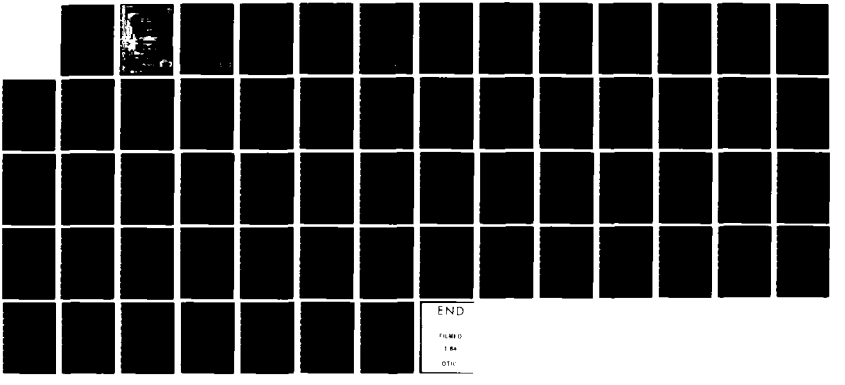
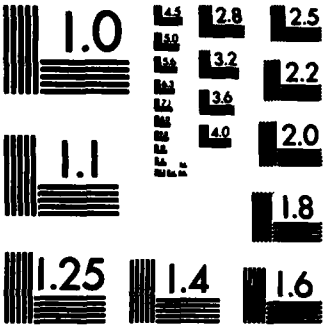
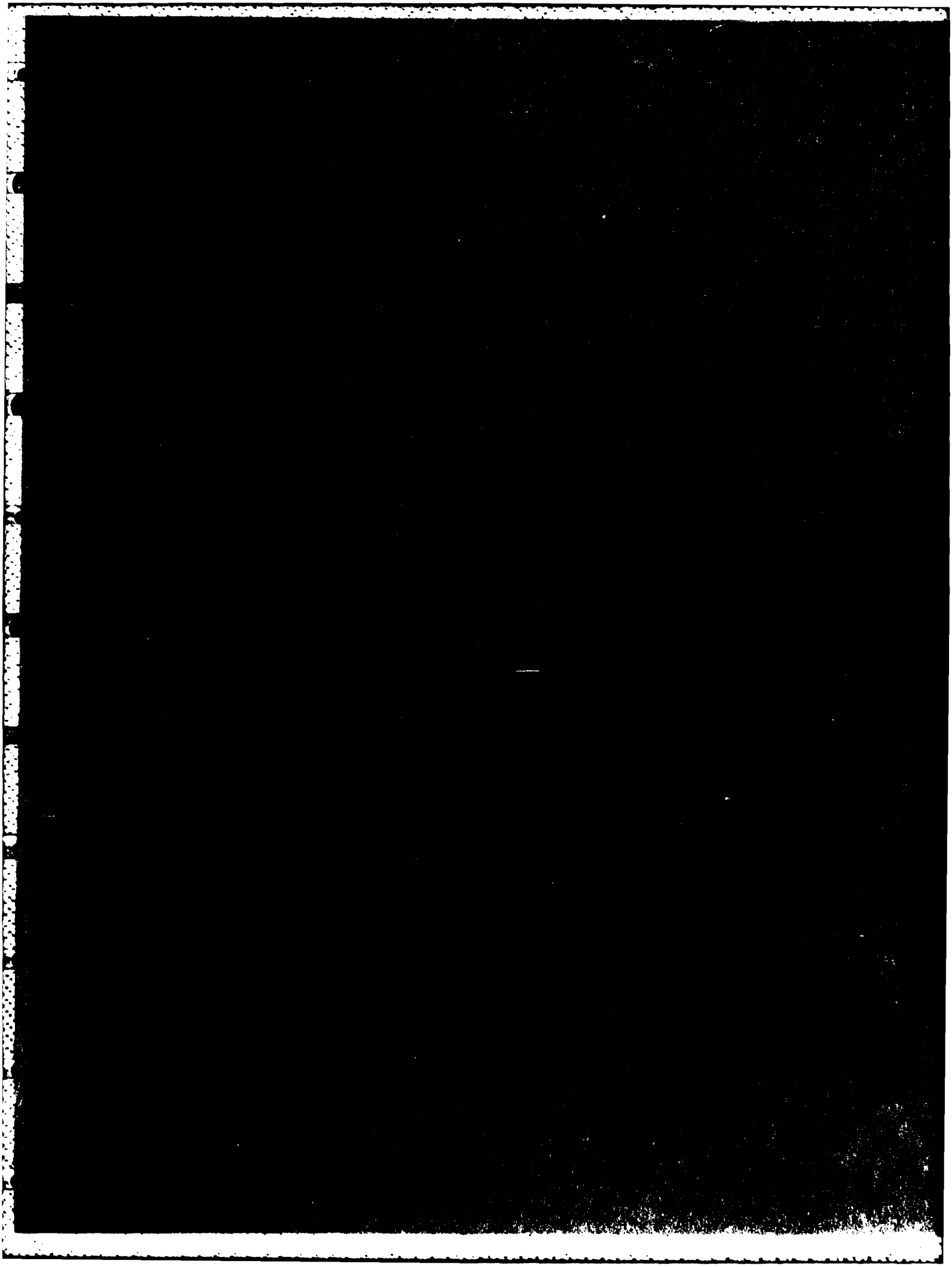
AD-A135 420 THE DISCRETE CONVOLUTION METHOD FOR SOLVING SOME LARGE MOMENT MATRIX EQUA. (U) SYRACUSE UNIV NY DEPT OF ELECTRICAL AND COMPUTER ENGINEERING. H L WYO ET AL. 1/1

UNCLASSIFIED JUL 83 SYRU/DECE/TR-83/14 N00014-76-C-0225 F/G 12/1 NL







SYRU/DECE/TR-83/14

THE DISCRETE CONVOLUTION METHOD FOR SOLVING SOME LARGE MOMENT MATRIX EQUATIONS

by

Htay L. Nyo
Roger F. Harrington

Department of
Electrical and Computer Engineering
Syracuse University
Syracuse, New York 13210

Technical Report No. 21
July 1983

Contract No. N00014-76-C-0225

Approved for public release; distribution unlimited

Prepared for

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
ARLINGTON, VIRGINIA 22217

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SYRU/DECE/TR-83/14	2 GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE DISCRETE CONVOLUTION METHOD FOR SOLVING SOME LARGE MOMENT MATRIX EQUATIONS		5. TYPE OF REPORT & PERIOD COVERED Technical Report No. 21 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Htay L. Nyo Roger F. Harrington		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0225
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dept. of Electrical & Computer Engineering Syracuse University Syracuse, New York 13210		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE July 1983 13. NUMBER OF PAGES 59
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Program
Discrete Convolution
Large Matrices
Method of Moments

