \, \frac{1}{(1+xy)^3} g(y) = g(x) + \frac{1}{2(1+x)^2} - x, \quad 0 < x < 1. \tag{9A}$$

The exact solution is $g(x) = x$. When a four-point Gaussian quadrature formula is employed, the computed solution $\hat{g}(t_i)$ at the points t_i is correct to four significant places, as shown in Table 1; when a 20-point Gaussian quadrature formula is used, the computed solution agrees to 15 significant places with the exact solution.

Table 1

EXACT AND COMPUTED SOLUTIONS OF (9A); N = 4

i	$g(t_i)$	$\hat{g}(t_i)$
1	.0694318	.0694366
2	.3300095	.3300128
3	.6699905	.6699952
4	.9305682	.9305794

Example 2

$$\int_0^1 dy\, K(x,y)\, g(y) = g(x) - 1, \quad 0 < x < 1, \tag{9B}$$

where

$$K(x,y) = \left\{ \begin{matrix} 1, & x \geq y \\ 0, & x < y \end{matrix} \right\}.$$

(This example can also be interpreted as a Volterra integral equation.) The exact solution is $g(x) = e^x$. Using a 10-point Gaussian quadrature formula, despite the discontinuous kernel, yields the reasonably good solution $\hat{g}(s_i)$ of Table 2.

Table 2

EXACT AND COMPUTED SOLUTIONS OF (9B); N = 10

s_i	$\hat{g}(s_i)$	$g(s_i)$
.01305	1.034	1.013
.06747	1.118	1.070
.1603	1.256	1.174
.2833	1.451	1.328
.4256	1.702	1.530
.5744	1.998	1.776
.7167	2.308	2.048
.8397	2.592	2.316
.9325	2.802	2.541
.9869	2.898	2.683

In most cases, however, the exact solution is unknown and it might be thought that one way to determine whether or not the numerical solutions are accurate is to increase the order of the quadrature formula and solve the system (7) again. For Fredholm integral equations of the second kind, this process is feasible since the system (7) is well-conditioned, even for high-order quadrature formulae. The explanation for this phenomenon is that if λ is not close to a characteristic value of matrix BD, then matrix $BD - \lambda I$ is well-conditioned. Matrix BD itself can be ill-conditioned (as seen in the next section).

Once the approximate solution $\hat{g}(s_i)$ is known at a sufficient number of points, any form of interpolating function may be used to connect these points. It is worthwhile to note, however, that a ready-made interpolation procedure is already at hand once (7) has been solved. Namely, solve for $g(x)$ on the right hand side of (1), and approximate the integral by (3), to obtain the interpolated approximation

$$\hat{g}(x) = \frac{1}{\lambda}\sum_{j=1}^{N} w_j K(x, s_j)\hat{g}(s_j) - \frac{1}{\lambda} f(x), \quad a < x < b, \tag{10}$$

which automatically goes through the points $\hat{g}(s_i)$ at $x = s_i$.

INTEGRAL EQUATIONS OF THE FIRST KIND

The Fredholm integral equation of the first kind may be obtained from (1) by putting $\lambda = 0$:

$$\int_a^b dy\ K(x,y)\, g(y) = f(x), \quad a < x < b. \tag{11}$$

When this equation is viewed as an operator on g, it is readily noticed that discontinuous functions g are taken into continuous functions f. More specifically, define $C^0[a,b]$ to be the space of all functions g such that $\int_a^b dx\,|g(x)|$ exists and is finite, and $C^n[a,b]$, $n \geq 1$, to be the space of all functions possessing at least n derivatives. Then equation (11) may be said to define an operator taking functions in $C^0[a,b]$ into functions in $C^n[a,b]$, where n is an integer such that $K(x,y)$ possesses the n^{th} partial derivative with respect of x, since we have by a well-known theorem

$$\int_a^b dy\ \frac{\partial^n}{\partial x^n}\left[K(x,y)\right] g(y) = f^{(n)}(x).$$

In other words, the more partial derivatives the kernel $K(x,y)$ has with respect to x, the "smoother" the function $f(x)$ in (11) must be. Stated a third way, since $C^n[a,b]$ is a subspace of $C^0[a,b]$, any attempted inversion, numerical or otherwise, of equation (11) can pose considerable problems. In each particular integral equation of the first kind, such fundamental questions as existence and uniqueness of solutions are open to question.

If one attempts a straightforward application of the method of collocation by putting $\lambda=0$ in equation (7), then one discovers that the matrix BD is often badly ill-conditioned, even for low-order quadrature formulae. Consider the following example.

Example 3

$$\int_0^1 dy\, \cos(\pi xy)\, g(y) = \frac{\sin(\pi x)}{\pi x}, \quad 0 < x < 1. \tag{12}$$

Here, a solution is given by $g(x)=1$ for $0<x<1$. Gaussian quadrature of orders 5, 6, and 7 all give 6 or 7 correct signfificant digits for the approximate solution to (12), but for orders 8, 9, 10 and higher, the accuracy rapidly deteriorates, as shown in Table 3. For $N \geq 11$, the results get progressively worse, because the matrix BD becomes more ill-conditioned.

Table 3

COMPUTED SOLUTIONS OF (12); N = 8, 9, 10.

