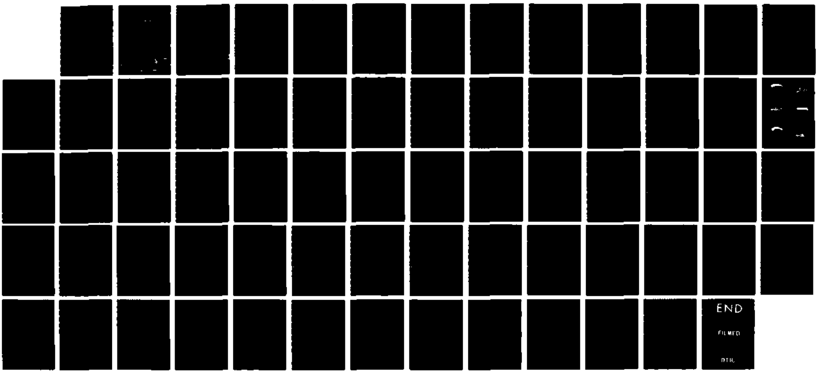
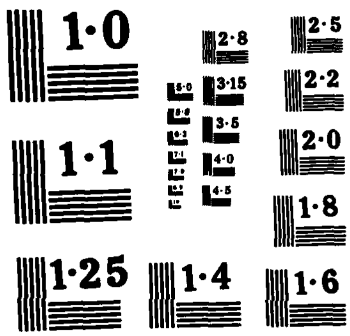
AD-A158 087 A TARGET TRACKER USING A DOPPLER COMPENSATED CORRELATION TECHNIQUE(U) TETRA TECH INC ARLINGTON VA B R ELDRIDGE 29 MAY 85 N00014-81-C-0535 1/1

UNCLASSIFIED F/G 12/1 NL




1.0
2.8
2.5
3.15
2.2
3.5
2.0
1.1
4.0
4.5
1.8
1.25
1.4
1.6

RESEARCH REPORT ON A TARGET TRACKER USING A DOPPLER COMPENSATED CORRELATION TECHNIQUE

by

B. R. Eldridge

Prepared in Response to: Contract N00014-81-C-0535
with
The Office of Naval Research
May 29, 1985

Tetra Tech/Honeywell, Inc
1911 N. Fort Myer Dr.
Arlington,VA 22209

RESEARCH REPORT ON A TARGET TRACKER USING A DOPPLER COMPENSATED CORRELATION TECHNIQUE

by

B. R. Eldridge

Prepared in Response to: Contract N00014-81-C-0535
with
The Office of Naval Research
May 29, 1985

Tetra Tech/Honeywell, Inc
1911 N. Fort Myer Dr.
Arlington,VA 22209

PREFACE

This document is the final research report on the investigation of a mathematical algorithm to do target tracking, using doppler compensated correlation techniques on input time series streams from several passive acoustic sensors. The algorithm was developed and programmed into a testbed on a VAX-750 computer and was tested using simulated time series data generated by the Tetra Tech Broadband Signal Simulator. Algorithm performance proved dissapointing due to: (1) Numerical instabilities induced by structural anomolies in the sample signal autocorrelation function; (2) The extreme sensitivity of objective function to choice of signal characteristics and processing parameters; (3) Computational intensity of the algorithm.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

For the past year, Tetra Tech has been involved in the development and analysis of an algorithm for tracking maneuvering submarines using Doppler compensated correlation techniques. The intended goal was the identification of an algorithm which would be more responsive to target kinematic changes than the usual Kalman filter, thereby providing timely and accurate estimates of target position course and speed at various times along the track. The work was carried out under contract N00014-81-C-0535 with the Office of Naval Research.

The algorithm was developed and a testbed computer code was generated for implementing and testing the algorithm. This program was written in FORTRAN-77 on a VAX-750 computer and is set up to use the Tetra Tech Broadband Signal Simulator (BSS) outputs as as its input time series. No attempt was made in this effort to use actual at sea data. The FORTRAN listings of the testbed program are included in Appendix A of this report.

The algorithm research carried out by Tetra Tech assumed the availability of base banded, bandlimited digitally sampled time series from several passive acoustic sensors. The availability of information such as bearings and time delays were not assumed in the testbed program, since it was felt that working directly with the time series data would provide a more convenient means of inplementing a maneuvering target tracker. Due to the general structure of the algorithm, additional measurement types such as those mentioned above can be easily added if desired.

As is well known, one of the characteristics of a sequential or Kalman type of estimating scheme is the tendency of the estimator to build up "inertia" and thereby make it unresponsive to changes in target course and speed after long periods of tracking time. This may be overcome by certain ad hoc schemes such as frequent reinitializations or possibly by tampering with the weights so as to cause the algorithm to have a shorter memory, etc.

In view of this it seemed reasonable to employ an estimation scheme which works directly with selected blocks of time series data in a batch mode, and which can be highly overlapped from estimation to estimation. Once the algorithm has been initialized and is operation, the previous estimate of target state can be used to initialize the trial solution for the current estimation. The covariance matrix of the initialization state is not carried over and therfore the process is without a memory. However, if sufficient overlap in the input time series is used, the output states should show mimimal change from estimation to estimation while still reflecting the most current information available from the time series.

Time series generated by a single moving source and received at two or more spacially separated sensors will exhibit different Doppler and time delay characteristics at each of the receivers. These characteristics are, of course, dependent on sensor-target geometry and kinematics, as well as sound propagation physics. By applying the appropriate time and Doppler compensation to the received time series, pairwise time series correlations between sensors can be maximized. By linking the time and Doppler compensation to assumed target motion, one can adjust the target state parameters to maximize (or minimize) an appropriately chosen function of the corresponding pairwise cross correlation estimates.

The assumption of digitally sampled time series sampled at a uniform sample rate suggests that the time and Doppler compensation be done in the time domain using an interpolative resampling technique. This invloves estimating time series values whose sample times lie between the discreet sample times of the input time series. For band limited signals, the well known Sampling Theorem provides a rational means of performing the required interpolation using neighboring time series points and the sinc function as an interpolating function. This, in effect, generates a piecewise continuous representation of the original time series thereby allowing resampling at arbitrary times which are in concert with trial target state parameters. Also, such a scheme allows analytic evaluation of the gradient vector with respect to the state variables and weighting vectors.

As has been mentioned above, Tetra tech has implemented these ideas into a testbed algorithm on the VAX-750 digital computer. Inputs to the algorithm consist of up to 10 channels of time series data. For each channel, the algorithm requires an estimation of signal to noise ratio (SNR) along with the standard deviation of the SNR for that channel. The algorithm is also sensitive to such inputs as integration time, processing bandwidth and center frequency, station location, sound speed in water, time series overlap, and initialization parameters such as position course and speed. The target kinematic model consists of polynomials in water time of up to degree 5 for x,y and z. The order of the polynomials is user specified. Model output consists of estimated target parameters, their associated error variances, and the size and orientation of the 2-σ containment ellipsoid.

The algorithm has been tested using simulated time series generated by the BSS. The BSS can emulate the complex time series generated by a moving target having user specified kinematic and spectral output signal characteristics.

Section 2 of this report gives an overall description of the workings of the algorithm. Section 3 describes the Gauss-Newton estimation scheme as it applies to this effort, and Section 4 outlines and justifies the doppler compensation scheme that we have employed. Section 5 contains the objective function minimization process description and algorithm performance is reported in Section 6. Finally, Section 7 presents the summary and conclusions of this effort. Appendix A contains the FORTRAN listings of the testbed program developed as part of this effort.

2. ALGORITHM DESCRIPTION

The algorithm estimation scheme assumes the availability of several channels of bandlimited, basebanded, discreetly sampled digital data for which all of the pertinent parameters such as sample rate, bandwidth, and center frequency are known, and all of which contain signal from a common emitter. For the purposes of this analysis, we will assume that the noise on each channel is mutually uncorrelated. We will also assume that the target is moving along some 3-dimensional trajectory, which is given by the vector function $\mathbf{P}(\mathbf{s};t)$, where $\mathbf{s}$ is the state vector to be estimated and t is the time along the trajectory.

The testbed version of the algorithm assumes that each of the components of $\mathbf{P}(\mathbf{s};t)$ is an n'th order polynomial in t, and the state vector $\mathbf{s}$ consists of the coefficients of these polynomials. The testbed user may specify n to be any non negative integer up to and including 5. In practice, n is usually chosen to be 1, resulting in linear target motion at constant speed. In this case, $\mathbf{P}(\mathbf{s};t)$ is given by

$$\mathbf{P}(\mathbf{s};t)=\mathbf{P}_0+\mathbf{V}t \qquad (2.1)$$

and the state vector $\mathbf{s}$ may be represented in transposed form by

$$\mathbf{s}=[\mathbf{P}_0{}^T,\mathbf{V}^T]^T \qquad (2.2)$$

The superscript T denotes the usual matrix transpose operator.

In order to keep the algorithm tractible, we have assumed linear, constant speed sound propagation. More complicated propagation models could have been incorporated, but it was deemed an unnecessary complication at these early stages of algorithm development and feasibility analysis.

The central idea of the algorithm is to mimimize a quadratic form of "system functions". The system functions are dependent on the collection of pairwise normalized sample correlation envelope functions which have been adjusted to account for assumed target kinematics. Each sample correlation envelope function is obtained by correlating the samples from a selected reference channel with modified sets of samples that have been interpolated and resampled from each the other channels comprising the tracking system. The resampling times are calculated as a function of the current value of the state vector, sensor kinematics, the assumed propagation model, and channel signal processing parameters such as center frequency, sample rate, and bandwidth.

To be specific, let us consider a pair of channels, say channels X and Y. Pick a set of samples $\{X_1,X_2,\ldots,X_n\}$ from channel X. These samples correspond to arrival times $\{u_1,u_2,\ldots,u_n\}$ on channel X. In order to do the motion compensation, we need to calculate the corresponding arrival times $\{v_1,v_2,\ldots,v_n\}$ on channel Y. This is done by using the candidate source trajectory **P(s;t)** to calculate emitter times $\{t_1,t_2,\ldots,t_n\}$ corresponding to the X channel arrival times $\{u_1,u_2,\ldots,u_n\}$, and using these emitter times to project the corresponding arrival times $\{v_1,v_2,\ldots,v_n\}$ on channel Y. We then interpolate and resample the Y channel at the $\{v_1,v_2,\ldots,v_n\}$ thereby obtaining a new set of samples $\{Y_1,Y_2,\ldots,Y_n\}$. The interpolation is accomplished using a truncated sinc function as an interpolating function. Details of the interpolation scheme are provided in Section 4 of this report.

The magnitude squared cross correlation estimate γxy is then calculated by

$$\gamma xy=|\sum X_i Y_i^*|/\{\sum|X_i|^2\sum|Y_i|^2\} \qquad (2.3)$$

where the three sums in the above expression are taken over $i=1,2,\ldots,n$ and the superscript (*) denotes complex conjugation. The values of the γxy so obtained are used to form the aformentioned system functions Fxy which are used in the minimization process. The system functions are given by

$$Fxy=\ln(Gxy/\gamma xy) \qquad (2.4)$$

where Gxy is the a priori expected value of γxy. Gxy is related to the input SNR estimates on each channel by

$$Gxy=[(1+SNRx^{-1})(1+SNRy^{-1})]^{-1} \tag{2.6}$$

Note that the Fxy are chosen such that they have value 0 when given error free information.