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes an iterative technique to solve the matrix equation formulated by the Method of Moments. The advantage is that the number of multiplicative operations required is of the order NlogN, instead of the order N^2 to N^3 required by other solution techniques. Sample computations are made for the thin wire scatterers, linear arrays, and planar arrays. The number of iterations required for a given type of problem is found to be almost independent of the size of the matrix. The largest example given in

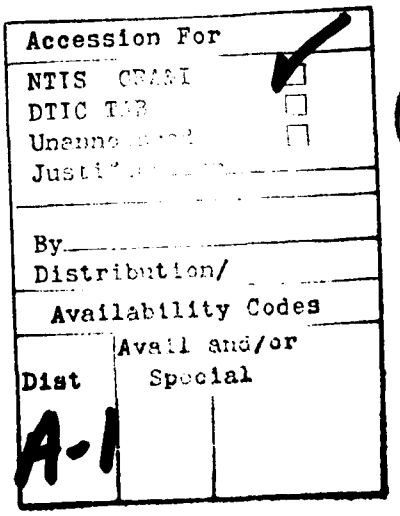
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (continued from previous page)

this report is for a planar array with 1849 antennas, which required four minutes of computing time on an IBM 4341.

CONTENTS



I. INTRODUCTION

The Discrete Convolution Method (DCM) is an iterative solution technique for solving the matrix equation formulated by using the Method of Moments (MOM) [1]. This is accomplished essentially by looking at the matrix equation, as a (set of) convolution equation(s). The DCM can solve a properly formulated NxN matrix equation, with only NlogN order multiplicative operations, instead of N^3 as in Gaussian Eiimination. The number of iterations needed for a given accuracy is also found to be practically independent of the size N.

II. FORMULATION

We first prove that a properly formulated MOM matrix equation can be viewed as a convolution process and then develop the solution technique. Depending on the type of problem, the equation can be reformulated into two types of convolution processes.

1) For some MOM problems, the matrix equation can be rewritten as

$$\sum_{n=1}^{N} Z_{mn} J_n = V_m \quad ,m=1,2,\ldots,N \qquad (1)$$

where V_m's are all known. For most of these problems it is

possible to choose expansion functions, renumber them and add dummy segments (if needed), so that (1) becomes

$$\sum_{n_1=1}^{N_1}\sum_{n_2=1}^{N_2}\cdots\sum_{n_M=1}^{N_M} Z_{p_1p_2\cdots p_M q_1q_2\cdots q_M} J_{q_1q_2\cdots q_M} = V_{p_1p_2\cdots p_M} \qquad (2)$$

where $p_1=1,2,\ldots,N_1$

$p_2=1,2,\ldots,N_2$

. .

. .

$p_M=1,2,\ldots,N_M$

Here $p_1p_2\cdots p_M$ are a renumbering of the original N segments plus the added dummy segments, a total of $N_1N_2\cdots N_M$. Furthermore

$$Z_{p_1p_2\cdots p_M q_1q_2\cdots q_M} = Z_{p_1-q_1,p_2-q_2,\ldots p_M-q_M} \qquad (3)$$

and $V_{p_1p_2\cdots p_M}$ are not all known. The values of V's corresponding to dummy elements are unknown but J's corresponding to these are zero. If we call the domain of original segments S and the domain of all segments S_e, then

$V_{p_1p_2\cdots p_M}$ are known

$J_{p_1p_2\cdots p_M}$ are unknown (to be solved for) (4)

if $p_1p_2\cdots p_M \in S$ and

$V_{p_1p_2\cdots p_M}$ are unknown

$J_{p_1p_2\cdots p_M}$ are known (=0)

if $p_1 p_2 \ldots p_M \in (S_e - S)$.

Combining (2) and (3), we get

$$\sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} \cdots \sum_{n_M=1}^{N_M} Z(p_1-q_1, p_2-q_2, \ldots p_M-q_M) J(q_1, q_2, \ldots q_M)$$

$$= V(p_1, p_2, \ldots, p_M) \qquad (5)$$

where $p_1=1,2,\ldots N_1; p_2=1,2,\ldots N_2; \ldots p_M=1,2,\ldots N_M$.

But (5) can be easily recognized as an M dimensional convolution equation [2]. Therefore (5) can be rewritten as

$$\vec{Z} * \vec{J} = \vec{V} \qquad p_1 p_2 \ldots p_M \in S_e \qquad (6)$$

where "*" denotes convolution of the appropriate (M here) order or dimension. To give a one dimensional example, consider scattering from two quarter wavelength straight wires that lie along a single axis 0.15 wavelength apart, as shown in Fig. 1(a).

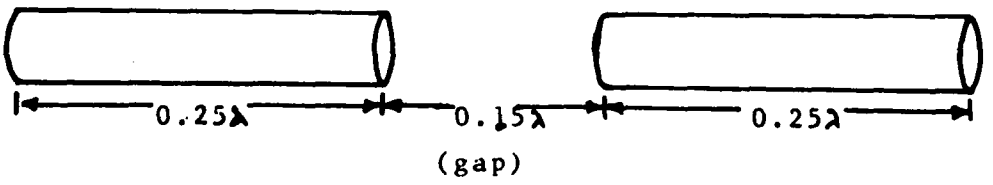


Fig. 1(a) Two quarter wavelength wires seperated by 0.15 wavelength gap

Fig. 1(b) Segmentation used for the problem of Fig. 1(a)

Further, suppose that we break the wire into 0.05 wavelength segments and also add three dummy segments in between as shown in Fig. 1(b). The matrix equation is then

$$\sum_{n=1}^{13} Z_{mn} J_n = V_m \qquad ,m=1,2,\ldots,13 \tag{7}$$

However, it is clear that the value of Z_{mn} depends only on $(m-n)$. Therefore (7) can be rewritten as

$$\sum_{n=1}^{13} Z(m-n)J(n) = V(m) \qquad ,m=1,2,\ldots,13 \tag{8}$$

In other words, the equation is a one dimensional convolution,

$$\vec{Z} * \vec{J} = \vec{V} \tag{9}$$

where

V_m are known

J_m are unknown for m=1,2,3,4,5,9,10,11,12,13

V_m are unknown

J_n are known for m=6,7,8 (10)

Other examples of the one dimensional convolution are helical wires, infinite strips, infinite circular cylindrical segments, linear antenna arrays, etc.