N = 8		N = 9		N = 10	
ξ_i	$\hat{g}(\xi_i)$	ξ_i	$\hat{g}(\xi_i)$	ξ_i	$\hat{g}(\xi_i)$
.01986	.99917	.01592	1.25796	.01305	5.70302
.10167	1.00053	.08198	.84076	.06747	-1.84768
.23723	.99987	.19331	1.03748	.16029	1.62953
.40828	1.00003	.33787	.99216	.28330	.88319
.59172	.99999	.50000	1.00185	.42556	1.02308
.76277	1.00000	.66213	.99946	.57444	.99467
.89833	1.00000	.80669	1.00020	.71669	1.00148
.98014	1.00000	.91802	.99991	.83970	.99950
		.98408	1.00004	.93253	1.00021
				.98695	.99991

This example and several others give convincing evidence that the method of collocation is not a particularly good method to use directly on integral equations of the first kind, because of the difficulty of establishing the correctness

of the results, even if a unique solution is known to exist. No recourse is possible to higher order quadrature formulae here, as in the case of integral equations of the second kind, since the matrix BD becomes progressively more ill-conditioned.

Instead of putting $\lambda = 0$ in equation (1) to obtain the integral equation of the first kind immediately, why not let λ approach 0 in steps, thus successively approximating integral equations of the first kind with integral equations of the second kind? It might then be expected that the solutions to the integral equations of the second kind approach some solution of the integral equation of the first kind, provided of course that the integral equation of the first kind possesses solutions at all.

More specifically, suppose the sequence $\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3, \ldots$ converges to 0, and that none of the $\tilde{\lambda}_\kappa$ are characteristic values $\hat{\lambda}_i$ of the matrix BD. (It has been found computationally that little care is needed in choosing this sequence; however, if there is any cause for concern, it is certainly possible to compute the characteristic values $\hat{\lambda}_i$ of the matrix BD and choose the $\tilde{\lambda}_\kappa$ to be midway between adjacent characteristic values.) Let $g_\kappa(x)$ be the solution of

$$\int_a^b dy\, K(x,y)\, g_\kappa(y) = \tilde{\lambda}_\kappa\, g_\kappa(x) + f(x), \quad a < x < b. \tag{13}$$

It is believed that $g_\kappa(x)$ converges uniformly to some bounded solution $g(x)$ of (11) if any such solutions of (11) exist. Unfortunately, the only proof of this statement presented herein is computational in nature.

The next five examples are intended to be a good sample of situations that actually arise in the application of this method.

Example 4

$$\int_0^1 dy\, \left(x - \frac{1}{1+y}\right) g(y) = \frac{3}{2}x^2 - 1, \quad 0 < x < 1. \tag{14}$$

Since the integral on the left side of (14) must always yield an expression of the form $\alpha x + \beta$ for any function $g(y)$, there does not exist a solution to this equation. However, the equation

$$\int_0^1 dy \left(x - \frac{1}{1+y}\right) g_k(y) = \tilde{\lambda}_k g_k(x) + \frac{3}{2}x^2 - 1, \quad 0 < x < 1, \qquad (15)$$

certainly has a unique solution $g_k(x)$ for each non-characteristic value of $\tilde{\lambda}_k$, and this solution must obviously be a quadratic function of its argument x. Table 4 shows the results of solving (15) for $\tilde{\lambda}_1 = -10^{-4}$, $\tilde{\lambda}_2 = -10^{-8}$, and $\tilde{\lambda}_3 = 0$, using a 6-point Gaussian quadrature formula. On the basis of the information in Table 4, one is led to the (correct) conclusion that there do not exist any bounded solutions of (14).

Table 4

COMPUTED SOLUTIONS OF (14); N = 6

ξ_i	$\hat{g}_1(\xi_i)$	$\hat{g}_2(\xi_i)$	$\hat{g}_3(\xi_i)$
.033765	.169443E4*	.169037E8	- .190339E19
.169395	.165699E3	.162407E7	.396491E18
.380690	- .111633E4	- .111843E8	.858363E18
.619309	-.953767E3	- .954527E7	.162675E19
.830605	.616148E3	.616577E7	- .511557E19
.966235	.232968E4	.233087E8	.505322E19

Example 5

$$\int_0^1 dy \left(x - \frac{1}{1+y}\right) g(y) = \frac{3}{2}x - 1, \quad 0 < x < 1. \qquad (16)$$

Here, we must obviously have $\int_0^1 dy\, g(y) = \frac{3}{2}$ and $\int_0^1 dy \frac{g(y)}{1+y} = 1$. An example is $g_p(y) = 1+y$. Since any function orthogonal to both 1 and $\frac{1}{1+y}$ may be added to the particular solution $g_p(y) = 1+y$, the solution to (16) is not unique. Choosing $\tilde{\lambda}_1 = -10^{-8}$, $\tilde{\lambda}_2 = -10^{-10}$, and $\tilde{\lambda}_3 = 0$, gives the results of Table 5, when Gaussian quadrature with only 5 points is used. The results for $\hat{g}_1$ and $\hat{g}_2$ are excellent. If higher-order Gaussian

*E4 denotes a multiplicative factor of 10^4.

Table 5

COMPUTED SOLUTIONS OF (16); N = 5

ξ_i	$\hat{g}_1(\xi_i)$	$\hat{g}_2(\xi_i)$	$\hat{g}_3(\xi_i)$
.04691008	1.0469103	1.0469101	-41.338
.23076534	1.2307654	1.2307653	33.503
.50000000	1.4999999	1.5000000	1.000
.76923466	1.7692345	1.7692347	- 4.000
.95308992	1.9530897	1.9530899	- 8.000

formulae are used, even better results may be obtained. It is easily seen that the solutions are converging to the particular solution $g_p(y) = 1+y$ (when $\tilde{\lambda}_k \neq 0$).