Finally, Q(**s**), a positive definite quadratic form of the Fxy, is formed over all channel pairs, and by using gradient methods, the state vector **s** is adjusted so as to mimimize Q(**s**). The vector $\mathbf{s}_0$ which mimimizes Q(**s**) is taken as the state estimate.

A fallout of the minimization process is an estimate of the state covariance matrix. This matrix is used to calculate the ellipsoidal containment region which provides the user with a geometric indication of algorithm performance.

3. THE ESTIMATION SCHEME

3.1 Preliminary Details and Notation

this section details the estimation scheme in rather general mathematical terms. Preliminary to the discussion we establish the following notation which will be used throughout the remainder of this report.

If **X** and **Y** are n-dimensional complex valued vectors, the complex inner product of **X** and **Y**, denoted by $\langle \mathbf{X},\mathbf{Y} \rangle$, is defined by

$$\langle \mathbf{X},\mathbf{Y} \rangle = \sum X_i Y_i^* \tag{3.1}$$

where the sum is taken over $i=1,2,\ldots,n$, and the X_i and Y_i are the complex valued components of **X** and **Y**, respectively. The (*) notation denotes complex conjugation. Note that the complex inner product is conjugate symmetric in that the following relationship holds:

$$\langle \mathbf{Y},\mathbf{X} \rangle = \langle \mathbf{X},\mathbf{Y} \rangle^* \tag{3.2}$$

We define the norm of **X**, denoted by $\|\mathbf{X}\|$, by

$$\|\mathbf{X}\| = \sqrt{\langle \mathbf{X},\mathbf{X} \rangle} \tag{3.3}$$

In this notation, Equation 2.3 of Section 2 becomes

$$\gamma_{xy} = |\langle \mathbf{X},\mathbf{Y} \rangle|^2 / (\|\mathbf{X}\|^2 \|\mathbf{Y}\|^2) \tag{3.4}$$

If each of the components of the complex valued vector is a differentiable function of some real parameter θ, then denote the vector consisting of the corresponding derivatives (partial derivatives) by $d\mathbf{X}/d\theta$ ($\partial\mathbf{X}/\partial\theta$). It is easy to verify that the following useful relationship is true

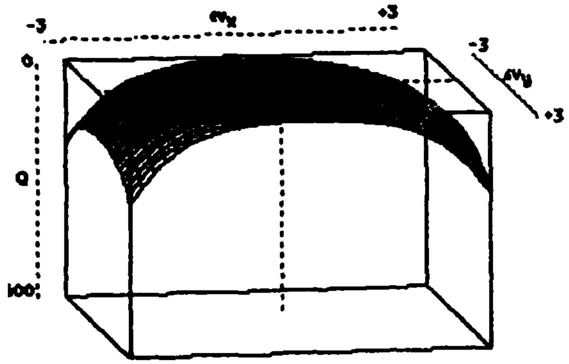


Figure 6.2
Objective Function vs errors in Velocity Probes
fc=20 Hz
Tint=20 sec
BW= 6 Hz
Position Offset =(0,0)

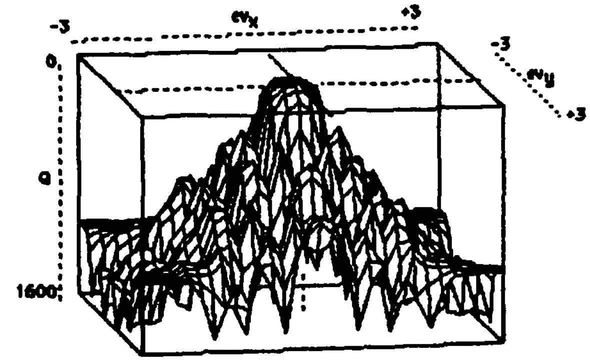


Figure 6.3
Objective Function vs errors in Velocity Probes
fc=20 Hz
Tint=100 sec
BW= 6 Hz
Position Offset =(0,0)

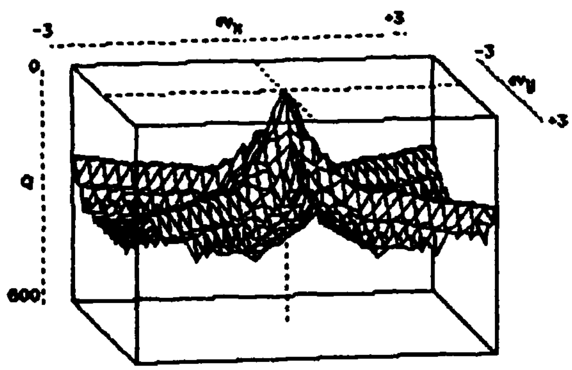


Figure 6.4
Objective Function vs errors in Velocity Probes
fc=250 Hz
Tint=20 sec
BW= 6 Hz
Position Offset =(0,0)

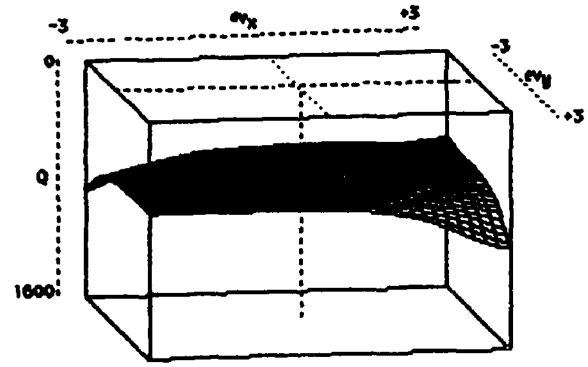


Figure 6.5
Objective Function vs errors in Velocity Probes
fc=20 Hz
Tint=20 sec
BW= 6 Hz
Position Offset =(300,300)

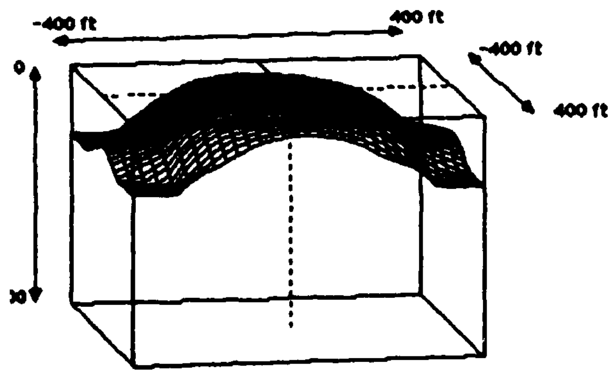


Figure 6.6
Objective Function vs Position Error
fc=20 Hz
Tint=20 sec
BW=6 Hz
Velocity Errors = (0,0)

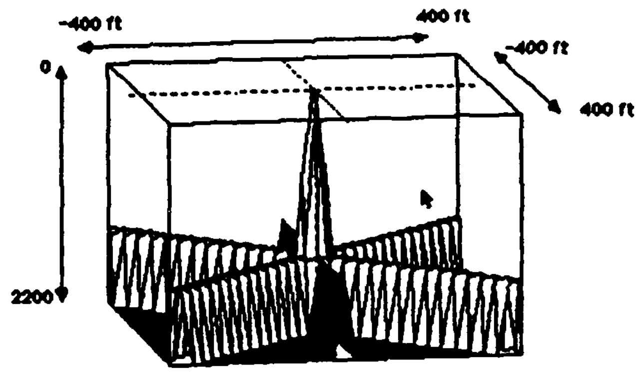


Figure 6.7
Objective Function vs Position Error
fc=80 Hz
Tint=20 sec
BW=100 Hz
Velocity Errors = (0,0)

The following figures show the kinds of difficulties one faces even under the best of circumstances. Figures 6.2-6.5 show plots of the objective function versus errors in the probe or trial solution velocities(ft/sec) in both the x and y direction for several sets of processing parameters. The assumed position error is assumed to be fixed at 0 ft. The velocity errors range from ±3 ft/sec in both the x and y direction. The center of the portion of the x-y plane shown represents 0 error. The z axis represents the values of the objective function and has been inverted for the sake of this presentation. The z axis ranges from 0 (top) to 1600 (bottom). The signal bandwidth is assumed to be 8 Hz.

Figure 6.2 was generated assuming a center frequency of 20 Hz with an integration time of 20 seconds and presents a very clean objective function over the plot range. In Figure 6.3, the integration time has been increased 100 seconds. The resulting plot exhibits a much more spiked peak around (0,0) and the outskirts show a good deal of ripple. Figure 6.4 is similar to Figure 6.2 except that the center frequency of the signal has been shifted to 250 Hz. This band shift has caused the peak to become very sharp with even more ripple evidenced in the outskirts. The processing situations depicted in Figures 6.3 and 6.4 would present problems to the tracker. Figure 6.5 has identical parameters as Figure 6.2 with the exception that the assumed position error has been offset to 300 ft. in both the x and y directions resulting in a much more planar shape and having increased the overall magnitude of the objective function considerably.

Figures 6.6 and 6.7 show plots of the objective function versus errors in the trial position estimates with the geometry in Figure 6.1 applying. The position errors range from ± 400 ft in both the the x and y directions. Figure 6.6 corresponds to a signal bandwidth of 8 Hz and presents a rather smooth function with essentially no unusual structure. In Figure 6.7 we have increased the signal bandwidth to 100 Hz and moved the center frequency to 80 Hz. The resulting plot contains considerable structure with the parameters, we developed a computer program to generate the expected value of Q(s) for a particular signal autocorrelation function. The outputs are sensitive to the aformentioned signal and processing parameters as well as target/sensor geometry and kinematics. The autocorrelation function we have chosen is triangular on the interval [-T,T] and has a spectral density function of the form $\mathrm{sinc}^2(\omega T/2)$. Its bandwidth is approximately 1/T. This particular autocorrelation function has the advantage of being integrable in closed form. The program is setup to generate surfaces of expected values of the tracker objective function, holding the position probes fixed and letting the velocity probes vary, or holding the velocity probes fixed and letting the position probes vary.

Some results are presented for the geometry shown in Figure 6.1. Here the target is assumed to be in the center of a square box with the sensors located at the vertices. The distance from the target to each of the sensors is assumed to be 2 nmi. A SNR of 6 dB with a σ_{SNR}=3 dB is assumed.

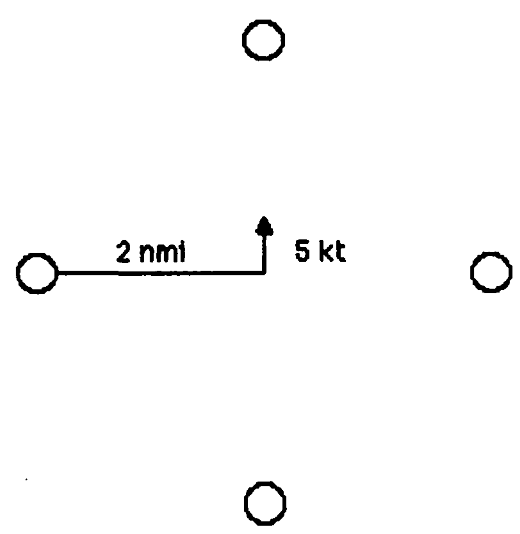


Figure 6.1
Test Geometry

6. ALGORITHM PERFORMANCE

6.1 Parameter Selection

Since the objective function Q(s) is a very complicated function (most of which is carried out in the complex domain) of a number of processing parameters, some thought had to be given to their effects on algorithm behavior. The user has control over such things as processing bandwidth, center frequency, and integration time.