A two dimensional example is the MOM formulation of a linear antenna array problem using several expansion functions for each antenna, as shown in Fig. 2(b).

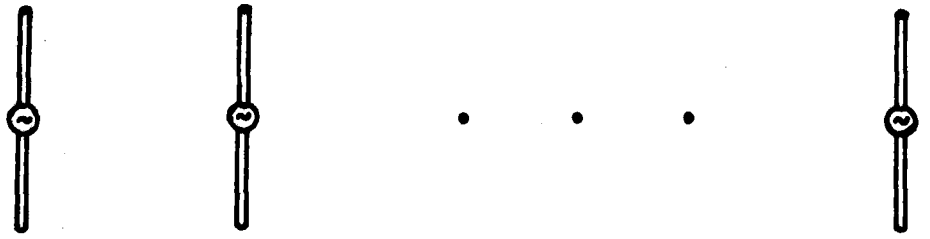


Fig. 2(a) A linear array of wire dipoles

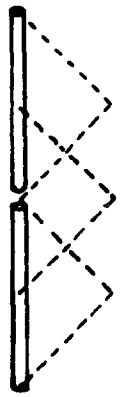


Fig. 2(b) Expansion functions used for the antenna current

If p_1 denotes antenna number and p_2 denotes expansion function number for each antenna, it is easy to see that

$$Z_{p_1 p_2 q_1 q_2} = Z(p_1-q_1, p_2-q_2) \qquad (11)$$

Note that if we use only one expansion function per segment we get a one dimensional convolution equation.

A three dimensional example is the solution of a rectangular antenna array problem by using several functions per antenna. The problem is as shown in Fig. 3.

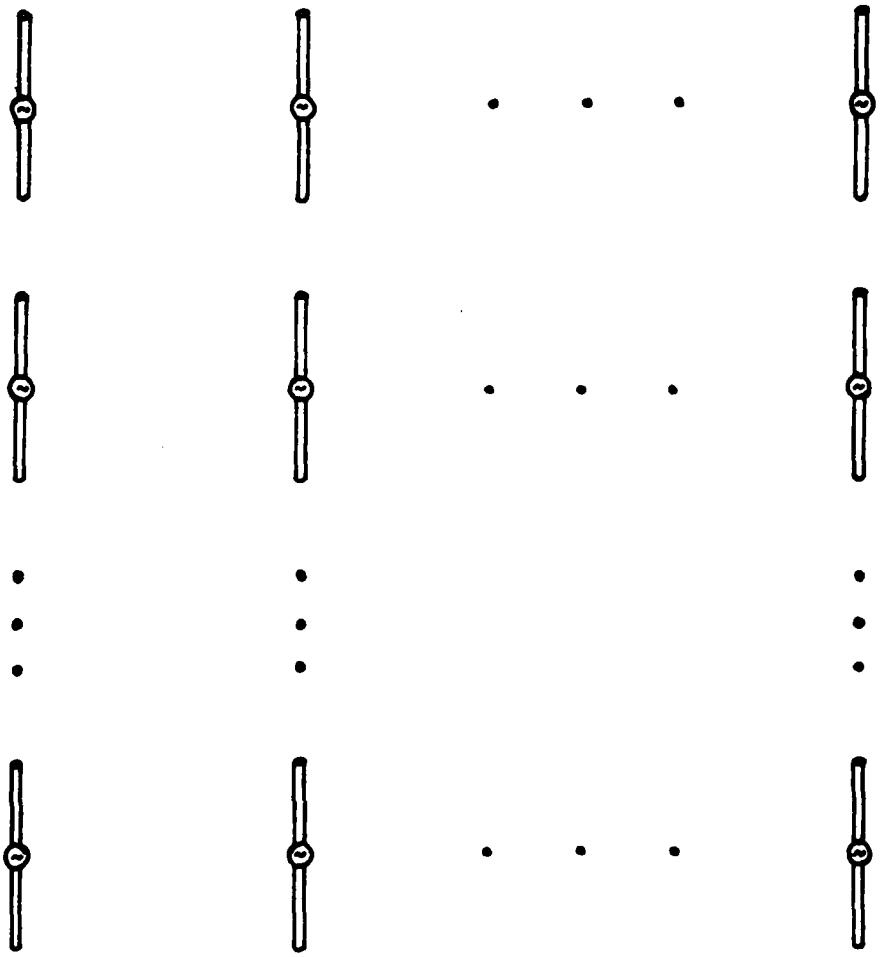


Fig. 3 A planar array of wire dipole antennas

Note that if we use a single expansion function per antenna we get a two dimensional convolution equation. A four dimensional equation would be produced by cubic array problems using multiple expansions, etc.

2) Most of the remaining MOM problems can be reformulated

into a set of convolution equations, though neither the reformulation nor the solution is as straight forward as in case (1). In fact, the reformulation will depend on the type of problem and the expansion function(s) chosen. Two examples are given here to demonstrate the general technique.

For the first example consider an arbitrarily shaped flat scatterer. Take N rectangular segments to approximate its shape as shown in Fig. 4.

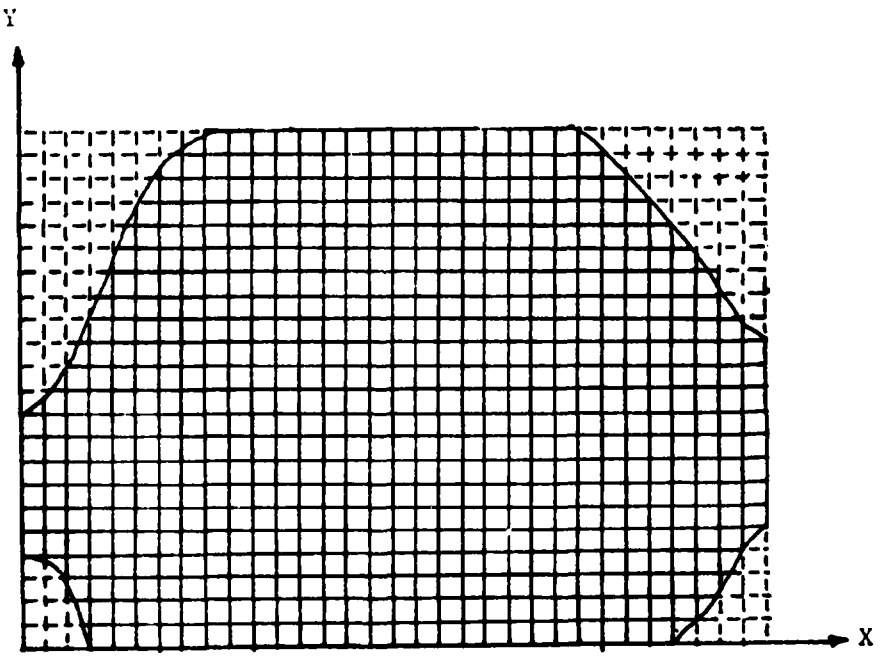


Fig. 4 A flat conducting scatterer approximated by rectangular subsections

Add (N_e-N) segments (dummy segments) to get a full rectangle. Now, on each segment (or group of segments) we

choose two independent current expansion functions; one in the x direction and the other in the y direction, as shown in Fig. 5.