Example 6

$$\int_0^1 dy\, \cos(\pi x y)\, g(y) = \frac{\sin(2x)}{\pi x}, \quad 0 < x < 1. \tag{17}$$

Here there exists a solution with a simple discontinutity:

$$g(x) = \left\{ \begin{array}{ll} 1, & 0 < x < \frac{2}{\pi} \cong .636 \\ 0, & \frac{2}{\pi} < x < 1 \end{array} \right\}.$$

Using an 11-point Gaussian quadrature formula and putting $\tilde{\lambda}_1 = -10^{-6}$, $\tilde{\lambda}_2 = -10^{-9}$, and $\tilde{\lambda}_3 = -10^{-19}$, we get the solutions of Table 6. These solutions are not better because the integrand is, in fact, not continuous so that the quadrature is not particularly good.

Table 6

COMPUTED SOLUTIONS OF (17); N = 11

ξ_i	$\hat{g}_1(\xi_i)$	$\hat{g}_2(\xi_i)$	$\hat{g}_3(\xi_i)$
.01089	.91588	1.0159	1.0091
.05647	.92752	1.0095	1.0087
.13492	.97852	.9866	1.0028
.24045	1.06966	.9786	.9784
.36523	1.10212	1.0427	1.0045
.50000	.91608	1.0075	1.0526
.63477	.49963	.5399	.5352
.75955	.08230	-.0399	-.0997
.86508	-.09775	-.0649	.0519
.94353	-.03132	.1090	-.0315
.98911	.10835	-.0955	.0182

Example 7

$$\int_0^1 dy\ \cos(\pi xy)\, g(y) = \cos(x),\quad 0 < x < 1. \tag{18}$$

Here there exists at least one solution with a very bad kind of discontinuity, namely, the Dirac delta function, $g(x) = \delta(x - 1/\pi)$. With $\tilde{\lambda}_1 = -10^{-5}$, $\tilde{\lambda}_2 = -10^{-9}$, and $\tilde{\lambda}_3 = -10^{-15}$, and with 12-point Gaussian quadrature, a solution resembling a continuous approximation of $g(x)$ does indeed show up; see Table 7. Considering the numerical difficulty of this problem, the solutions shown in Table 7 should be considered as good results. Better solutions may be had with higher-order quadrature formulae.

Table 7

COMPUTED SOLUTIONS OF (18); N = 12

s_i	$\hat{g}_1(s_i)$	$\hat{g}_2(s_i)$	$\hat{g}_3(s_i)$
.911E-2	.737	-1.376	-1.399
.479E-1	.805	-1.069	-1.149
.115	1.120	.297	.059
.206	1.795	2.871	2.853
.316	2.509	4.553	5.213
.437	2.475	2.737	2.404
.563	1.274	- .568	-1.267
.684	- .258	- .748	.582
.794	- .779	.798	- .224
.885	- .178	- .004	.057
.952	.408	- .829	.014
.991	.397	.995	- .024

Example 8

$$\int_0^1 dy \cos(\pi x y)\, g(y) = \frac{\sin(\pi x)}{\pi x}, \quad 0 < x < 1. \tag{19}$$

This equation was treated in Example 3 in a straightforward manner, with poor results. Here we put $\tilde{\lambda}_1 = -10^{-3}$, $\tilde{\lambda}_2 = -10^{-6}$, and $\tilde{\lambda}_3 = -10^{-10}$, with a 10-point Gaussian quadrature formula. Whereas the solution before with N = 10 was very poor, we now have the good results of Table 8. These solutions clearly converge to the solution $g(x) = 1$.

Table 8

COMPUTED SOLUTIONS OF (19); N = 10

ξ_i	$\hat{g}_1(\xi_i)$	$\hat{g}_2(\xi_i)$	$\hat{g}_3(\xi_i)$
.01304 67357	.99739 04370	.99999 40301	1.00000 00067
.06746 83167	.99736 45162	.99999 44020	.99999 99954
.16029 52159	.99727 35342	.99999 60232	.99999 99962
.28330 23029	.99727 36762	.99999 90304	1.00000 00072
.42556 28305	.99795 80272	1.00000 12796	.99999 99941
.57443 71695	.99981 58041	1.00000 05868	1.00000 00028
.71669 76971	1.00192 20787	.99999 90874	.99999 99998
.83970 47841	1.00204 72316	.99999 99732	.99999 99987
.93253 16833	.99923 61422	1.00000 08581	1.00000 00019
.98695 32643	.99565 76614	.99999 90819	.99999 99985

Thus, it is seen that the proposed method is a very good one. It can be indicative of the nonexistence of exact solutions, or of the existence of "spikes" within an exact solution. If the exact solution contains simple discontinuities, the computed solution can give an acceptable indication of this fact. The method is best, of course, when the solutions are all continuous. Its most interesting feature is its ability to yield a particular solution even when the solution to the integral equation is not unique.