For example, if one chose to process a signal having a sufficiently high center frequency for a sufficiently long period of time, Q(s) could become quite sensitive to trial solution errors in velocity. In fact one would expect to see spike like behaviour in the objective function in the neighborhood of the true velocity components, which could conceivably cause convergence difficulties.

Similarly, a wide bandwidth signal could cause similar spike like behavior in trial solution position estimates. Small positional errors wind up in the ripple of the objective function which wreaks havoc on the convergence process we have chosen.

On the other hand, if the processing band is too narrow, the lack of time delay resolution may not provide any meaningful information to the algorithm, again resulting in poor behavior.

These were in fact some of the problems that were encountered during the early stages of algorithm testing. Since the testbed computer code was newly developed, it was not known whether poor early algorithm performance was due to bugs in the program, or whether the algorithm just did not work, or whether we had just chosen bogus processing parameters. After considerable reexamination ,rechecking, and rederiving the mathematics, we decided that they were correct. We could find no bugs in the program and so our only recourse was to give careful scrutiny to our choice of test parameters.

In order to gain insight to the effects of processing

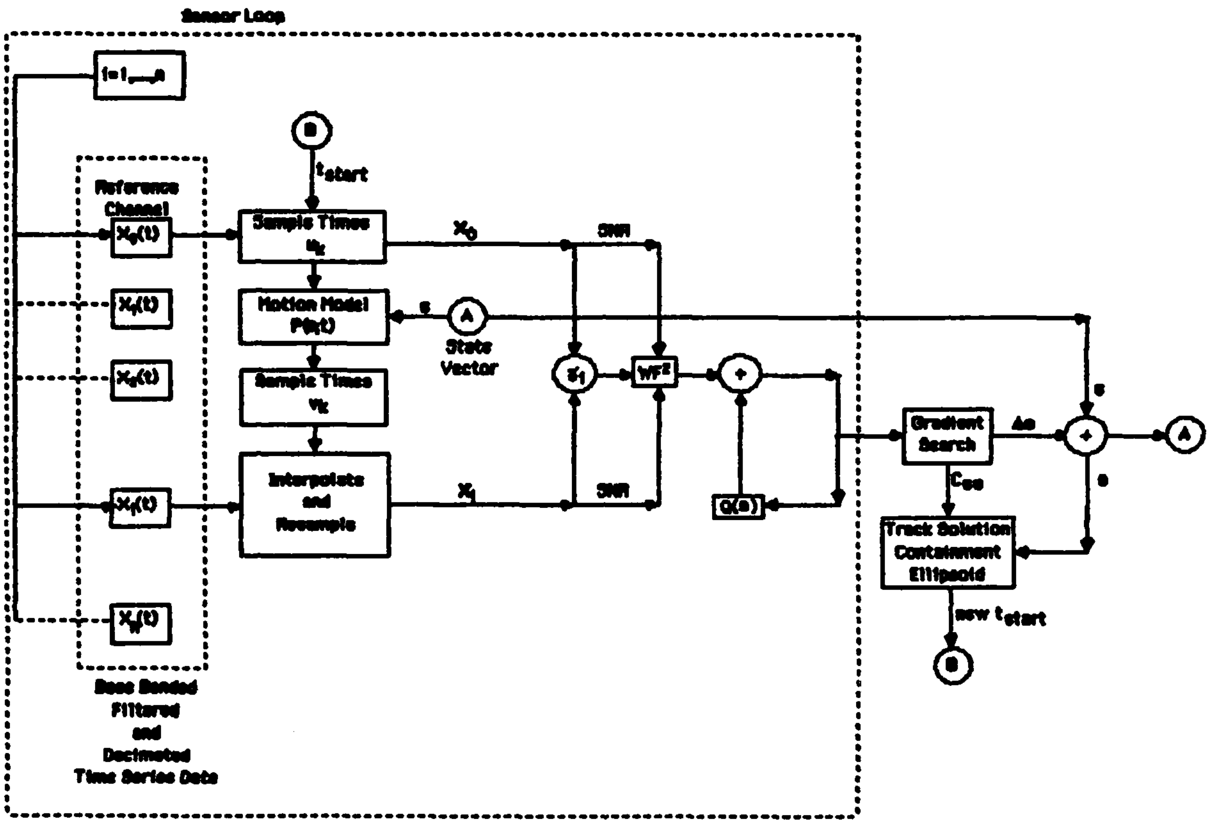


Figure 5.1
Tracker Flow Logic

which yields

$$\partial s=-(F_s^T W F_s)^{-1}(F_s^T W F)=C_{ss}\,(F_s^T W F) \qquad (5.5)$$

The advantage of this approach is that it does not require the computation of second derivatives, and that an estimate of the state covariance matrix falls out of the process.

In summary, if **s** is the current trial solution for the process, we form the next iterate **s'** by calculating ∂**s** as in the preceeding equation and forming **s'** by

$$s' = s + \partial s \qquad (5.6)$$

To stop the process we check to see if the magnitude of ∂**s** is within some predescribed tolerance. If so the process is stopped and the current value of the state vector is returned as the solution. If not the process continues until a solution is returned or the maximum iteration count is exceeded.

The entire process is described in Figure 5.1.

5. THE MIMIMIZATION PROCESS

5.1 The Iteration Scheme

The function Q(s) is a complicated function of the state vector and measurement vector, the mimimization of which does not seem amenable to closed form solutions. We therefore must rely on iterative techniques to solve the problem. The following paragraphs describe the technique we have chosen to accomplish the minimization.

Recall that Q(s) is a quadratic form in the system functions and may be written

$$Q(s)=F^TWF \tag{5.1}$$

where the weighting matrix **W** is chosen to be the inverse of the covariance matrix of the system residual vector. Under the assumption of slowly varying weights, the gradient vector of Q(s) may be written

$$\nabla Q(s)=2F_s{}^TWF. \tag{5.2}$$

At a local mimimun we have the necessary condition that

$$\nabla Q(s)=2F_s{}^TWF=0 \tag{5.3}$$

An approach which has been used sucessfully is demonstrated in the following discussion. Suppose that the algorithm has reached a stage such that **s** is the current value of the trial state vector. We would like to find the perturbation ∂**s** to add to **s** which will improve the estimate. A reasonable approach is to solve the the following perturbed gradient equation for ∂s

$$F_s{}^TW(F + F_s\partial s)=0 \tag{5.4}$$

receiver times using $Q_h(t)$ as the interpolating function, thereby generating a set of Y channel samples which reflect the Doppler corrections implied by the current value of the state vector.

In order to obtain the receiver times on the Y channel, we solve the following pair of equations for v_k given u_k:

$$u_k = T_k + |\mathbf{P}(\mathbf{s},T_k) - \mathbf{P}_x|/c \tag{4.6}$$
$$v_k = T_k + |\mathbf{P}(\mathbf{s},T_k) - \mathbf{P}_y|/c$$

where $\mathbf{P}_x$ and $\mathbf{P}_y$ are the respective position vectors of the X and Y channel receivers, c is the speed of sound in water, and T_k is the emitter time. This is done by solving the first equation for T_k using a Newton- Raphson technique, and then using the second equation with the value of T_k so obtained to obtain v_k.

Recall that we are assuming that all of the input time series data has been basebanded from some center frequency f_c. Because of this, a phase correction prior to correlation is required on both channels. This merely amounts to heterodyning the samples back up to their original center frequency f_c. Form the complex vectors $\mathbf{X}$ and $\mathbf{Y}$ whose k'th components are given by $\exp\{2\pi j f_c(u_k-u_0)\}X_{n+k-1}$ and $\exp\{2\pi j f_c(v_k-v_0)\}Y_{n+k-1}$, respectively. Then γxy is given by

$$\gamma xy = |\langle \mathbf{X},\mathbf{Y}\rangle|^2/\{\|\mathbf{X}\|^2\|\mathbf{Y}\|^2\} \tag{4.7}$$

4.3 System Function Derivatives

The algorithm uses gradient methods to minimize $Q(\mathbf{s})$, which is the quadratic form of the system functions Fxy as discussed above. This necessitates the calculation of the derivatives of $Q(\mathbf{s})$ with respect to each of the state variables. Most of the work is done in computing the derivatives of the γxy with respect to the measurement vector and each of the state variables. The mathematical development of these derivatives is straightforward but extremely tedious and will not be included here.

reconstructed from its samples, provided that the digital sample rate is at least as great as the bandwidth of the signal. The reconstruction uses the "sine x over x" or sinc as an interpolating function. Specifically, if $Z(t)$ is a time series having non zero frequency content only in the interval $[-b/2,b/2]$, and if $Z(t)$ is uniformly sampled over all time at a sample rate $f_d \geq b$, producing samples Z_n, $-\infty \leq n \leq \infty$, and such that Z_0 corresponds to a sample time of 0, then $Z(t)$ is reconstructed exactly from its samples by

$$Z(t)=\sum \text{sinc}\{\pi(f_d t-n)\}Z_n \qquad (4.3)$$

The above sum is taken over all n. This, however, involves summing over an infinite number of elements. It therefore seems reassonable to approximate the reconstruction of $Z(t)$ from its samples by using a time limited version of the sinc function. If we define the interpolating function $Q_h(t)$ by

$$Q_h(t)=\begin{cases} \text{sinc}(t), & |t| \leq h \\ 0 & |t| > h \end{cases} \qquad (4.4)$$

, then $Z(t)$ may be approximated by

$$Z(t) \approx \sum Q_h\{\pi(f_d t-n)\}Z_n \qquad (4.5)$$

which, for any given value of n involves only finite sums.

Let us assume we are working with channels X and Y and we wish to calculate γxy for the current value of the state vector. Pick a set of reference samples $\{X_n, X_{n+1}, \ldots, X_{n+M-1}\}$ from channel X. These samples correspond to the set of channel X receiver times $\{u_n, u_{n+1}, \ldots, u_{n+M-1}\}$. We the use **P(s;t)** and the assumed propagation model to determine the corresponding set of channel Y receiver times $\{v_n, v_{n+1}, \ldots, v_{n+M-1}\}$. Channel Y is then interpolated at these

4. DOPPLER COMPENSATION SCHEME

4.1 Time Series Assumptions

The algorithm assumes the availability of M channels of data, and that the time series for each data channel contains basebanded, bandlimited data that has been sampled at at least the Nyquist sampling rate. For the sake of notational simplicity, we will assume that all of the channels have the same center frequency, f_c, and the same digital sampling rate f_d. We also assume that the n'th sample on each channel occurs at the same time. The time between samples, Δt, is given by

$$\Delta t=1/f_d \tag{4.1}$$

Therefore if the 0'th sample corresponds to t_0, then the n'th sample corresponds to t_n, where

$$t_n=t_0 + n\Delta t \tag{4.2}$$

The algorithm requires an estimate of the signal to noise ratio and its corresponding error variance on each channel of input data. It is assumed that these are independently specified and will be denoted SNR_m, where the subscript m refers to the particular channel designator.

4.2 Resampling and Phase Compensation

In order to effect the proper Doppler compensation, it is necessary to interpolate between samples in the time domain, with the interpolation times reflecting the current value of the state vector.