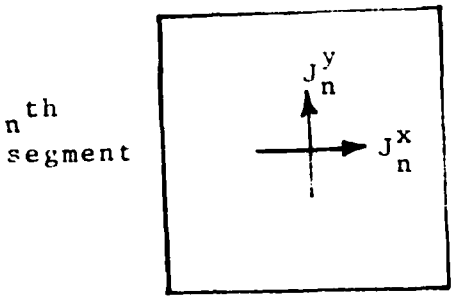


Fig. 5 Expansion functions used on the n^{th} subsection

It is apparent that we then have the following matrix equation.

$$\begin{bmatrix} [Z^{xx}] & [Z^{xy}] \\ [Z^{yx}] & [Z^{yy}] \end{bmatrix} \begin{bmatrix} I^x \\ I^y \end{bmatrix} = \begin{bmatrix} V^x \\ V^y \end{bmatrix} \tag{12}$$

Here, $[Z^{xx}]$, $[Z^{xy}]$, $[Z^{yx}]$, and $[Z^{yy}]$ are all block Toeplitz matrices of size N_e. Renumbering the segments in terms of rows and columns, (12) can be rewritten as

$$\sum_{n_2=1}^{N_2} \sum_{n_1=1}^{N_1} Z^{xx}_{m_1 m_2 n_1 n_2} I^x_{n_1 n_2} + \sum_{n_2=1}^{N_2} \sum_{n_1=1}^{N_1} Z^{xy}_{m_1 m_2 n_1 n_2} I^y_{n_1 n_2} = V^x_{m_1 m_2}$$

$$\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{yx}_{m_1m_2n_1n_2} I^{x}_{n_1n_2} + \sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{yy}_{m_1m_2n_1n_2} I^{y}_{n_1n_2} = V^{y}_{m_1m_2}$$

$$m_1=1,2,\ldots,N_1;\ m_2=1,2,\ldots,N_2 \qquad (13)$$

where $N_1N_2=N_e$, N_1 being the number of segments in the x direction and N_2 being the number of segments in the y direction. Also

$$Z^{xx}_{m_1m_2n_1n_2} = Z^{xx}(m_1-n_1, m_2-n_2)$$

$$Z^{yx}_{m_1m_2n_1n_2} = Z^{yx}(m_1-n_1, m_2-n_2) \qquad (14)$$

$$Z^{xy}_{m_1m_2n_1n_2} = Z^{xy}(m_1-n_1, m_2-n_2)$$

$$Z^{yy}_{m_1m_2n_1n_2} = Z^{yy}(m_1-n_1, m_2-n_2)$$

Therefore (13) can be rewritten as

$$\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{xx}(m_1-n_1, m_2-n_2) I^{x}(n_1,n_2) +$$

$$\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{xy}(m_1-n_1, m_2-n_2) I^{y}(n_1,n_2) = V^{x}(m_1,m_2)$$

$$\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{yx}(m_1-n_1,m_2-n_2)I^x(n_1,n_2)+$$

$$\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{yy}(m_1-n_1,m_2-n_2)I^y(n_1,n_2) = V^y(m_1,m_2)$$

$$m_1=1,2,\ldots,N_1;\ m_2=1,2,\ldots,N_2 \qquad (15)$$

Now (15) is obviously a convolution equation set, written symbolically as

$$Z^{xx} * I^x + Z^{xy} * I^y = V^x$$

$$Z^{yx} * I^x + Z^{yy} * I^y = V^y \qquad (16)$$

For the second example consider an arbitrarily shaped solid imperfect conductor or dielectric. Now take N rectangular cubic segments to approximate its shape, as shown in Fig. 6.

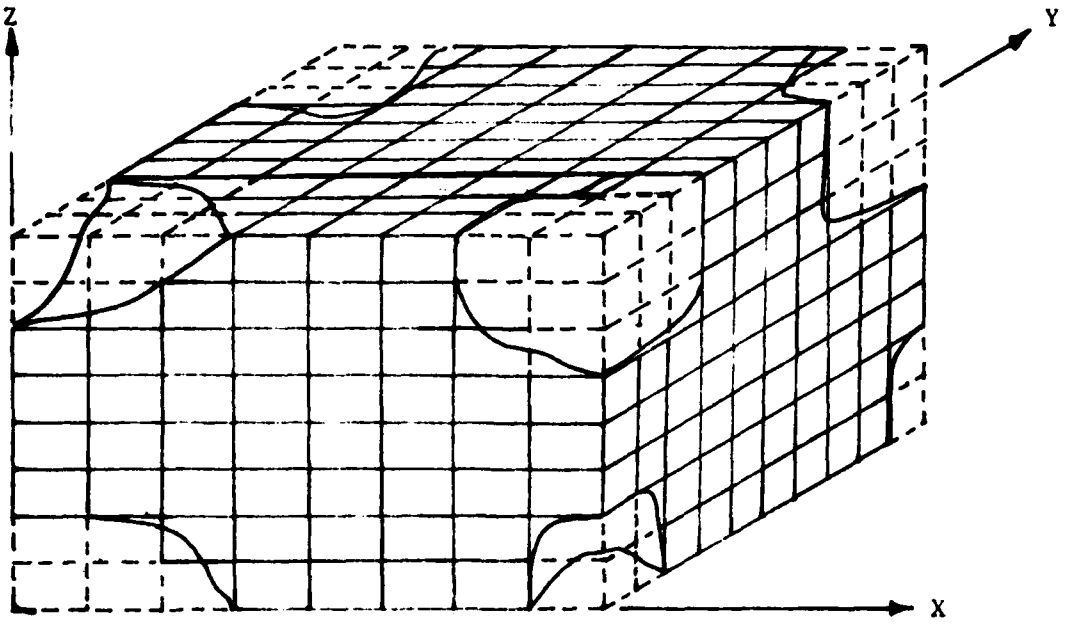


Fig. 6 A solid dielectric or imperfectly conducting scatterer approximated by cubic subsections

Add N_e-N dummy segments to get a full rectangular cube. To solve the problem of scattering from the imperfect conductor or dielectric using the MOM formulation, we choose for each segment (or group of segments) three independent current expansion functions; one in the x direction, another in the y direction and the third in the z direction as shown in Fig. 7.