CHARACTERISTIC VALUE PROBLEMS

When $f(x)$ is zero in (1), the integral equation takes the form of (2). The general approximate solution* $\hat{G}$ in (7) then will be identically zero unless

$$\lambda = \hat{\lambda}_1 \quad \text{or} \quad \lambda = \hat{\lambda}_2 \quad \text{or} \cdots \text{or} \quad \lambda = \hat{\lambda}_N , \tag{20}$$

where $\{\hat{\lambda}_n\}$ are the characteristic values of matrix BD. Each characteristic value $\hat{\lambda}_n$ $(1 \le n \le N)$ of matrix BD is an approximation to the first N characteristic values $\{\lambda_n\}$ of kernel $K(x,y)$, (the lower-order ones being better approximations). For $\lambda = \hat{\lambda}_n$, (6) becomes (since F = 0 here)

$$BD\,\hat{G}_n = \hat{\lambda}_n \hat{G}_n , \quad 1 \le n \le N. \tag{21}$$

*This method is also suggested in Ref. 1, p. 447.

Thus $\hat{\mathfrak{G}}_n$ is proportional to the normalized characteristic vector of matrix BD, corresponding to characteristic value $\hat{\lambda}_n$. Scaled versions of $\hat{\mathfrak{G}}_n$ are approximations to the characteristic functions $\phi_n(x)$ of kernel $K(x,y)$. That is,

$$\phi_n(\xi_i) \cong c_n \hat{g}_n(\xi_i), \quad 1 \le i \le N, \tag{22}$$

for $1 \le n \le N$. Thus matrix BD contains all the information necessary for simultaneous approximate determination of N characteristic values and functions of kernel $K(x,y)$.

The method above yields samples of the approximate solution $\hat{g}_n(x)$ only at the points $x = \xi_i, 1 \le i \le N$. Evaluation of $\hat{g}_n(x)$ could be accomplished by polynomial interpolation, for example, or by employing (3) in (1) to obtain

$$\hat{g}_n(x) \equiv \frac{1}{\hat{\lambda}_n} \sum_{j=1}^{N} w_j K(x,\xi_j) \hat{g}_n(\xi_j). \tag{23}$$

The approximation $\hat{g}_n(x)$ obtained from the right side of (23) automatically goes through the values $\hat{g}_n(\xi_i), 1 \le i \le N$; see (4).

In order to obtain an approximation to the characteristic function $\phi_n(x)$ in (2), we let the approximation be

$$\hat{\phi}_n(x) = c_n \hat{g}_n(x), \tag{24}$$

and choose c_n such that $\hat{\phi}_n(x)$ has unit energy. That is,

$$1 = \int_a^b dx\, \hat{\phi}_n^2(x) = c_n^2 \int_a^b dx\, \hat{g}_n^2(x) \cong c_n^2 \sum_{j=1}^{N} w_j \hat{g}_n^2(\xi_j). \tag{25}$$

Solving for c_n and substituting in (24) and (23), there follows

$$\hat{\phi}_n(x) \cong \frac{\sum_{j=1}^{N} w_j K(x,\xi_j) \hat{g}_n(\xi_j)}{\hat{\lambda}_n \left[\sum_{j=1}^{N} w_j \hat{g}_n^2(\xi_j)\right]^{1/2}}. \tag{26}$$

The right side of (26) is an interpolated approximation to the n^{th} characteristic function $\phi_n(x)$ of kernel $K(x,y)$.

Even and Odd Solutions

Before embarking on examples, it should be pointed out that advantage should be taken of any symmetry properties of the kernel $K(x,y)$. Suppose the integral equation is of the form

$$\int_{-a}^{a} dy\, K(x,y)g(y) = \lambda g(x), \quad -a < x < a. \tag{27}$$

Now suppose that the kernel satisfies

$$K(-x,y) = K(x,y). \tag{28}$$

Then

$$\lambda g(-x) = \int_{-a}^{a} dy\, K(-x,y)g(y) = \int_{-a}^{a} dy\, K(x,y)g(y) = \lambda g(x). \tag{29}$$

Therefore $g(x)$ must be even. Then letting $t = -y$, and using the evenness of $g(x)$,

$$\int_{-a}^{0} dy\, K(x,y)\, g(y) = \int_{0}^{a} dt\, K(x,-t)\, g(t). \tag{30}$$

Therefore (27) becomes

$$\int_{0}^{a} dy\, \left[K(x,y) + K(x,-y)\right] g(y) = \lambda g(x), \quad 0 < x < a, \tag{31}$$

and the interval has been cut in half. This is very advantageous for numerical computation, since either fewer computations need be carried out, or increased accuracy for the same number of computations can be realized.

For example, for the kernel

$$K(x,y) = \frac{1}{A + B\cos(x-y)} + \frac{1}{A + B\cos(x+y)}, \tag{32}$$

(28) is sotisfied, leading to even $g(x)$. However, for the kernel

$$K(x,y) = \frac{1}{A + B\cos(x-y)}, \tag{33}$$

(28) is not satisfied, and $g(x)$ need not be even.

Conversely if

$$K(-x,y) = -K(x,y), \tag{34}$$

then $g(x)$ must be odd; again, the interval can be cut in half in a manner similar to (31).

Example 9

$$K(x,y) = \frac{1}{1+h^2-2h\cos[\pi(x-y)]} + \frac{1}{1+h^2-2h\cos[\pi(x+y)]}, \quad 0<x,y<1. \tag{35}$$

Since

$$K(x,y) = \frac{2}{1-h^2}\left[1 + \sum_{n=1}^{\infty} h^n \sqrt{2}\cos(n\pi x)\sqrt{2}\cos(n\pi y)\right], \quad |h|<1, \tag{36}$$

we have

$$\lambda_n = \frac{2}{1-h^2}h^n, \quad n \geq 0;$$

$$\phi_0(x) = 1; \quad \phi_n(x) = \sqrt{2}\cos(n\pi x), \quad n \geq 1. \tag{37}$$

The first integration rule adopted in (3) is the Trapezoidal rule. For $h=0.5$, the approximations to the characteristic values for $N=9$ and 20 are listed in Table 9.