The well known sampling theorem from signal processing states that a complex valued bandlimited signal can be completely

$$B=(F_s{}^T WF) \tag{3.18}$$

Suppose we want to add a new measurement set to the algorithm that is independent of those already incorporated and can be described with a single system equation. This merely requires the specification of the corresponding system function and its associated partial derivatives. In keeping with our earlier discussion, let us further assume that the partial of the new system equation with respect to its measument vector are independent of the state vector. In this case we first form the new scalar weight w by

$$w=1/\sum[(\partial F/\partial m_i)\sigma_{mi}]^2 \tag{3.19}$$

where F is the new system function, the m_i are its associated measurements and

$$\partial F/\partial m_F=[\partial F/\partial m_i]. \tag{3.20}$$

The updated **B** vector and Z matrix are given by

$$B=B_{old} + wF(\partial F/\partial S) \tag{3.21}$$

$$Z=Z_{old} + w\,(\partial F/\partial S)\,(\partial F/\partial S)^T \tag{3.22}$$

where $\partial F/\partial S$ is the vector of partials of F with respect to the elements of the state vector.

$$F_s W F = 0 \tag{3.13}$$

where F_s denotes the matrix whose mn'th element is $\partial F_m/\partial s_n$. If the above equation holds, a perturbation in the measurement vector $\partial \mathbf{m}$ induces a perturbation in the state vector $\partial \mathbf{s}$, which to within first order terms, obeys the relationship

$$F_s^T W(F + F_s \partial s + F_m \partial m) = 0. \tag{3.14}$$

This implies

$$\partial s \approx -(F_s^T W F_s)^{-1} F_s^T W F_m \partial m, \tag{3.15}$$

which yields, after substituting C_{mm}^{-1} for $\mathbf{W}$

$$C_{ss} \approx E(\partial s \partial s^T) = (F_s^T W F_s)^{-1} \tag{3.16}$$

The expression for C_{ss} given above is extremely convenient and can be used to provide geometrical insight to algorithm behaviour. In particular, it is used to derive the ellipsiodal containment region of the current estimate of the state vector.

3.3 Incorporation of Additional Measurements

The form of the estimator used in this algorithm has the advantage that it is easy to incorporate new types of measurements should the need arise. If, for example, the system can provide independent estimates of position and velocity or if another independent sensor comes on line, the new measurements so provided may be entered as an additive partitions to the system weighting matrix and gradient vector. Let us first establish the following notation. Let

$$Z = (F_s^T W F_s) \tag{3.17}$$

We now provide our rational for our choice of **W**. Let **m** denote the vector of measurements associated with F. In our case, each component of **m** is an SNR estimate from one of the output channels. For any fixed value of **s**, the perturbation in the vector F induced by a perturbation in the vector **m** is approximated by

$$\partial F \approx [F_{\mathbf{m}}] \partial \mathbf{m} \tag{3.9}$$

Here $[F_{\mathbf{m}}]$ is the matrix whose ij'th element is $\partial F_i / \partial m_j$. Assuming the error distribution is zero mean with covariance matrix $C_{\mathbf{mm}}$, and that the approximation to ∂F holds over the probable values of **m**, we may approximate the covariance matrix of F by

$$C_{\mathbf{FF}} \approx F_{\mathbf{m}} C_{\mathbf{mm}} F_{\mathbf{m}}^T \tag{3.10}$$

The weighting matrix **W** referred to in in Equation (3.8) is chosen to be $(C_{\mathbf{FF}})^{-1}$. Therefore the scalar function Q(**s**) is given by

$$Q(\mathbf{s}) = F^T (C_{\mathbf{FF}})^{-1} F \tag{3.11}$$

This choice of **W** has intuitive appeal in that measurements with higher variance get less weight and, therefore, have less of an effect on the final outcome.

We now turn our attention to the approximation of the output state covariance matrix, which will be denoted by $C_{\mathbf{ss}}$. The components of the state vector $\mathbf{s}_0$ that minimizes Q(**s**) satisfy the equation

$$\partial Q / \partial s_i = 2(\partial F / \partial s_i) \mathbf{W} F + F^T (\partial \mathbf{W} / \partial s_i) F = 0 \ , i = 1,2,\ldots,k \tag{3.12}$$

If we assume slowly varying weights which enables us to ignore the second term in the above equation, then $\mathbf{s}_0$ satisfies the system of equations

$$\partial\langle X,Y\rangle/\partial\theta=\langle\partial X/\partial\theta,Y\rangle + \langle X,\partial Y/\partial\theta\rangle. \qquad (3.5)$$

The above relationship is indespensible in computing the gradient derivatives during the minimization process.

3.2 Theoretical Development

The general theoretical basis for the estimation scheme is an extension of the techniques of linear optimal estimation theory to the nonlinear case. We evaluate a set of system equations Fxy as given by Equation ( ) and then minimize a positive definite quadratic form of the Fxy. Suppose we have k such system functions for a particular application. Changing notation for convenience, let the system functions be denoted by $F_1,F_2,\ldots,F_k$ and let

$$\mathbf{F(s)}=[F_1,F_2,\ldots, F_k]^T \qquad (3.6)$$

denote the k-dimensional vector of system equations. Note that the solvability of the above equations implies that $n_s \le k$, where n_s is the dimentionality of the state vector.

If the geometric and signal assumptions are perfectly compatible with the measurement data, there exists a value of the state vector $\mathbf{s}_0$ for which

$$\mathbf{F(s)=0.} \qquad (3.7)$$

In general, however the data does not support perfect compatibility, and for each value of the state vector **s**, **F(s)** may be considered a vector of F-residuals. An optimal estimate of **s** is a vector which minimizes a particular quadratic form in the F-residuals. If **W** is an appropriately chosen positive definite symmetric matrix, the optimal estimate is the value which minimizes the scalar function

$$Q(\mathbf{s})=\mathbf{F}^T\mathbf{WF}. \qquad (3.8)$$

crisscrosses being corresponding to time delays among the various sensors. Again, finding the minimum of such a function if the solution starts off of the main peak presents quite a formadible task to the tracker.

We have presented these results to demonstrate difficulties in choosing a set of operating parameters for which we may have some hope of achieving positive results. To this end and after having a large number of cases as above, we chose to work with a time series centered at 20 Hz containing an 8 Hz signal in 10 Hz wide total processing bin. The signals were generated by the BBS for sensors having the geometry described in Figure 6.1 above, and had an overall SNR of +6dB in the processing band to each of the sensors. Using an integration time of 20 seconds and initialiazing the algorithm with "truth" at various points along the target track, the algorithm evidenced unstable convergence in nearly every case. Since we had gone over the computer program and the mathematics very carefully and could find no errors, we had to look elsewhere for an explanation of the poor algorithm performance. We believe the answer lies in the form of the system functions.

The **Z** matrix and **B** vector defined in Section 3 of this report are used extensively in the Gauss-Newton mimimization technique that we chose to implement for this algorithm. Note that both **Z** and **B** contain the partial derivatives of the system functions with respect to each of the elements of the state vector. When the system is near a solution, the partials of the system functions are near 0, since they are essentially the derivatives of the signal autocorrelation function at t=0. This causes the **Z** matrix to become numerically unstable as the process nears a solution, thereby causing unpredictable algorithm behaviour. Efforts to accomodate alternate forms of the system functions turned proved unsucessful because we could not find a tractible method of comparing time series that did not involve correlations. The only other alternative to sucessfully implement the algorithm would be to incorporate a more sophisticated minimization technique which would overcome the instability problems. This, too, could have drawbacks since most such algorithms involve several one dimensional searches which require several objective function evaluations which is quite computationally intensive.

We actually tried a method which involves halving the length of the search vector ∂s until $Q_{K+1}<Q_K$ at which time we would recalculate a new search vector and continue the process. This proved to be too computationally intensive even for TW products on the order of 200 and including 4 sensor geometry. We did, however obtain some success in achieving algorithm convergence. However, each evaluation of Q(s) took about 2 minutes CPU time on the VAX, resulting in enormous total algorithm processing times. This was deemed unsatisfactory from the point of view of any practical application.

7. SUMMARY AND CONCLUSIONS

An algorithm to use doppler compensated resampling and correlation techniques on digitally sampled time series was developed and tested using synthetic time series data generated by the Tetra Tech Broadband Signal Simulator (BBS). The algorithm was designed to provide timely track estimates by using highly overlapped time series segments from the receiving sensors in a batch mode, thereby giving the process a short memory and increased responsiveness to target maneuvers. The target motion model consisted of polynomial functions of time of arbitrary order up to 5.

Algorithm performance proved dissapointing for several reasons:

(1) Computational intensity was more than was originally envisioned.

(2) Structure of the correlation functions induced numerical instabilities in the convergence process which could not be easily overcome.

(3) The structure of the objective function is, in general, quite irregular, thereby requiring careful, and perhaps limited, choice of processing parameters in order to have any hope of successful performance.

Some improvement in running time could be achieved by simplifying the resampling technique to use first and perhaps second order time series stretches based on the current value of the position and velocity estimates.

Curing the numerical instabilities seems to be the most difficult hurdle to overcome due to the often unusual structures evidenced in correlation functions. For this reason, we feel that any further endeavers to improve upon such an algorithm would prove to be dissapointing and recommend that further research be discontinued.