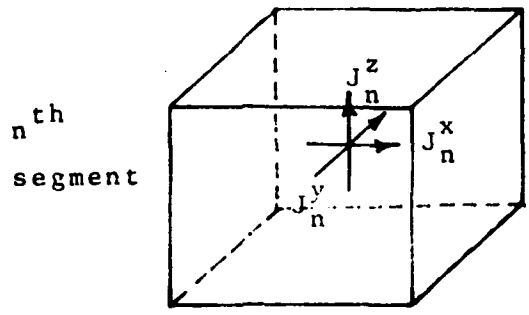


Fig. 7 Expansion functions used in the n^{th} subsection

It is apparent that we then have the following matrix equation.

$$\begin{bmatrix} [Z^{xx}] & [Z^{xy}] & [Z^{xz}] \\ [Z^{yx}] & [Z^{yy}] & [Z^{yz}] \\ [Z^{zx}] & [Z^{zy}] & [Z^{zz}] \end{bmatrix} \begin{bmatrix} I^x \\ I^y \\ I^z \end{bmatrix} = \begin{bmatrix} V^x \\ V^y \\ V^z \end{bmatrix} \qquad (17)$$

Here, $[Z^{xx}]$, $[Z^{xy}]$, etc. are all block block Toeplitz matrices of size N_e. Renumbering the segments in terms of rows and columns, we can write (17) as

$$\sum_{n_3=1}^{N_3} \sum_{n_2=1}^{N_2} \sum_{n_1=1}^{N_1} Z^{xx}_{m_1 m_2 m_3 n_1 n_2 n_3} I^x_{n_1 n_2 n_3} +$$

$$\sum_{n_3=1}^{N_3}\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{xy}_{m_1m_2m_3n_1n_2n_3}\, I^{y}_{n_1n_2n_3} +$$

$$\sum_{n_3=1}^{N_3}\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{xz}_{m_1m_2m_3n_1n_2n_3}\, I^{z}_{n_1n_2n_3} = V^{x}_{m_1m_2m_3}$$

$$\text{etc.} \qquad (18)$$

Here $N_1N_2N_3=N_e$, N_1 are the number of segments in the x direction, N_2 are the number of segments in the y direction, and N_3 are the number of segments in the z direction. Also,

$$Z^{xx}_{m_1m_2m_3n_1n_2n_3} = Z^{xx}(m_1-n_1,m_2-n_2,m_3-n_3)$$

$$Z^{yx}_{m_1m_2m_3n_1n_2n_3} = Z^{yx}(m_1-n_1,m_2-n_2,m_3-n_3) \qquad (19)$$

$$Z^{zx}_{m_1m_2m_3n_1n_2n_3} = Z^{zx}(m_1-n_1,m_2-n_2,m_3-n_3)$$

etc.

Therefore (18) can be rewritten as,

$$\sum_{n_3=1}^{N_3}\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{xx}(m_1-n_1,m_2-n_2,m_3-n_3)\, I^{x}(n_1,n_2,n_3) +$$

$$\sum_{n_3=1}^{N_3}\sum_{n_2=1}^{N_2}\sum_{n_1=1}^{N_1} Z^{yx}(m_1-n_1,m_2-n_2,m_3-n_3)\, I^{y}(n_1,n_2,n_3) +$$

$$\sum_{n_3=1}^{N_3} \sum_{n_2=1}^{N_2} \sum_{n_1=1}^{N_1} Z^{xz}(m_1-n_1, m_2-n_2, m_3-n_3)\, I^z(n_1, n_2, n_3)$$

$$= V^x(m_1, m_2, m_3) \qquad (20)$$

etc.

Now (20) is obiously a set of convolution equations. Symbolically, we can write this set as

$$\vec{Z}^{xx} * \vec{I}^x + \vec{Z}^{xy} * \vec{I}^y + \vec{Z}^{xz} * \vec{I}^z = \vec{V}^x$$

$$\vec{Z}^{yx} * \vec{I}^x + \vec{Z}^{yy} * \vec{I}^y + \vec{Z}^{yz} * \vec{I}^z = \vec{V}^y \qquad (21)$$

$$\vec{Z}^{zx} * \vec{I}^x + \vec{Z}^{zy} * \vec{I}^y + \vec{Z}^{zz} * \vec{I}^z = \vec{V}^z$$

Here "*" denotes three dimensional convolution.

Other examples of "two and three dimensional" problems of this type (giving sets of two and three dimensional convolution equations) are apertures in an infinite flat conductor, antenna arrays with both polarizations, non planar antenna arrays with more than one polarization, etc. Also, some problems of type (1) can be reformulated as type (2) problems. For example, a planar array problem using more than one expansion per antenna (say two) could be written in the form of (13) and hence (16), by replacing x and y in the equations by 1 and 2 and by numbering the two expansions 1 and 2.

III. SOLUTION METHODS

The methods that we use to solve (6) for the problems of the first type, and (16),(21) etc. for the problems of the second type are iterative. We will discuss here in detail, the method for solving (6). To demonstrate the general approach for the second type of problem, we also discuss the method for solving (16).

In (6), we are taking the convolution product of the left hand side and equating it to the right hand side for the values of $p_1 p_2 \ldots p_M$ in the region S_e. However, the full convolution process given by the left hand side of (6) produces results not only for S_e, but also for the regions outside of S_e. Specifically, convolution results are produced for

$$\begin{aligned} p_1 &= -N_1+2, -N_1+3, \ldots, -1, 0, 1, 2, \ldots, 2N_1-1 \\ p_2 &= -N_2+2, -N_2+3, \ldots, -1, 0, 1, 2, \ldots, 2N_2-1 \\ p_3 &= -N_3+2, -N_3+3, \ldots, -1, 0, 1, 2, \ldots, 2N_3-1 \\ &\text{etc.} \end{aligned} \tag{22}$$

However, values of $\vec{V}$ not in the region S are unknown and values of $\vec{J}$ not in the region S are known (equal to zero). If we call the values of $\vec{V}$ in region S to be $\vec{V}^i$ (for impressed) and outside S to be $\vec{V}^o$, then (6) can be rewritten as

$$\vec{Z} * \vec{J} = \vec{V}^i + \vec{V}^o = \vec{V} \tag{23}$$

Here, no restrictions are placed on the region of validity.