Table 9

CHARACTERISTIC VALUES VIA TRAPEZOIDAL RULE; h = 0.5

N = 9	N = 20	Exact
2.666675	2.666666	2.666667
1.333435	1.333333	1.333333
0.666839	0.666666	0.666667
0.333664	0.333333	0.333333
0.167320	0.166667	0.166667
0.084637	0.083333	0.083333
0.044271	0.041667	0.041667
0.026042	0.020833	0.020833
0.020834	0.010417	0.010417

The approximations up through λ_8 for $N=20$ are seen to be identical to the exact values. Those for $N=9$ are less accurate.

All the approximations to the characteristic functions, for $N=9$, were found to be accurate to at least six decimal places at the points $x=s_i$, even for $\phi_8(x)$. Thus, far better accuracy was obtained in estimating the characteristic functions than in estimating the characteristic values, for the kernel of (35).

When the interpolated approximation (26) was compared with the exact characteristic function (37), the worst ratio of answers was 0.9982. Thus, the interpolated function is not as accurate as the sample values at $\{s_i\}$.

When h was increased to 0.9, the approximate characteristic numbers were less accurate for $N = 20$, as depicted in Table 10. This is due to the fact that the kernel of (35) now has a large maximum-to-minimum ratio, and the approximation

Table 10

CHARACTERISTIC VALUES VIA TRAPEZOIDAL RULE; h = 0.9

N = 20	Exact
10.918	10.526
9.867	9.474
8.926	8.526
8.085	7.674

of (3) is not adequate. A larger value of N is needed in this case. (In practice, two different values of N would be tried, and increased until substantially the same results obtained.) However, it was again found that good approximations to the characteristic functions were obtained. In fact, at least four decimals were retained up through $\phi_{19}(x)$, at the points $\{s_i\}$. However, the interpolated approximation was far less accurate, being about 10% in error.

When the integration rule was that of Gauss quadrature, the results of Table 11 were obtained, for h = 0.5. These results are not as good as those

Table 11

CHARACTERISTIC VALUES VIA GAUSS QUADRATURE; h = 0.5

N = 9	Exact
2.667186	2.666667
1.333653	1.333333
0.667836	0.666667
0.335439	0.333333
0.172508	0.166667

obtained via the Trapezoidal rule. It is believed that the reason for this behavior is due to the fact that kernel $K(x,y)$ is even about $y = 0$ and $y = 1$, for any x. And since $\phi_n(y)$ is also even about $y = 0$ and $y = 1$, the Trapezoidal rule with end correction (Ref. 2) reduces to the simple case adopted here. Then since we are integrating over a period of a periodic function, very accurate results are possible via the Trapezoidal rule (see Ref. 2, sect. 2.9). For more general kernels, this behavior need not be expected. (However, another reason for choosing the Trapezoidal rule will be presented later in Example 11.) The accuracy in the characteristic functions obtained via Gauss quadrature was of comparable accuracy to that obtained for the characteristic values.

Example 10

$$K(x,y) = \frac{1}{1+h^2-2h\cos[\pi(x-y)]}, \quad -1 < x, y < 1. \tag{38}$$

Since

$$K(x,y) = \frac{2}{1-h^2}\left[\frac{1}{2} + \sum_{n=1}^{\infty} h^n\left\{\cos(n\pi x)\cos(n\pi y) + \sin(n\pi x)\sin(n\pi y)\right\}\right], \tag{39}$$

we have

$$\{\lambda_n\} = \frac{2}{1-h^2}\left\{1, h, h, h^2, h^2, \ldots\right\},$$

$$\{\phi_n(x)\} = \left\{\frac{1}{\sqrt{2}}, \cos(\pi x), \sin(\pi x), \cos(2\pi x), \sin(2\pi x), \ldots\right\}. \qquad (40)$$

Thus, double roots occur for the characteristic values for this kernel.

For the Trapezoidal rule of integration, and for $h = 0.5$, the results of the approximate technique are given in Table 12. There is no problem in evaluating the double roots, nor in obtaining two linearly independent characteristic vectors of the matrix BD. Again, very good accuracy in characteristic function approximation at the points $\{\xi_i\}$ was obtained.

Table 12

CHARACTERISTIC VALUES VIA TRAPEZOIDAL RULE; $h = 0.5$

N = 20	Exact
2.666676	2.666667
1.333346	1.333333
1.333335	1.333333
0.666688	0.666667
0.666688	0.666667
0.333375	0.333333
0.333374	0.333333

Example 11

$$K(x,y) = \exp(-|x-y|), \quad 0 \le x, y \le 1. \qquad (41)$$

The exact characteristic values and functions are given in Ref. 3, p. 116. The results of applying the Trapezoidal rule are given in Table 13 for N = 9, 20, and 50. The relatively poorer accuracy obtained in this example is due to the cusp in the kernel (41) at $x = y$.