APPENDIX A

TESTBED COMPUTER PROGRAM LISTINGS

```
              PROGRAM TRDRIV
              INCLUDE 'TRCNTRL.CMN'
   1          COMMON/TRCNTRL/NDEG,NSAMP,TSTRT_REF,TEND_REF,NSKIP,MAXIT,EPSILON
   1        . ,NCHAN,TRUNC,NSTATE,C
              INCLUDE 'TRWORK.CMN'
   1          DOUBLE PRECISION ZZ,DX,F,FX,BB,RHOSQ
   1          COMMON/TRWORK/SV(3,0:10),F,FX(31),BB(31),RHO,RHOSQ,ZZ(961),ITCOUNT,TS
   1        . ,DX(31),X(31),COV(4,4),DETCOV
              INCLUDE 'TRTIMS.CMN'
   1          COMMON/TRTIMS/FILE_REF(32),CFQ_REF,BSZ_REF,SR_REF,T_REF,POS_REF(3)
   1        . ,FNS_REF,SNR_REF,SIGSNR_REF
   1        . ,FILE_RES(32,5),CFQ_RES(5),BSZ_RES(5),SR_RES(5),T_RES(5)
   1        . ,POS_RES(3,5),FNS_RES(5),SNR_RES(5),SIGSNR_RES(5),COH(5),BIAS(5)
   1        . ,SIGSQCOH(5),ZRESBUF(0:4200),T_BUF
   1          BYTE FILE_REF,FILE_RES
   1          COMPLEX ZRESBUF
              DATA C/4900./
              DIMENSION P(3),V(3)
              DATA LN/21/,IRCMX/256/,EPSILON/1.E-5/
              CALL QAI(5,' NO OF CHANNELS = ',NCHAN)
              DO 10 N=1,NCHAN
                      CALL TEXTI(5,'$CHANNEL',N)
                      CALL QAA(5,'+  TIME SERIES INPUT FILE: ',FILE_RES(1,N))
                      CALL TEXTI(5,'$CHANNEL',N)
                      CALL QAR(5,'+  SNR(DB) = ',SNR_RES(N))
                      CALL TEXTI(5,'$CHANNEL',N)
                      CALL QAR(5,'+ SIGMA SNR(DB) = ',SIGSNR_RES(N))
     10       CONTINUE
              CALL TEXT(5,' ')
              CALL TEXT(5,' CHANNEL   FRQ.      BINSZ    SRATE    STIME
            .    NSAMP')
              DO 20 N=1,NCHAN
                      CALL GTSHDR(LN,FILE_RES(1,N),IRCMX,FNS_RES(N),BSZ_RES(N),
            .         SR_RES(N),TSSR,CFQ_RES(N),T_RES(N),POS_RES(1,N),IERR)
                      WRITE(5,1300)N,CFQ_RES(N),BSZ_RES(N),SR_RES(N),T_RES(N)
            .         ,FNS_RES(N)
                      POS_RES(1,N)=POS_RES(1,N)*6076.
                      POS_RES(2,N)=POS_RES(2,N)*6076.
                      POS_RES(3,N)=POS_RES(3,N)*6076.
     20       CONTINUE
   1300       FORMAT(I5,F12.3,F10.3,F10.3,F7.0,F10.0)
              CALL TEXT(5,' ')
              CALL QAR(5,' STARTING TARGET ESTIMATION TIME(SEC) = ',TSTRT_REF)
              CALL QAR(5,' ENDING TARGET ESTIMATION TIME(SEC) = ',TEND_REF)
              CALL QAI(5,' TIME SKIP BETWEEN ESTIMATIONS(SEC) = ',TSKIP)
              CALL QAI(5,' NO OF SAMPLES PER INTEGRATION = ', NSAMP)
              CALL QAI(5,' DEGREE OF POSITION VS TIME POLYNOMIALS = ',NDEG)
              CALL QAR(5,' SINC FUNCTION TRUNCATION PT = ',TRUNC)
              CALL QAR(5,' INITIAL X-COORDINATE(FT) =       ',XINIT)
              CALL QAR(5,' INITIAL Y-COORDINATE(FT) =       ',YINIT)
              CALL QAR(5,' INITIAL Z-COORDINATE(DEPTH-FT) = ',ZINIT)
              CALL QAR(5,' INITIAL SPEED(KTS) =             ',VINITK)
              VINIT=VINITK*6076.029/3600.
              CALL QAR(5,' INITIAL HEADING(DEG) =           ',BINIT)
              DO 30 N=0,NDEG
                      SV(1,N)=0.
                      SV(2,N)=0.
```

```
                       SV(3,N)=0.
        30     CONTINUE
               SV(1,0)=XINIT
               SV(2,0)=YINIT
               SV(3,0)=ZINIT
               SV(1,1)=VINIT*SIND(BINIT)
               SV(2,1)=VINIT*COSD(BINIT)
               NSTATE=2*NDEG+2
               TEST=TSTRT_REF
               DO 1000 WHILE(TEST.LE.TEND_REF)
                       CALL TRACK(TEST,ZZ,BB,IERR)
                       IF(IERR.EQ.0)CALL TRERREF(IERR)
                       CALL TRUPDAT(P,V,TEST)
                       CALL TEXT(5,' ')
                       CALL TEXTR(5,' TS(SEC) = ',TEST)
                       CALL TEXTR(5,' X(TS) = ',P(1))
                       CALL TEXTR(5,' Y(TS) = ',P(2))
                       SPEED=SQRT(V(1)**2+V(2)**2)*3600./6076.
                       COURSE=ATAN2D(V(1),V(2))
                       CALL TEXTR(5,' SPEED(KTS) = ',SPEED)
                       CALL TEXTR(5,' COURSE(DEG) = ',COURSE)
                       CALL TRUPDAT(P,V,TEST+TSKIP)
                       SV(1,0)=P(1)
                       SV(2,0)=P(2)
                       SV(3,0)=P(3)
                       SV(1,1)=V(1)
                       SV(2,1)=V(2)
                       SV(3,1)=V(3)
                       DO 40 K=2,NDEG
                               SV(1,N)=0
                               SV(2,N)=0
                               SV(3,N)=0
        40             CONTINUE
                       TEST=TEST+TSKIP
        1000   CONTINUE
               STOP
               END
```

```
              SUBROUTINE TRUPDAT(P,V,TIME)
              INCLUDE 'TRWORK.CMN'
1             DOUBLE PRECISION ZZ,DX,F,FX,BB,RHOSQ
1             COMMON/TRWORK/SV(3,0:10),F,FX(31),BB(31),RHO,RHOSQ,ZZ(961),ITCOUNT,TS
1         .   ,DX(31),X(31),COV(4,4),DETCOV
              INCLUDE 'TRCNTRL.CMN'
1             COMMON/TRCNTRL/NDEG,NSAMP,TSTRT_REF,TEND_REF,NSKIP,MAXIT,EPSILON
1         .   ,NCHAN,TRUNC,NSTATE,C
              DIMENSION P(3),V(3)
              TI=TIME-TS
              DO 10 N=1,3
                      P(N)=SV(N,NDEG)
                      V(N)=0.
     10       CONTINUE
              DO 30 M=NDEG-1,0,-1
                      MP1=M+1
              DO 20 N=1,3
                      P(N)=P(N)*TI+SV(N,M)
                      V(N)=V(N)*TI+(MP1)*SV(N,MP1)
     20       CONTINUE
     30       CONTINUE
              RETURN
              END
```

```
              SUBROUTINE TRACK(TT,Z,B,IERR)
              INCLUDE 'TRCNTRL.CMN'
 1            COMMON/TRCNTRL/NDEG,NSAMP,TSTRT_REF,TEND_REF,NSKIP,MAXIT,EPSILON
 1        .   ,NCHAN,TRUNC,NSTATE,C
              INCLUDE 'TRWORK.CMN'
 1            DOUBLE PRECISION ZZ,DX,F,FX,BB,RHOSQ
 1            COMMON/TRWORK/SV(3,0:10),F,FX(31),BB(31),RHO,RHOSQ,ZZ(961),ITCOUNT,TS
 1        .   ,DX(31),X(31),COV(4,4),DETCOV
              INCLUDE 'TRTIMS.CMN'
 1            COMMON/TRTIMS/FILE_REF(32),CFQ_REF,BSZ_REF,SR_REF,T_REF,POS_REF(3)
 1        .   ,FNS_REF,SNR_REF,SIGSNR_REF
 1        .   ,FILE_RES(32,5),CFQ_RES(5),BSZ_RES(5),SR_RES(5),T_RES(5)
 1        .   ,POS_RES(3,5),FNS_RES(5),SNR_RES(5),SIGSNR_RES(5),COH(5),BIAS(5)
 1        .   ,SIGSQCOH(5),ZRESBUF(0:4200),T_BUF
 1            BYTE FILE_REF,FILE_RES
 1            COMPLEX ZRESBUF
              DOUBLE PRECISION FT,Z,DET,AWK,BWK,B,RFACT,ZV,ZW
              DIMENSION Z(NSTATE,NSTATE),B(NSTATE)
              DIMENSION P(3),V(3),UO(3),UN(3),DP(3),DT(3),AWK(31),BWK(31)
              COMPLEX ZREF(4000),ZRES(4000),DZRESDT,TWOPIJ
              COMPLEX FTC,ZEXP,DZRESDTH
              COMPLEX*16 ZU,ZDW(33),ZDU(33),DFACT,CDOT
              DATA NBUFREF/4000/,NBUFRES/4500/,TWOPIJ/(0.,6.283185307)/,LN/21/
              DATA IRCMX/256/,MXLOOP/20/,ALN10/.230258509/
              TS=TT
              SFACT=(NSAMP-1.)/(NSAMP**2+NSAMP)
              DO 5 I=0,NDEG
                      X(I+1)=SV(1,I)
                      X(I+NDEG+2)=SV(2,I)
   5          CONTINUE
   C                  X(NSTATE)=SV(3,0)
              ILOOP=0
              GWGOLD=1.E30
   10         DO 14 I=1,NSTATE
                      B(I)=0.
              DO 14 J=1,I
                      Z(I,J)=0.
                      Z(J,I)=0.
   14         CONTINUE
              GWG=0.
              DO 700 NREF=1,NCHAN-1
              CALL GTSHDR(LN,FILE_RES(1,NREF),IRCMX,FNS_REF,BSZ_REF,SR_REF,TSSR,
          .   CFQ_REF,T_REF,POS_REF,IERR)
              POS_REF(1)=POS_REF(1)*6076.
              POS_REF(2)=POS_REF(2)*6076.
              POS_REF(3)=POS_REF(3)*6076.
              SNR_REF=SNR_RES(NREF)
              SIGSNR_REF=SIGSNR_RES(NREF)
              PCCALC=1./(1.00001+10.**(-SNR_REF/10.))
              DELTA_T=1./SR_REF
              TINT=NSAMP*DELTA_T
              NMIN=MAX(TRUNC,(TT-TINT/2.-T_REF)*SR_REF)
              CALL GTSDAT(LN,FILE_RES(1,NREF),IRCMX,NMIN,NSAMP,ZREF,IERR)
              CALL HET(ZREF,1.,NSAMP,CFQ_REF,SR_REF)
              ZV=CDOT(ZREF,ZREF,NSAMP)
              DO 700 NC=NREF+1,NCHAN
                      COH(NC)=PCCALC/(1.00001+10.**(-SNR_RES(NC)/10.))
```

```
                BIAS(NC)=(1.-COH(NC))**2*(1.+2.*COH(NC)/NSAMP)/NSAMP
                SIGSQREF=(ALN10*COH(NC)/(1.+10.**(SNR_REF/10.))*SIGSNR_REF)**2
                SIGSQREF=(ALN10*COH(NC)/(1.+10.**(SNR_RES(NC)/10.))
     .          *SIGSNR_RES(NC))**2
                SIGSQCOH(NC)=SIGSQREF+SIGSQRES
 30     DO 40 N=1,NSTATE
                ZDW(N)=(0.,0.)
                ZDU(N)=(0.,0.)
 40     CONTINUE
        F=0.
        ZU=(0.,0.)
        ZW=(0.,0.)
        ZNUM=(0.,0.)
        TF=T_REF+NMIN/SR_REF
        CALL TRUPDAT(P,V,TF)
        TG=TF-ABSRVSM(1.,P,-1.,POS-REF,3)/C
        ANORRES=0.
        ZEXP=(1.,0.)
        CALL TOT1(TAUBAR,TW,TG,TF,POS_REF,POS_RES(1,NC),EPSILON,ERR)
 50     DO 600 NRF=1,NSAMP
                CALL TRUPDAT(P,V,TW)
                CALL RSMADD(UO,1.,P,-1.,POS_REF,3)
                CALL RSMADD(UN,1.,P,-1.,POS_RES(1,NC),3)
                DO=ABSRV(UO,3)
                DN=ABSRV(UN,3)
                CALL RVSCM(UO,1./DO,UO,3)
                CALL RVSCM(UN,1./DN,UN,3)
                CALL XN_DXN(TAUBAR,NC,ZRES(NRF),DZRESDT)
                ZRES(NRF)=ZRES(NRF)*ZEXP
                DFACT=TWOPIJ*CFQ_RES(NC)*ZRES(NRF)+ZEXP*DZRESDT
                DSNDTK=RDOT(V,UN,3)/C
                DSODTK=RDOT(V,UO,3)/C
                RFACT=(1.D0+DSNDTK)/(1.D0+DSODTK)
                DO 60 NS=1,NSTATE
                        CALL DPDTH(DP,TW,NS)
                        DSNDTH=RDOT(UN,DP,3)/C
                        DSODTH=RDOT(UO,DP,3)/C
                        DTAUDTH=-DSODTH*RFACT+DSNDTH
                        DZRESDTH=DFACT*DTAUDTH
                        ZDW(NS)=ZDW(NS)+DZRESDTH*CONJG(ZRES(NRF))
                        ZDU(NS)=ZDU(NS)+DZRESDTH*CONJG(ZREF(NRF))
 60             CONTINUE
                ZW=ZW+ZRES(NRF)*CONJG(ZRES(NRF))
                ZU=ZU+ZREF(NRF)*CONJG(ZRES(NRF))