If we take the Discrete Fourier Transform (DFT) of (23), on the basis of $3N_1-2$ elements for p_1, $3N_2-2$ elements for p_2 etc., then we get an algebraic equation [2].

$$\tilde{Z}\,\tilde{J} = \tilde{V} \tag{24}$$

Here "$\sim$" denotes transformed quantities. Equation (24) is true for each transformed quantity; in other words,

$$\tilde{Z}(k_1,k_2,\ldots,k_M)\tilde{J}(k_1,k_2,\ldots,k_M) = \tilde{V}(k_1,k_2,\ldots,k_M)$$
$$k_1=1,2,\ldots,3N_1-2;\ k_2=1,2,\ldots,3N_2-2;\ldots;\ \text{etc.} \tag{25}$$

Therefore, if we know $\vec{V}$ for all values of $p_1p_2\ldots p_M$, then we can determine $\tilde{V}$ and find $\tilde{J}$ by

$$\tilde{J}(k_1,k_2,\ldots,k_M) = \frac{\tilde{V}(k_1,k_2,\ldots,k_M)}{\tilde{Z}(k_1,k_2,\ldots,k_M)} \tag{26}$$

The inverse DFT (IDFT) then gives $\vec{J}$. However, we know only $\vec{V}^i$ and not $\vec{V}^o$. Thus the following procedure is used:

STEP 1-Assume $\vec{V}^o$. Normally we take all (initial) values of $\vec{V}^o$ to be zero. (As shown later in the Appendix, the "distance" of the initial guess from the correct value does not effect the convergence; only the number of iterations needed.) Call this first guess of $\vec{V}^o$ by $\vec{V}^o_{(1)}$.

STEP 2-Take the DFT of $\vec{Z}$ on the basis of $3N_1-2$ for p_1, $3N_2-2$ for p_2, etc. to get $\tilde{Z}$.

STEP 3-Compute $\vec{V}_{(1)} = \vec{V}^i + \vec{V}^o_{(1)}$ (27)

STEP 4-Take the DFT of $\vec{V}_{(1)}$ on the same basis as in step 2 to get $\tilde{V}_{(1)}(k_1,k_2,\ldots,k_M)$

STEP 5-Compute $\tilde{J}_{(1)}(k_1,k_2,\ldots,k_M)$ using (26).

STEP 6-Take the IDFT (on the same basis as DFT in step 2) of $\tilde{J}_{(1)}$ to get $\vec{J}_{(1)}$.

STEP 7-Since $\vec{J}_{(1)}$ is not the correct answer, it will have nonzero values outside S. Change the values of $\vec{J}_{(1)}$ outside S to zero. (This is the same as truncating or projecting $\vec{J}_{(1)}$ onto S.) Call this $\vec{J}^{p}_{(1)}$.

STEP 8-Take the DFT of $\vec{J}^{p}_{(1)}$ to get $\tilde{J}^{p}_{(1)}(k_1,k_2,\ldots,k_M)$

STEP 9-Compute $\tilde{V}^{p}_{(2)}$ ($=\tilde{Z}\ \tilde{J}^{p}$, as given in equation (25)).

STEP 10-Take IDFT of $\tilde{V}^{p}_{(2)}$ to get $\vec{V}^{p}_{(2)}$. (Note $\vec{V}^{p}=\vec{Z}*\vec{J}^{p}_{(2)}$)

STEP 11-Since $\vec{J}^{p}_{(1)}$ is not yet the correct answer, values of $\vec{V}^{p}_{(2)}$ on S are not be equal to $\vec{V}^{i}$. Here, we can check the accuracy by comparing $\vec{V}^{i}$ with $\vec{V}^{p}_{(2)}$ on S. One method is to check the maximum $(\vec{V}^{i}-\vec{V}^{p}_{(2)})/\vec{V}^{i}$ for all elements, as well as the average. If the maximum and average are below a certain value (say .1% and .01% respectively), then stop. We can also use the criterion of the convergence of $\vec{J}_{(n)}$, i.e., the relative magnitude of $\vec{J}_{(n)}-\vec{J}_{(n-1)}$ in comparison to $\vec{J}_{(n)}$, or combine the two criterions.

STEP 12-If we decide that more iterations are needed, then change the values of $\vec{V}^{p}_{(2)}$ on S to $\vec{V}^{i}$. Call this $\vec{V}_{(2)}$.

STEP 13-Replace $\vec{V}_{(1)}$ in step 4 by $\vec{V}_{(2)}$.

STEP 14-Continue from step 4 onwards until the criterions in step 11 are satisfied.

The solution techniques for sets of convolution equations produced by the problems of the second type are not as straight forward. To illustrate the general technique, consider (16). In (16), we are taking the convolution products of the left hand sides and equating them to the right hand sides for values of $m_1 m_2$ in the region S_e. However, the full convolution process, as given by the left hand sides of (16) produces results not only for S_e, but also for regions outside S_e. Specifically, convolution results are produced for

$$\begin{aligned} m_1 &= -N_1+2, -N_1+3, \ldots, -1, 0, 1, 2, \ldots, 2N_1-1 \\ m_2 &= -N_2+2, -N_2+3, \ldots, -1, 0, 1, 2, \ldots, 2N_2-1 \end{aligned} \tag{28}$$

as in the first type of problems. However, values of $\vec{V}^x$ and $\vec{V}^y$ not in the region S are unknown, and the values of $\vec{I}^x$ and $\vec{I}^y$ not in the region S are known to be equal to zero. If we denote the values of $\vec{V}^x$ and $\vec{V}^y$ in the region S by $\vec{V}^{xi}$ and $\vec{V}^{yi}$ (for impressed), and those outside S by $\vec{V}^{xo}$ and $\vec{V}^{yo}$, then (16) can be rewritten as,

$$\begin{aligned} \vec{Z}^{xx} * \vec{I}^x + \vec{Z}^{xy} * \vec{I}^y &= \vec{V}^{xi} + \vec{V}^{xo} = \vec{V}^x \\ \vec{Z}^{yx} * \vec{I}^x + \vec{Z}^{yy} * \vec{I}^y &= \vec{V}^{yi} + \vec{V}^{yo} = \vec{V}^y \end{aligned} \tag{29}$$

Here, no restrictions are placed on the region of validity.