Attempts to apply either Simpson's rule or Gauss quadrature to this example lead to poorer results than use of the Trapezoidal rule. The reason for this is that, for maximum accuracy, the integral in (2) should be broken into the ranges (0, x) and (x, 1), and the integrand approximated in each

Table 13

CHARACTERISTIC VALUES VIA TRAPEZOIDAL RULE

N=9	N=20	N=50	Exact
0.738619	0.738777	0.738806	0.738811
0.139433	0.138257	0.138042	0.138004
0.047308	0.045474	0.045146	0.045088
0.023878	0.021757	0.021393	0.021329
0.015078	0.012728	0.012345	0.012279

region. However, for Gauss quadrature, since x must be selected as the points $\{\xi_i\}$, the integrals over $(0,\xi_i)$ and $(\xi_i,1)$ do not themselves yield to Gauss quadrature without additional different sampling plans within these intervals. For equi-spaced choices of x (as for the Simpson and Trapezoidal rules), the parabolic approximation of Simpson's rule cannot be applied at certain points. For example, when $x = \xi_2 = \frac{1}{N-1}$, there is only one panel in the interval (0,x). Also as x is changed from ξ_i to ξ_{i+1}, the number of panels changes from even to odd (or vice versa) in the intervals (0, x) and (x,1). Thus a composite Simpson-Trapezoidal rule would have to be adopted. However, the simple Trapezoidal rule suffers no such problems, giving a linear approximation to the integrand for all $x = \xi_i$. Furthermore, the two weight factors $\frac{1}{2}$ applied to the point x in the two intervals (0,x) and (x, 1) combine to give the standard Trapezoidal rule applied to the entire interval (0, 1). Thus the method described earlier for forming matrix D of Trapezoidal weights $\{w_i\}$ applies directly, for this kernel where the cusp lies at y = x. A similar conclusion would hold for a discontinuous kernel along the line y = x, provided the average value of the kernel were used in matrix B for x = y.

It is possible to apply Simpson's rule or Gauss quadrature to this particular kernel, and ignore the cusp. However, since the integrand of (2) is not being well approximated, poorer results can be anticipated. The results of Table 14 corroborate this. Slightly poorer characteristic values were obtained via Simpson's rule than Gauss integration; however, the reverse was true for the characteristic function approximations. The Trapezoidal rule outperformed both in all aspects, however.

Table 14

CHARACTERISTIC VALUES FOR N = 9

Simpson	Gauss	Trapezoidal	Exact
0.742329	0.742030	0.738619	0.738811
0.141696	0.140302	0.139433	0.138004
0.049161	0.048005	0.047308	0.045088
0.026400	0.024658	0.023878	0.021329
0.010096	0.016482	0.015078	0.012279

SUMMARY

The method of collocation for numerical solution of Fredholm integral equations has been investigated extensively via examples, and found to be workable and accurate when appropriate care is taken. For example, the selection of the integration rule is important, and the way non-existent solutions or discontinuous solutions manifest themselves in the numerical approaches considered must be known and anticipated.

The method of solution of integral equations of the first kind, as limits of integral equations of the second kind, is found to be a very worthwhile method in the case where continuous solutions exist. When a discontinuous solution exists, and the discontinuous portion $g_d(x)$ can be estimated, the integral equation (11) can be put in the form (by letting $g(x) = g_c(x) + g_d(x)$)

$$\int_a^b dy\ K(x,y)\, g_c(y) = f(x) - \int_a^b dy\ K(x,y)\, g_d(y),\quad a < x < b. \tag{42}$$

The right-hand side of (42) is known (when $g_d(x)$ is known or estimated); thus (42) is an integral equation for $g_c(x)$, to which the numerical approach can be applied with more numerical accuracy. A recursive procedure, whereby $g_d(x)$ is estimated more and more accurately, is especially worthy of consideration.

The FORTRAN program used for the above computations is listed in the Appendix. The six possible input variables are read in via NAMELIST and are defined within the comment section of the main program. Three external

subroutines, D Q U A D, F, and T K, are used to give the quadrature formula weights and abscissas, to define the function $f(x)$, and to define the kernel $K(x,y)$, respectively. All computations are double-precision, except the calculation of characteristic values and vectors which is single-precision (because the only routine now available for non-symmetric matrices, EIGENP, is single precision).