                TAULAST=TAUBAR
                TG=TW+DELTA_T
                TF=TF+DELTA_T
                CALL TOT1(TAUBAR,TW,TG,TF,POS_REF,POS_RES(1,NC),EPSILON,ERR)
                FT=CFQ_RES(NC)*(TAUBAR-TAULAST)
                FTC=MOD(FT,1.D0)
                ZEXP=ZEXP*CEXP(TWOPIJ*FTC)
 600    CONTINUE
        RHOSQ=ZU*CONJG(ZU)
        RHOSQ=RHOSQ/(ZV*ZW)
        F=LOG((COH(NC)+BIAS(NC))/RHOSQ)
        WF=(COH(NC)+BIAS(NC))**2/SIGSQCOH(NC)
        GWG=GWG+WF*F**2
```

```
              DO 650 NS=1,NSTATE
                      FX(NS)=2.*DREAL(ZDW(NS))/ZW-2.*DREAL(ZDU(NS)/CONJG(ZU))
      650     CONTINUE
              DO 680 I=1,NSTATE
                      B(I)=B(I)+WF*F*FX(I)
              DO 670 J=1,I
                      Z(I,J)=Z(I,J)+FX(I)*FX(J)*WF
                      Z(J,I)=Z(I,J)
      670     CONTINUE
      680     CONTINUE
      700     CONTINUE
              CALL DMINV(Z,NSTATE,DET,AWK,BWK)
              CHK=0.
              DO 720 I=1,NSTATE
      C               DX(I)=0.
      C               DO 710 K=1,NSTATE
      C                       DX(I)=DX(I)-Z(I,K)*B(K)
      C710            CONTINUE
                      CHK=CHK+B(I)*B(I)
      720     CONTINUE
              ABSB=SQRT(CHK)
              DO 725 I=1,NSTATE
                      DX(I)=-.5*GWG*B(I)/ABSB
      725     CONTINUE
              IF(GWG.GE.GWGOLD)GO TO 810
              DO 730 I=0,NDEG
                      X(I+1)=X(I+1)+DX(I+1)
                      X(I+NDEG+2)=X(I+NDEG+2)+DX(I+NDEG+2)
                      SV(1,I)=X(I+1)
                      SV(2,I)=X(I+NDEG+2)
      730     CONTINUE
      C       SV(3,0)=X(NSTATE)
              GWGOLD=GWG
              ILOOP=ILOOP+1
      800     IF(ILOOP.LE.MXLOOP)GO TO 10
              IERR=1
              RETURN
      810     IERR=0
              RETURN
              END
```

```
        SUBROUTINE TOT1(T1,TW,TG,TO,PO,P1,EPSLN,ERR)
  C
  C     THIS SUBROUTINE SOLVES FOR THE RECEIVER TIME, T1, ON CHANNEL 1,
  C     WHICH CORRESPONDS TO THE RECEIVER TIME,TO,ON CHANNEL 0. THE
  C     CURRENT VALUE OF THE STATE VECTOR IS USED TO MAKE THESE CALCU-
  C     LATIONS. PO AND P1 ARE THE POSITION VECTORS OF STATIONS 0 AND 1,
  C     RESPECTIVELY. TG IS THE INITIAL GUESS AT THE INTERMEDIATE WATER
  C     TIME,TW. THE SUBROUTINE RETURNS T1,TW,AND ERR. THE VARIABLE
  C     EPSLN IS THE TOLERENCE IN SECONDS FOR THE CONVERGENCE CRITERION.
  C     THE SUBROUTINE WILL ITERATE AT MOST 20 TIMES IF CONVERGENCE IS
  C     NOT MET. IT WILL THEN SIGNAL WITH AN ERROR MESSAGE TO THE OPERATOR
  C     TERMINAL AND SET ERR = 1. OTHERWISE THE SUBROUTINE WILL RETURN WITH
  C     ERR = 0.
  C
        INCLUDE '[ELDRIDGE]TRCNTRL.CMN'
1       COMMON/TRCNTRL/NDEG,NSAMP,TSTRT_REF,TEND_REF,NSKIP,MAXIT,EPSILON
1       ,NCHAN,TRUNC,NSTATE,C
        DIMENSION P(3),PO(3),P1(3),V(3),U(3)
        ERR=0.
        TL=TG
        DO 10 N=1,20
                CALL TRUPDAT(P,V,TL)
                G=ABSRVSM(1.,P,-1.,PO,3)
                CALL RSMADD(U,1.,P,-1.,PO,3)
                CALL RVSCM(U,1./G,U,3)
                F=G+C*(TL-TO)
                FP=C+RDOT(U,V,3)
                DELT=-F/FP
                TL=TL+DELT
                IF(ABS(DELT).LE.EPSLN)GO TO 20
   10   CONTINUE
        CALL TEXT(5,' NO CONVERGENCE AFTER 20 ITERATIONS')
        ERR=1.
        RETURN
  C
  C     NOW SET TW = TL AND PLUG THIS VALUE INTO THE RECEIVER TIME
  C     EQUATION  FOR CHANNEL 1 TO OBTAIN T1
  C
   20   TW=TL
        CALL TRUPDAT(P,V,TW)
        G1=ABSRVSM(1.,P,-1.,P1,3)
        T1=TW+G1/C
        RETURN
        END
```

```
      FUNCTION ABSRV(F,N)
C
C     THIS FUNCTION COMPUTES THE EUCLIDEAN NORM OF AN N-DIMENSIONAL
C     REAL VECTOR F
C
      DIMENSION F(1)
      ABSRV=0.
      DO 10 I=1,N
          ABSRV=ABSRV+F(I)*F(I)
10    CONTINUE
      ABSRV=SQRT(ABSRV)
      RETURN
      END
```

```
      FUNCTION ABSRVSM(S1,F1,S2,F2,N)
C
C     THIS FUNCTION COMPUTES THE ABSOLUTE VALUE OF THE REAL
C     VECTOR
C             S1*F1+S2*F2
C     WHERE S1 AND S2 ARE REAL SCALORS AND F1 AND F2 ARE REAL
C     N-DIMENSIONAL VECTORS.
C
      DIMENSION F1(1),F2(1)
      ABSRVSM=0.
      DO 10 I=1,N
              T=S1*F1(I)+S2*F2(I)
              ABSRVSM=ABSRVSM+T*T
   10 CONTINUE
      ABSRVSM=SQRT(ABSRVSM)
      RETURN
      END
```

```
      SUBROUTINE RVSCM(F,S1,F1,N)
C
C     THIS SUBROUTINE PERFORMS A REAL SCALOR MULTIPLY
C             F=S1*F1
C     WHERE F,F1 ARE N-DIMENSIONAL REAL VECTORS AND S1 IS A
C     REAL SCALOR.
C
      DIMENSION F(1),F1(1)
      DO 10 I=1,N
              F(I)=S1*F1(I)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FCTSDAT(LN,DFTFIL,IRCMX,NMIN,NSAMP,Z,IERR)
      ENTRY FTSDAT(LN,DFTFIL,IRCMX,NMIN,NSAMP,Z,IERR)
C
C     THIS SUBROUTINE WRITES COMPLEX DATA TO FILES COMPATIBLE WITH
C     BBGENT. THE PROGRAM WRITES THE FIRST NSAMP VALUES
C     OF THE COMPLEX ARRAY Z INTO NSAMP COMPLEX
C     SAMPLES STARTING WITH THE ANMIN'TH SAMPLE IN THE FILE.
C
C
C     THIS SUBROUTINE ASSUMES IRCMX COMPLEX DFT's  PER RECORD.
C
      IMPLICIT COMPLEX (Z)
      DIMENSION Z(1),ZBUF(256)
      BYTE DFTFIL(1)
      DATA IHEADER/1/,MAXZBUF/256/
C
      IF(IRCMX.GT.MAXZBUF)GO TO 30
      CALL CLOSE(LN)
      OPEN(UNIT=LN,FILE=DFTFIL,STATUS='UNKNOWN',RECL=IRCMX*2,ACCESS='DIRECT',
     .     BLOCKSIZE=IRCMX*8,ASSOCIATEVARIABLE=IR)
      IERR=0
      RCMX=IRCMX
      ANMIN=NMIN+1
      ISREC=NMIN/RCMX+IHEADER+1
      ANMAX=NMIN+NSAMP
      IEREC=(ANMAX-1.)/RCMX+IHEADER+1
      ADATA1=ANMIN
      N1=0
      IR=ISREC
      DO 10 IREC=ISREC,IEREC
              IBUF1=AMOD(ADATA1-1.,RCMX)+1
              ADATA2=AMIN1((IREC-IHEADER)*RCMX,ANMAX)
              IBUF2=AMOD(ADATA2-1.,RCMX)+1
              IF(IBUF2-IBUF1+1.EQ.IRCMX)GO TO 4
              READ(LN'IR,ERR=3)(ZBUF(I),I=1,IRCMX)
3             IR=IREC
4             DO 5 I=IBUF1,IBUF2
                      N1=N1+1
                      ZBUF(I)=Z(N1)
5             CONTINUE
              WRITE(LN'IR,ERR=20)(ZBUF(I),I=1,IRCMX)
              ADATA1=ADATA1+IBUF2-IBUF1+1
10    CONTINUE
      CALL CLOSE(LN)
      RETURN
20    CALL TEXT(5,' FTSDAT/FCTSDAT: ERROR WRITING FILE! ')
      IERR=1
      CALL CLOSE(LN)
      RETURN
30    CALL TEXTI(5,' FTSDAT/FCTSDAT:IRCMX EXCEDES MAXZBUF. IRCMX = ',IRCMX)
      CALL TEXTI(5,' CURRENT VALUE OF MAXZBUF = ',MAXZBUF)
      CALL TEXT(5,' IF YOU INCREASE MAXZBUF,MAKE SURE YOU INCREASE THE')
      CALL TEXT(5,' DIMENSION OF ZBUF ACCORDINGLY.')
      STOP
      END
```

```
C        ---------------------------------------------------=================
         SUBROUTINE GCTSDAT(LN,DFTFIL,IRCMX,NMIN,NSAMP,Z,IERR)
         ENTRY GTSDAT(LN,DFTFIL,IRCMX,NMIN,NSAMP,Z,IERR)
C
C        THIS SUBROUTINE OBTAINS COMPLEX DATA FROM FILES GENERATED
C        BBGENT. THE PROGRAM RETURNS WITH THE FIRST NSAMP VALUES
C        OF THE COMPLEX ARRAY Z LOADED FROM THE NSAMP COMPLEX
C        SAMPLES STARTING WITH THE ANMIN'TH SAMPLE IN THE FILE. THE FIRST
C        SAMPLE IN THE FILE IS NUMBERED 0 TO THE USER
C
C
C        THIS SUBROUTINE ASSUMES IRCMX COMPLEX DFT's  PER RECORD.
C
         IMPLICIT COMPLEX (Z)
         DIMENSION Z(1),ZBUF(256)
         BYTE DFTFIL(1)
         DATA IHEADER/1/,MAXZBUF/256/
C
         IF(IRCMX.GT.MAXZBUF)GO TO 30
         CALL CLOSE(LN)
         OPEN(UNIT=LN,FILE=DFTFIL,STATUS='OLD',RECL=IRCMX*2,ACCESS='DIRECT',
     .        BLOCKSIZE=IRCMX*8,ASSOCIATEVARIABLE=IR)
         IERR=0
         RCMX=IRCMX
         ANMIN=NMIN+1
         ISREC=NMIN/RCMX+IHEADER+1
         ANMAX=NMIN+NSAMP
         IEREC=(ANMAX-1.)/RCMX+IHEADER+1
         IR=ISREC
         ADATA1=ANMIN
         N1=0
         DO 10 IREC=ISREC,IEREC
                 READ(LN'IR,ERR=20)(ZBUF(I),I=1,IRCMX)
                 IBUF1=AMOD(ADATA1-1.,RCMX)+1
                 ADATA2=AMIN1((IREC-IHEADER)*RCMX,ANMAX)
                 IBUF2=AMOD(ADATA2-1.,RCMX)+1
                 DO 5 I=IBUF1,IBUF2
                         N1=N1+1
                         Z(N1)=ZBUF(I)
5                CONTINUE
                 ADATA1=ADATA1+IBUF2-IBUF1+1
10       CONTINUE
         CALL CLOSE(LN)
         RETURN
20       CALL TEXT(5,' GTSDAT/GCTSDAT: ERROR READING FILE: ')
         IERR=1
         CALL CLOSE(LN)
         RETURN
30       CALL TEXTI(5,' GTSDAT/GCTSDAT:IRCMX EXCEDES MAXZBUF. IRCMX = ',IRCMX)
         CALL TEXTI(5,' CURRENT VALUE OF MAXZBUF = ',MAXZBUF)
         CALL TEXT(5,' IF YOU INCREASE MAXZBUF,MAKE SURE YOU INCREASE THE')
         CALL TEXT(5,' DIMENSION OF ZBUF ACCORDINGLY.')
         STOP
         END
```

```
C=============================================================================
        SUBROUTINE FTSHDR(LN,DFTFIL,IRCMX,TDFTS,BINSZE,SSR,TSSR,
     .  BINFRQ,TDEL,FVEC,IERR)
        BYTE DFTFIL(1)
        DIMENSION FVEC(3)
C
C       BINSZE = BIN SIZE(HZ)
C       BINFRQ = CENTER FREQUENCY(HZ) OF I'TH BIN
C       SSR  = SPECTRAL SAMPLE RATE
C       TSSR = TIME SERIES SAMPLE RATE(HZ)
C       IRCMX =        COMPLEX RECORD SIZE (256)
C       TDFTS=         TOTAL # OF DFT COEFF.S (samples)
C       TDEL  =        channel delay
C       FVEC(3) =      POSITION VECTOR OF SENSOR
C
C       ======================================================================
C
C
        CALL CLOSE(LN)
        OPEN(UNIT=LN,FILE=DFTFIL,STATUS='UNKNOWN',RECL=IRCMX*2,ACCESS='DIRECT',
     .      BLOCKSIZE=IRCMX*8,ASSOCIATEVARIABLE=IR)
C
        IR=1
        IERR=0
        WRITE(LN'IR,ERR=900)IRCMX,TDFTS,BINSZE,SSR,TSSR,BINFRQ,TDEL,FVEC
        GO TO 999
C
900     CALL TEXT(5,' FTSHDR: WRITE ERROR')
        IERR=1           ! write error
999     CONTINUE
        CALL CLOSE(LN)
        RETURN
        END
```

```
      SUBROUTINE GTSHDR(LN,DFTFIL,IRCMX,TDFTS,BINSZE,SSR,TSSR,
     . BINFRQ,TDEL,PVEC,IERR)
      BYTE DFTFIL(1)
      DIMENSION PVEC(3)
C
C     BINSZE = BIN SIZE(HZ)
C     BINFRQ = CENTER FREQUENCY(HZ) OF I'TH BIN
C     SSR  = SPECTRAL SAMPLE RATE