If we take the DFT of (29) on the basis of $3N_1-2$ elements for m_1 and $3N_2-2$ elements for m_2, we then get the algebraic equations [2]

$$\tilde{Z}^{xx}(k_1,k_2)\tilde{I}^{x}(k_1,k_2)+\tilde{Z}^{xy}(k_1,k_2)\tilde{I}^{y}(k_1,k_2)=\tilde{V}^{x}(k_1,k_2)$$

$$\tilde{Z}^{yx}(k_1,k_2)\tilde{I}^{x}(k_1,k_2)+\tilde{Z}^{yy}(k_1,k_2)\tilde{I}^{y}(k_1,k_2)=\tilde{V}^{y}(k_1,k_2)$$

$$k_1=1,2,\ldots,3N_1-2;\ k_2=1,2,\ldots,3N_2-2 \qquad (30)$$

Equation (30) is true for each transformed quantity. Therefore, if we know $\tilde{V}^{x}$ and $\tilde{V}^{y}$ for all values of m_1 and m_2, we can find $\tilde{V}^{x}$ and $\tilde{V}^{y}$. Since (30) can be written as,

$$\begin{bmatrix} \tilde{Z}^{xx}(k_1,k_2) & \tilde{Z}^{xy}(k_1,k_2) \\ \tilde{Z}^{yx}(k_1,k_2) & \tilde{Z}^{yy}(k_1,k_2) \end{bmatrix} \begin{bmatrix} \tilde{I}^{x}(k_1,k_2) \\ \tilde{I}^{y}(k_1,k_2) \end{bmatrix} = \begin{bmatrix} \tilde{V}^{x}(k_1,k_2) \\ \tilde{V}^{y}(k_1,k_2) \end{bmatrix} \qquad (31)$$

we can find $\tilde{I}^{x}$ and $\tilde{I}^{y}$ easily as

$$\tilde{I}^{x}(k_1,k_2) = \frac{\tilde{Z}^{yy}(k_1,k_2)\tilde{V}^{x}(k_1,k_2)-\tilde{Z}^{xy}(k_1,k_2)\tilde{V}^{y}(k_1,k_2)}{\tilde{Z}^{yy}(k_1,k_2)\tilde{Z}^{xx}(k_1,k_2)-\tilde{Z}^{xy}(k_1,k_2)\tilde{Z}^{yx}(k_1,k_2)} \qquad (32)$$

$$\tilde{I}^{y}(k_1,k_2) = \frac{\tilde{V}^{x}(k_1,k_2) - \tilde{Z}^{xx}(k_1,k_2)\tilde{I}^{x}(k_1,k_2)}{\tilde{Z}^{xy}(k_1,k_2)}$$

The IDFT then gives $\hat{I}^{x}$ and $\hat{I}^{y}$. However, we know only $\hat{V}^{xi}$, $\hat{V}^{yi}$ and not $\hat{V}^{xo}$, $\hat{V}^{yo}$. Thus the following iterative procedure can be used.

STEP 1-Assume $\hat{V}^{xo}$, $\hat{V}^{yo}$. Normally, we take all elements

of $\vec{V}^{xo}$, $\vec{V}^{yo}$ to be zero. Denote this first guess of $\vec{V}^{xo}$, $\vec{V}^{yo}$ by $\vec{V}^{xo}_{(1)}$, $\vec{V}^{yo}_{(1)}$.

STEP 2-Take the DFT of $\vec{Z}^{xx}$, $\vec{Z}^{xy}$, $\vec{Z}^{yx}$, and $\vec{Z}^{yy}$ on the basis of $3N_1-2$ for m_1 and $3N_2-2$ for m_2 to get $\tilde{Z}^{xx}$, $\tilde{Z}^{xy}$, $\tilde{Z}^{yx}$, and $\tilde{Z}^{yy}$.

STEP 3-Compute $\vec{V}^{x}_{(1)} = \vec{V}^{xi} + \vec{V}^{xo}_{(1)}$

$$\vec{V}^{y}_{(1)} = \vec{V}^{yi} + \vec{V}^{yo}_{(1)} \tag{33}$$

STEP 4-Take the DFT of $\vec{V}^{x}_{(1)}$, $\vec{V}^{y}_{(1)}$ on the same basis as in step 2 to get $\tilde{V}^{x}_{(1)}(k_1,k_2)$ and $\tilde{V}^{y}_{(1)}(k_1,k_2)$.

STEP 5-Compute $\tilde{I}^{x}_{(1)}(k_1,k_2)$, $\tilde{I}^{y}_{(1)}(k_1,k_2)$ using (32).

STEP 6-Take the IDFT (on the same basis as DFT in step 2) of $\tilde{I}^{x}_{(1)}$ and $\tilde{I}^{y}_{(1)}$.

STEP 7-Since $\vec{I}^{x}_{(1)}$ and $\vec{I}^{y}_{(1)}$, are not the correct answer, they have nonzero values outside S. Therefore, change the values of $\vec{I}^{x}_{(1)}$ and $\vec{I}^{y}_{(1)}$ outside S to zero. (This is the same as projecting $\vec{I}^{x}_{(1)}$ and $\vec{I}^{y}_{(1)}$ onto S.) Call these $\vec{I}^{xp}_{(1)}$ and $\vec{I}^{yp}_{(1)}$.

STEP 8-Take DFT's of $\vec{I}^{xp}_{(1)}$, $\vec{I}^{yp}_{(1)}$ to get $\tilde{I}^{xp}_{(1)}(k_1,k_2)$ and $\tilde{I}^{yp}_{(1)}(k_1,k_2)$.

STEP 9-Compute $\tilde{V}^{xp}_{(2)}$ and $\tilde{V}^{yp}_{(2)}$ as given in (30).

STEP 10-Take IDFTs of $\tilde{V}^{xp}_{(2)}$ and $\tilde{V}^{yp}_{(2)}$ to get $\vec{V}^{xp}_{(2)}$ and $\vec{V}^{yp}_{(2)}$. (Note that

$$\vec{V}^{xp}_{(2)} = \vec{Z}^{xx} * \vec{I}^{xp}_{(2)} + \vec{Z}^{xy} * \vec{I}^{yp}_{(2)} \text{ and}$$

$$\vec{V}^{yp}_{(2)} = \vec{Z}^{yx} * \vec{I}^{xp}_{(2)} + \vec{Z}^{yy} * \vec{I}^{yp}_{(2)})$$

STEP 11-Since $\vec{I}^{xp}_{(1)}$, $\vec{I}^{yp}_{(1)}$ are not yet the correct answers, values of $\vec{V}^{xp}_{(2)}$ and $\vec{V}^{yp}_{(2)}$ on S are not equal to $\vec{V}^{xi}$ and $\vec{V}^{yi}$. Here we can check the accuracy by comparing $\vec{V}^{xi}$, $\vec{V}^{yi}$ with $\vec{V}^{xp}_{(2)}$, $\vec{V}^{yp}_{(2)}$ on S. The same kind(s) of criterion(s) as in step 11 of the solution procedure for the problems of the first type can be used to determine whether or not the iteration has converged.