REFERENCES

1. F. B. Hildebrand, Methods of Applied Mathematics, Prentice-Hall Inc., N.Y., Second Printing, 1954.

2. P. J. Davis and P. Rabinowitz, Numerical Integration, Blaisdell Publ. Co., Waltham, Mass., 1967.

3. C. W. Helstrom, Statistical Theory of Signal Detection, Pergamon Press, N. Y. 1960.

APPENDIX

```
PARAMETER NR=100,NC=NR+1
PARAMETER M=50
DOUBLE PRECISION A(NR,NC),V(2),Y(NR),W(NR),F,TK
DOUBLE PRECISION ALPHA,BETA,DALAMB
DIMENSION JC(NC),B(M,M),EVR(M),EVI(M),VECR(M,M),VECI(M,M),INDIC(M)
LOGICAL EIGEN
NAMELIST/DATA/EIGEN,DALAMB,ALPHA,BETA,LIMIT1,LIMIT2,LIMIT3
LIMIT1 = 3
LIMIT2 = 20
LIMIT3 = 1
EIGEN = .FALSE.
*****************************************************************

IF EIGEN = .TRUE. , THEN THE CHARACTERISTIC VALUE PROBLEM
IS TREATED.....
        DALAMB * G(X)  =  INTEGRAL ( TK(X,Y)*G(Y)*DY  )
BOTH CHARACTERISTIC VALUES AND CHARACTERIC VECTORS ARE FOUND.
PRINT STATEMENTS MUST BE INSERTED FOR THIS CASE.

*****************************************************************

IF EIGEN = .FALSE.  , THE THE PROGRAM COMPUTES THE SOLUTION OF
FREDHOLM INTEGRAL EQUATIONS OF EITHER THE FIRST OR SECOND KINDS.
ANY INTERVAL OF INTEREST IS ASSUMED TO BE FROM ALPHA TO BETA.
F(X) IS THE FUNCTION WHOSE INVERSE IS SOUGHT.
TK(X,Y) IS THE KERNAL. DALAMB IS THE SCALE FACTOR.
IF DALAMB = 0.D0, THE EQUATION IS OF THE FIRST KIND...
        F(X) = INTEGRAL (  TK(X,Y)*G(Y)*DY  )
IF DALAMB NE 0.D0, THE EQUATION IS OF THE SECOND KIND...
        F(X) + DALAMB * G(X)  =  INTEGRAL ( TK(X,Y)*G(Y)*DY  )

*****************************************************************

IN ALL  CASES, ANY SYMMETRY WHICH WOULD RESULT IN REDUNDANT
EQUATIONS IS ASSUMED TO BE REMOVED.IF THIS IS NOT DONE, AN
ERROR RETURN FROM DGJR SHOULD OCCUR.

*****************************************************************
```

```
C  THREE EXTERNAL FUNCTION  SUBROUTINES ARE NECESSARY...
C  DQUAD(Y,W,N,ALPHA,BETA,$)    ,   WHERE...
C     Y IS THE ABSCISSA ARRAY FOR SOME QUADRATURE FORMULA
C     W IS THE WEIGHT   ARRAY FOR SOME QUADRATURE FORMULA
C     N IS THE ORDER OF THE QUADATURE FORMULA
C  TK(X,Y)   ,   WHERE...
C     Y IS THE VARIABLE OF INTEGRATION
C     X IS THE REMAINING ARGUMENT
C  F(X)   ,   WHERE... X IS AS IN TK(X,Y)
C
C*****************************************************************
C
C  SEVEN VARIABLES MAY BE READ IN NAMELIST. AT THE END OF EACH
C  CASE, THE PROGRAM ATTEMPTS TO READ NEW DATA.
C
C*****************************************************************
C
    1 READ (3,DATA)
      WRITE(4,DATA)
C
      DO 10 N=LIMIT1,LIMIT2,LIMIT3
      CALL DQUAD(Y,W,N,ALPHA,BETA,$250)
      MC = N + 1
      DO 101 I=1,N
      DO 101 J=1,N
      A(I,J) =TK(Y(I),Y(J))*W(J)
  101 CONTINUE
      IF(EIGEN) GO TO 125
      DO 99 I=1,N
   99 A(I,I) = A(I,1) - DALAMB
      DO 100 I=1,N
  100 A(I,MC) = F(Y(I))
      V(1) = 4.
      CALL DGJR(A,NC,NR,N,MC,$250,JC,V)
      PRINT 43,N,DALAMB
      PRINT 41 , ( Y(I),A(I,MC),I=1,N)
      GO TO 10
  250 PRINT 42
```

```
      PRINT 41,(Y(I),W(I),I=1,N)
      PRINT 41, V(1),V(2)
      PRINT 47 , ((A(I,J),J=1,MC),I=1,N)
   10 CONTINUE
      PRINT 44
      IF(NR.NE.0) GO TO 1
  125 CONTINUE
      DO 30 I=1,M
      DO 30 J=1,M
   30 B(I,J) = A(I,J)
      CALL EIGENP(M,M,B,27.,EVR,EVI,VECR,VECI,INDIC)
      ************************************************************
      EIGENP FINDS CHAR. VALUES AND VECTORS FOR A REAL NON-SYMMETRIC M/
      EVR AND EVI CONTAIN THE REAL AND IMAGINARY PARTS OF CHAR. VALUES,
      VECR AND VECI CONTAIN THE REAL AND IMAG. PARTS OF THE CHAR. VECT(
      REFERENCE... USL TECH MEMO 2070-163 , 12 MAY 1969.

      AT THIS POINT, APPROPRIATE PRINT STATEMENTS SHOULD BE INSERTED.
      ************************************************************
      IF(NR.NE.0) GO TO 1
   41 FORMAT(/2D28.18)
   42 FORMAT(' ERROR RETURN ')
   43 FORMAT(1H1,//,13X,'ABSCISSA',20X,'ORDINATE',20X,I5,' POINTS',10X,
     1 'DALAMB = ',D18.12,//)
   44 FORMAT(1H1)
   47 FORMAT(/6D21.14)
  111 STOP
      END
```

Solution Of Large Hermitian Eigenproblems On Virtual And Cache Memory Computers

R. L. Streit

Solution of Large Hermitian Eigenproblems on Virtual and Cache Memory Computers

Roy L. Streit
Naval Underwater Systems Center
New London Laboratory
New London, CT 06320

The impact of cache and virtual memories on the performance of state-of-the-art software has not yet been fully explored. This letter documents the truth of this statement by presenting the results of a computational experiment performed on the VAX 11/780 (a computer possessing both these features) with well established and documented FORTRAN subroutines designed for high accuracy numerical solution of the ordinary Hermitian eigenproblem $Az = \lambda z$. Easily implemented changes in the storage allocation of the complex matrix A resulted in roughly half an order of magnitude improvement in both elapsed CPU and wall clock time, as well as in significantly improved overall system throughput. These changes do not affect the numerical algorithm in any way.