C     TSSR = TIME SERIES SAMPLE RATE(HZ)
C     IRCMX =       COMPLEX RECORD SIZE (256)
C     TDFTS=        TOTAL # OF DFT COEFF.S (samples)
C     TDEL  =       channel delay
C     PVEC(3) =     POSITION VECTOR OF SENSOR
C
C     =====================================================================
C
C
      CALL CLOSE(LN)
      OPEN(UNIT=LN,FILE=DFTFIL,STATUS='OLD',RECL=IRCMX*2,ACCESS='DIRECT',
     .     BLOCKSIZE=IRCMX*8,ASSOCIATEVARIABLE=IR)
C
      IR=1
      IERR=0
      READ(LN'IR,ERR=900)IRCMX,TDFTS,BINSZE,SSR,TSSR,BINFRQ,TDEL,PVEC
      GO TO 999
C
900   CALL TEXT(5,' GTSHDR: READ ERROR')
      IERR=1          ! read error
999   CONTINUE
      CALL CLOSE(LN)
      RETURN
      END
```

```
      SUBROUTINE TRERREF(I)
C
C     THIS SUBROUTINE IS A DUMMY(AS IS ITS AUTHOR!).
C
      RETURN
      END
```

```
              SUBROUTINE DPDTH(D,T1,K)
              DIMENSION D(3)
              INCLUDE 'TRCNTRL.CMN'
  1           COMMON/TRCNTRL/NDEG,NSAMP,TSTRT_REF,TEND_REF,NSKIP,MAXIT,EPSILON
  1       .   ,NCHAN,TRUNC,NSTATE,C
              INCLUDE 'TRWORK.CMN'
  1           DOUBLE PRECISION ZZ,DX,F,FX,BB,RHOSQ
  1           COMMON/TRWORK/SV(3,0:10),F,FX(31),BB(31),RHO,RHOSQ,ZZ(961),ITCOUNT,TS
  1       .   ,DX(31),X(31),COV(4,4),DETCOV
              K1=K-1
              T=T1-TS
              IF(K1.LE.NDEG)THEN
                      D(1)=T**K1
                      D(2)=0.
                      D(3)=0.
              ELSE
              IF(K1.LE.2*NDEG+1)THEN
                      D(1)=0.
                      D(2)=T**(K1-NDEG-1)
                      D(3)=0.
              ELSE
                      D(1)=0.
                      D(2)=0.
                      D(3)=1.
              ENDIF
              ENDIF
              RETURN
              END
```

```
      FUNCTION DSINCDT(X)
      DOUBLE PRECISION DSINCDT,X
      IF(X.EQ.0.D0)THEN
              DSINCDT=0.D0
      ELSE
              DSINCDT=COS(X)/X-SIN(X)/X**2
      ENDIF
      RETURN
      END
```

```
      FUNCTION SINC(X)
      DOUBLE PRECISION SINC,X
      IF(X.EQ.0.D0)THEN
              SINC=1.D0
      ELSE
              SINC=SIN(X)/X
      ENDIF
      RETURN
      END
```

```
              SUBROUTINE XN_DXN(TBAR,N,ZXN,DZXNDT)
              INCLUDE '[ELDRIDGE]TRTIMS.CMN'
  1           COMMON/TRTIMS/FILE_REF(32),CFQ_REF,BSZ_REF,SR_REF,T_REF,POS_REF(3)
  1       .   ,FNS_REF,SNR_REF,SIGSNR_REF
  1       .   ,FILE_RES(32,5),CFQ_RES(5),BSZ_RES(5),SR_RES(5),T_RES(5)
  1       .   ,POS_RES(3,5),FNS_RES(5),SNR_RES(5),SIGSNR_RES(5),COH(5),BIAS(5)
  1       .   ,SIGSQCOH(5),ZRESBUF(0:4200),T_BUF
  1           BYTE FILE_REF,FILE_RES
  1           COMPLEX ZRESBUF
              INCLUDE '[ELDRIDGE]TRCNTRL.CMN'
  1           COMMON/TRCNTRL/NDEG,NSAMP,TSTRT_REF,TEND_REF,NSKIP,MAXIT,EPSILON
  1       .   ,NCHAN,TRUNC,NSTATE,C
              COMPLEX ZXN,DZXNDT,ZBUF(100)
              COMPLEX*16 DZXN,DDZXNDT
              DOUBLE PRECISION PI,SAMPN,ARG,SIGMA,SINC,DSINCDT
              DATA PI/3.14159265358979300/,LN/21/,IRCMX/256/
              TRUNC2=2*TRUNC
              SIGMA=PI*SR_RES(N)
              TD=TBAR-T_RES(N)
              SAMPN=TD*SR_RES(N)
              NMIN=SAMPN-TRUNC+1
              NMAX=NMIN+TRUNC2-1
              NPTS=NMAX-NMIN+1
              CALL GTSDAT(LN,FILE_RES(1,N),IRCMX,NMIN,NPTS,ZBUF,IERR)
              NN=1
              DZXN=(0.,0.)
              DDZXNDT=(0.,0.)
              DO 10 NS=NMIN,NMAX
                      ARG=(SAMPN-NS)*PI
                      DZXN=DZXN+SINC(ARG)*ZBUF(NN)
                      DDZXNDT=DDZXNDT+DSINCDT(ARG)*ZBUF(NN)
                      NN=NN+1
     10       CONTINUE
              ZXN=DZXN
              DZXNDT=SIGMA*DDZXNDT
              RETURN
              END
```

```
          SUBROUTINE CSMADD(Z,S1,Z1,S2,Z2,N)
C
C         THIS SUBROUTINE COMPUTES THE VECTOR SUM
C                 Z=S1*Z1+S2*Z2
C         WHERE Z,Z1,AND Z2 ARE COMPLEX N-DIMENTIONAL VECTORS AND
C         S1 AND S2 ARE COMPLEX SCALORS.
C
          COMPLEX Z,Z1,Z2,S1,S2
          DIMENSION Z(1),Z1(1),Z2(1)
          DO 10 I=1,N
                  Z(I)=S1*Z1(I)+S2*Z2(I)
   10     CONTINUE
          RETURN
          END
```

```
      SUBROUTINE RSMADD(F,S1,F1,S2,F2,N)
C
C     THIS SUBROUTINE COMPUTES THE VECTOR SUM
C           F=S1*F1+S2*F2
C     WHERE F,F1,AND F2 ARE REAL N-DIMENSIONAL VECTORS AND S1 AND S2
C     ARE REAL SCALORS
C
      DIMENSION F(1),F1(1),F2(1)
      DO 10 I=1,N
            F(I)=S1*F1(I)+S2*F2(I)
10    CONTINUE
      RETURN
      END
```

```
      COMPLEX FUNCTION CDOT*16(Z1,Z2,N)
C
C     THIS FUNCTION COMPUTES THE COMPLEX DOT PRODUCT BETWEEN THE
C     N-DIMENSIONAL COMPLEX VECTORS Z1 AND Z2
C
      COMPLEX Z1,Z2
      DIMENSION Z1(1),Z2(1)
      CDOT=(0.,0.)
      DO 10 I=1,N
              CDOT=CDOT+Z1(I)*CONJG(Z2(I))
 10   CONTINUE
      RETURN
      END
```

```
            FUNCTION RDOT(V1,V2,N)
      C
      C     THIS FUNCTION COMPUTES THE DOT PRODUCT BETWEEN THE REAL
      C     N-DIMENSIONAL VECTORS V1 AND V2.
      C
            DIMENSION V1(1),V2(1)
            RDOT=0.
            DO 10 I=1,N
                    RDOT=RDOT+V1(I)*V2(I)
      10    CONTINUE
            RETURN
            END
```

```
            SUBROUTINE CVSCM(Z,S1,Z1,N)
C
C           THIS SUBROUTINE PERFORMS A COMPLEX SCALOR MULTIPLY
C                   Z=S1*Z1
C           WHERE Z,Z1 ARE N-DIMENSIONAL COMZLEX VECTORS AND S1 IS A
C           COMPLEX SCALOR.
C
            COMPLEX Z,Z1,S1
            DIMENSION Z(1),Z1(1)
            DO 10 I=1,N
                    Z(I)=S1*Z1(I)
   10       CONTINUE
            RETURN
            END
```

```
C=======================================================================
      SUBROUTINE GRTSDAT(LN,DFTFIL,IRCMX,NMIN,NSAMP,B,IERR)
C
C     THIS SUBROUTINE OBTAINS REAL DATA FROM FILES GENERATED BY SUBROUTINE
C     PKTSDAT. THE PROGRAM RETURNS WITH THE FIRST NSAMP VALUES
C     OF THE REAL ARRAY B LOADED FROM THE NSAMP REAL
C     SAMPLES STARTING WITH THE ANMIN'TH SAMPLE IN THE FILE. THE FIRST
C     SAMPLE IN THE FILE IS NUMBERED 0 TO THE USER
C
C
C     THIS SUBROUTINE ASSUMES IRCMX REAL DATA SAMPLES  PER RECORD.
C
      DIMENSION B(1),BUF(256)
      BYTE DFTFIL(1)
      DATA IHEADER/1/,MAXBUF/256/
C
      IF(IRCMX.GT.MAXBUF)GO TO 30
      CALL CLOSE(LN)
      OPEN(UNIT=LN,FILE=DFTFIL,STATUS='OLD',RECL=IRCMX,ACCESS='DIRECT',
     .    BLOCKSIZE=IRCMX*4,ASSOCIATEVARIABLE=IR)
      IERR=0
      RCMX=IRCMX
      ANMIN=NMIN+1
      ISREC=NMIN/RCMX+IHEADER+1
      ANMAX=NMIN+NSAMP
      IEREC=(ANMAX-1.)/RCMX+IHEADER+1
      IR=ISREC
      ADATA1=ANMIN
      N1=0
      DO 10 IREC=ISREC,IEREC
              READ(LN'IR,ERR=20)(BUF(I),I=1,IRCMX)
              IBUF1=AMOD(ADATA1-1.,RCMX)+1
              ADATA2=AMIN1((IREC-IHEADER)*RCMX,ANMAX)
              IBUF2=AMOD(ADATA2-1.,RCMX)+1
              DO 5 I=IBUF1,IBUF2
                      N1=N1+1
                      B(N1)=BUF(I)
5             CONTINUE
              ADATA1=ADATA1+IBUF2-IBUF1+1
10    CONTINUE
      CALL CLOSE(LN)
      RETURN
20    CALL TEXT(5,' GRTSDAT: ERROR READING FILE: ')
      IERR=1
      CALL CLOSE(LN)
      RETURN
30    CALL TEXTI(5,' GRTSDAT:IRCMX EXCEDES MAXBUF. IRCMX = ',IRCMX)
      CALL TEXTI(5,' CURRENT VALUE OF MAXBUF = ',MAXBUF)
      CALL TEXT(5,' IF YOU INCREASE MAXBUF,MAKE SURE YOU INCREASE THE')
      CALL TEXT(5,' DIMENSION OF BUF ACCORDINGLY.')
      STOP
      END
```

```
C=============================================================================
      SUBROUTINE PRTSDAT(LN,DFTFIL,IRCMX,NMIN,NSAMP,B,IERR)
C
C     THIS SUBROUTINE WRITES REAL DATA TO FILES COMPATIBLE WITH
C     GRTSDAT. THE PROGRAM WRITES THE FIRST NSAMP VALUES
C     OF THE REAL ARRAY B INTO NSAMP REAL
C     SAMPLES STARTING WITH THE ANMIN'TH SAMPLE IN THE FILE.
C
C
C     THIS SUBROUTINE ASSUMES IRCMX REAL SAMPLES PER RECORD.
C
      DIMENSION B(1),BUF(256)
      BYTE DFTFIL(1)
      DATA IHEADER/1/,MAXBUF/256/
C
      IF(IRCMX.GT.MAXBUF)GO TO 30
      CALL CLOSE(LN)
      OPEN(UNIT=LN,FILE=DFTFIL,STATUS='UNKNOWN',RECL=IRCMX,ACCESS='DIRECT',
     .     BLOCKSIZE=IRCMX*4,ASSOCIATEVARIABLE=IR)
      IERR=0
      RCMX=IRCMX
      ANMIN=NMIN+1
      ISREC=NMIN/RCMX+IHEADER+1
      ANMAX=NMIN+NSAMP
      IEREC=(ANMAX-1.)/RCMX+IHEADER+1
      ADATA1=ANMIN
      N1=0
      IR=ISREC
      DO 10 IREC=ISREC,IEREC
              IBUF1=AMOD(ADATA1-1.,RCMX)+1
              ADATA2=AMIN1((IREC-IHEADER)*RCMX,ANMAX)
              IBUF2=AMOD(ADATA2-1.,RCMX)+1
              IF(IBUF2-IBUF1+1.EQ.IRCMX)GO TO 4
              READ(LN'IR,ERR=3)(BUF(I),I=1,IRCMX)
3             IR=IREC
4             DO 5 I=IBUF1,IBUF2
                      N1=N1+1
                      BUF(I)=B(N1)
5             CONTINUE
              WRITE(LN'IR,ERR=20)(BUF(I),I=1,IRCMX)
              ADATA1=ADATA1+IBUF2-IBUF1+1
10    CONTINUE
      CALL CLOSE(LN)
      RETURN
20    CALL TEXT(5,' PRTSDAT: ERROR WRITING FILE: ')
      IERR=1
      CALL CLOSE(LN)
      RETURN
30    CALL TEXTI(5,' PRTSDAT:IRCMX EXCEDES MAXBUF. IRCMX = ',IRCMX)
      CALL TEXTI(5,' CURRENT VALUE OF MAXBUF = ',MAXBUF)
      CALL TEXT(5,' IF YOU INCREASE MAXBUF,MAKE SURE YOU INCREASE THE')
      CALL TEXT(5,' DIMENSION OF BUF ACCORDINGLY.')
      STOP
      END
```

```
C=====================================================================================
        SUBROUTINE HET(Z,ANMIN,NSAMP,FRQ,SSR)
        IMPLICIT COMPLEX (Z)
        DIMENSION Z(1)
C
C       THIS SUBROUTINE PERFORMS COMPLEX HETRODYNING ON THE
C       INPUT COMPLEX SERIES CONTAINED IN THE ARRAY Z.
C
        DATA TWOPI/6.283185307/
        IF(FRQ.EQ.0.)RETURN
        PHIO=SIGN(1.,FRQ)*AMOD(ABS(FRQ)/SSR,1.)
        ZW=CEXP(CMPLX(0.,TWOPI*PHIO))
        PHIS=SIGN(1.,ANMIN*PHIO)*AMOD(ABS(ANMIN*PHIO),1.)
        ZS=CEXP(CMPLX(0.,TWOPI*PHIS))
        DO 10 N=1,NSAMP
        Z(N)=Z(N)*ZS
        ZS=ZS*ZW
10      CONTINUE
        RETURN
        END
```

COMMAND QUALIFIERS

```
  FORTRAN /NOOBJ/LIS/SHOW=(INCLUDE,NOMAP) TRDRIV1,TRUTILS1

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,INCLUDE,NOMAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19
```

COMPILATION STATISTICS

```
  Run Time:           18.35 seconds
  Elapsed Time:       39.47 seconds
  Page Faults:        23
  Dynamic Memory:     164 pages
```

```
        SUBROUTINE QAR(LUN,ITEXT,ANS)
        BYTE ITEXT(1),IZER
        DATA IZER/0/
5       DO 10 I=1,60
        IF(ITEXT(I).EQ.0)GO TO 20
10      CONTINUE
20      WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,60)
30      FORMAT(60A1,$)
        READ(LUN,40,ERR=5)ANS
40      FORMAT(G15.0)
        RETURN
        END
```

```
              SUBROUTINE QAX(LUN,ITEXT,ANS)
              BYTE ITEXT(1),IZER
              DATA IZER/0/
     5        DO 10 I=1,60
              IF(ITEXT(I).EQ.0)GO TO 20
    10        CONTINUE
    20        WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,60)
    30        FORMAT(60A1,$)
              READ(LUN,40,ERR=5)ANS
    40        FORMAT(Z6)
              RETURN
              END
```

```
          SUBROUTINE QAI(LUN,ITEXT,IANS)
          BYTE ITEXT(1),IZER
          DATA IZER/0/
   5      DO 10 I=1,60
          IF(ITEXT(I).EQ.0)GO TO 20
  10      CONTINUE
  20      WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,60)
  30      FORMAT(60A1,$)
          READ(LUN,40,ERR=5)IANS
  40      FORMAT(I15)
          RETURN
          END
```

```
      SUBROUTINE QAA(LUN,ITEXT,IANS)
      BYTE ITEXT(1),IZER,IANS(1),IBUF(40)
      DATA IZER/0/
5     DO 10 I=1,60
      IF(ITEXT(I).EQ.0)GO TO 20
10    CONTINUE
20    WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,60)
30    FORMAT(60A1,$)
      READ(LUN,40,ERR=5)NCHR,(IBUF(NC),NC=1,NCHR)
40    FORMAT(Q,40A1)
      IANS(NCHR+1)=0
      DO 50 I=1,NCHR
      IANS(I)=IBUF(I)
50    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TEXTR(LUN,ITEXT,ANS)
      BYTE ITEXT(1),IZER
      DATA IZER/0/
5     DO 10 I=1,60
      IF(ITEXT(I).EQ.0)GO TO 20
10    CONTINUE
20    WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,60),ANS
30    FORMAT(60A1,G12.4)
      RETURN
      END
```

```
         SUBROUTINE TEXTX(LUN,ITEXT,ANS)
         BYTE ITEXT(1),IZER
         DATA IZER/0/
   5     DO 10 I=1,60
         IF(ITEXT(I).EQ.0)GO TO 20
  10     CONTINUE
  20     WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,60),ANS
  30     FORMAT(60A1,Z8.8)
         RETURN
         END
```

```
      SUBROUTINE TEXT(LUN,ITEXT)
      BYTE ITEXT(1),IZER
      DATA IZER/0/
5     DO 10 I=1,72
      IF(ITEXT(I).EQ.0)GO TO 20
10    CONTINUE
20    WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,72)
30    FORMAT(72A1)
      RETURN
      END
```

```
      SUBROUTINE TEXTA(LUN,ITEXT,IANS)
      BYTE ITEXT(1),IZER,IANS(20)
      DATA IZER/0/
5     DO 10 I=1,52
      IF(ITEXT(I).EQ.0)GO TO 20
10    CONTINUE
20    WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,52),IANS
30    FORMAT(72A1)
      RETURN
      END
```

```
      BYTE ITEXT(1),IZER
      DATA IZER/0/
5     DO 10 I=1,60
      IF(ITEXT(I).EQ.0)GO TO 20
10    CONTINUE
20    WRITE(LUN,30)(ITEXT(I1),I1=1,I-1),(IZER,I1=I,60),IANS
30    FORMAT(60A1,I6)
      RETURN
      END
```

END

FILMED

9-85

DTIC