The particular programs discussed here were written by International Mathematical and Statistical Libraries, Inc., [1] hereinafter referred to as IMSL. The storage allocation scheme for the complex matrix A used in the IMSL routines is similar to that used earlier in the EISPACK routines developed by Argonne National Laboratories [2]. The EISPACK routines were not resident on the available computer, so only the IMSL routines were used here. However, it is highly probable that the observations recorded below would be substantially unchanged had the EISPACK routines been used instead of the IMSL routines.

The most natural way to store the complex Hermitian matrix A is simply to declare A to be COMPLEX. Thus, if a_{ij} is a particular entry in the matrix A, then the address of the real part of a_{ij} and the address of the imaginary part of a_{ij} differ numerically by precisely 1; that is, the real and imaginary parts of a_{ij} are adjacent to each other in computer memory.

The approach taken by both EISPACK and IMSL is to separate the matrix A into two real matrices, AR and AI, which contain all the real and imaginary parts, respectively, of A. Thus, we have the matrix equation $A = AR + i\,AI$ with AR and AI typed REAL. The primary drawback to this method is that the address of the real part of a_{ij} and the address of the imaginary part of a_{ij} differ numerically by at least n^2, where n is the order of the eigenproblem; that is the real and imaginary parts of a_{ij} are widely separated from each other in the computer memory for large order eigenproblems.

For large order eigenproblems, this storage method can cause one cache miss for each arithmetic operation performed, as well as a very significant number of page faults to disc, thus considerably increasing both CPU and wall clock time on a computer such as the VAX. These concerns were discussed with Thomas Aird of IMSL, who responded by suggesting [3] several easily effected modifications to the appropriate IMSL routines (namely, EIGCH, EHOUSH, and EHBCKH), which force the real and imaginary parts of every entry a_{ij} of the matrix A to be stored in adjacent memory locations throughout the computations. These were implemented and found to work as anticipated.*

The solution of an order n = 210 eigenproblem was computed in an applications program [4] using both the standard IMSL code and the modified IMSL code. Other than in the IMSL routines, the actual code executed was identical. Both executions were performed in a dedicated environment; i.e., no other users were permitted during program execution. Thus the program overhead (setting of the matrices, I/O operations, etc.) and the system overhead was identical in both cases. Table 1 shows that the modified routines significantly outperformed the standard routines.

	IMSL	MODIFIED IMSL
Elapsed CPU time	56m 34s	25m 39s
Elapsed wall clock time	61m 57s	27m 17s
Page fault count	15,274	11,260
Direct I/O count	1,112	1,111
Buffered I/O count	157	157
Peak virtual size (pages)	2,465	2,465
Working set size (pages)	1,024	1,024
Mounted volumes	0	0

Table 1: Eigenproblem Order n=210

It is also clear from Table 1 that the page fault count is not significant in this particular problem. Improved performance is due primarily to improved use of cache memory. The VAX transfers two contiguous words (64 bits) on a cache miss. Thus, if a complex number, say z, is needed for an arithmetic operation and if z is not in cache, then z will be found with only one cache miss if

*Ed. Note: I have been informed by IMSL that the next version of their library in which the arrays are stored in COMPLEX mode, will eliminate this problem.

the real and imaginary parts are stored in adjacent memory locations. In any other storage mode, moving z into cache memory would cost two cache misses. Thus, it is fair to say that every arithmetic operation resulted in a cache miss. If there were 50 n^3 arithmetic operations and a cache miss costs 1.8 μs, then this assumption accounts for 17 minutes of CPU time savings. The remainder of the observed improvement has not been accounted for.

The applications program was executed again, but this time the solution of an eigenproblem of order n = 420 was required. This eigenproblem was solved using both the standard and modified IMSL routines as before. A significant difference is that other users were freely permitted on the system throughout both of the program executions whose results are presented in Table 2. In this case, page faults may well have had the more significant impact on program performance.

	IMSL	MODIFIED IMSL
Elapsed CPU time	12h 15m 48s	3h 51m 52s
Elapsed wall clock time	20h 53m 09s	13h 14m 55s
Page fault count	9,453,587	2,997,395
Direct I/O count	3,760	3,860
Buffered I/O count	264	264
Peak virtual size (pages)	8,707	8,707
Working set size (pages)	1,024	1,024
Mounted volumes	0	0

Table 2: Eigenproblem Order n=420

In a multiuser environment, page faults to disk tie up a disk controller for several tens of milliseconds and the large eigenproblems described in this letter significantly reduced overall system throughput. In the broad view this is probably a more serious problem than individual program CPU times.

The contents of this letter were first documented in [5].

References

1. IMSL Library, International Mathematical and Statistical Libraries, Inc., 7500 Bellaire Boulevard, Houston, Texas.

2. B. T. Smith, et al, Matrix Eigensystem Routines – EISPACK Guide, Lecture notes in Computer Science, Vol. 6, Second Edition, Springer-Verlag, 1976.

3. Thomas Aird, private communication, 14 November 1979.

4. R. L. Streit, "Array Optimization Using Subarrays," NUSC Technical Report 5889, Naval Underwater Systems Center, New London Laboratory, New London, Connecticut, 23 March 1979.

5. R. L. Streit and B. G. Buehler, "The Efficient Solution of Large Hermitian Eigenproblems on Virtual and Cache Memory Computers," NUSC Technical Memorandum No. 801019, Naval Underwater Systems Center, New London, Connecticut, 29 January 1980.