STEP 12-If we decide that more iterations are needed, then change the values of $\vec{V}^{xp}_{(2)}$ and $\vec{V}^{yp}_{(2)}$ on S to be $\vec{V}^{xi}$ and $\vec{V}^{yi}$. Denote these $\vec{V}^{x}_{(2)}$, $\vec{V}^{y}_{(2)}$.

STEP 13-Replace $\vec{V}^{x}_{(1)}$ and $\vec{V}^{y}_{(1)}$ in step 4 by $\vec{V}^{x}_{(2)}$ and $\vec{V}^{y}_{(2)}$.

STEP 14-Continue from step 4 onwards until the criterion(s) of convergence in step 11 are satisfied.

IV. SAMPLE COMPUTATIONS AND COMMENTS

Computer Programs using the techniques devised in the preceding sections have been written. They are listed in the Appendix of this report. In this section we give the results of computations that were made using these programs. The routines and formulations for computing the impedence matrix (or mutual coupling matrix) [Z] are from [3],[4], and [5].

Two types of one dimensional problems and one type of two dimensional problems are solved. They are,

(i) scattering from straight thin wires

(ii) linear antenna arrays

(iii) planar antenna arrays

Fig. 8 shows the problem of scattering from a straight thin wire illuminated by a perpendicular plane wave. The Method of Moments formulation is made by breaking the wire into N_s segments of equal length and using triangular expansion functions, as given in [3]. Table 1 gives the number of iterations needed to get convergence using DCM for single thin wire problems. Comparison with LU decomposition method in terms of the number of multiplicative operations required is also given.

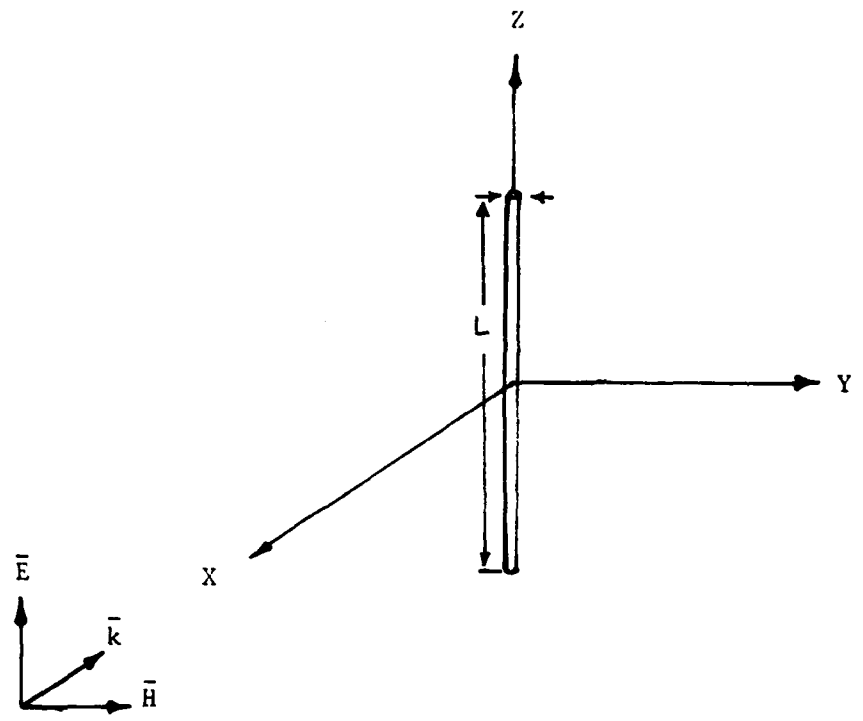


Fig. 8. Scattering from a thin straight wire.

Table 1. Comparison of DCM and LU decomposition methods

L	l	N_s	N_e	N_c	LU decomp (mult. ops.)	I	DCM (mult. ops.)
0.75	0.0625	12	5	15	42	10	1280
2.0	0.05	40	19	64	2286	9	6912
4.0	0.1	40	19	64	2286	6	4608
8.0	0.1	80	39	128	19773	11	19712
16.0	0.1	160	79	256	164346	5	20480
23.85	0.075	318	158	512	1314771	12	135168
32.0	0.1	320	159	512	1339893	6	67584

Here a = wire radius in wavelengths(0.013477089 for problems in Table 1)

L = length of wire in wavelengths

l = length of each segment in wavelengths

N_s = number of segments

N_e = number of expansion functions needed

N_c = basis on which FFT and IFFT are taken for DCM

I = number of iterations needed to get maximum error in current to be less than 1% or maximum error in field to be less than .1%

From Table 1 we see that, based on the number of complex multiplications required, DCM is faster than LU decomposition for problems with more than 40 expansions. Total computing time needed to solve the 32 wavelengths problem (last entry in Table 1) is 14.16 seconds on an IBM 4341. This includes computing time needed to set up the impedance matrix.

The problem of two thin wire scatters with a gap in between, as in Figure 1, was also solved. The number of iterations needed for .123% maximum field error and .00404% average field error was found to be 14. The problem is the same as the 23.85 wavelengths wire problem of Table 1, except that 114 segments in the middle are missing. Since the original problem needed 12 iterations for the same level of accuracy, the insertion of the gap does not seem to cause much increase in computing time.

Fig. 2 shows the problem of radiation from a linear antenna array. If one expansion function per antenna is used, then the Method of Moments formulation gives a matrix equation equivalent to a one dimensional convolution equation. The formulation used is as given in [4]. Since the mutual coupling matrix is Toeplitz in this case, it can be solved using the faster (N^2 order) algorithm for Toeplitz matrices as given in [4]. Therefore, Table 2 compares between computing time needed for DCM with the computing time needed for the (N^2 order) Toeplitz algorithm given in [4]. The computing time measurements were made on an IBM 4341. The matrix set up time is not included, which would be the same for both cases. All the problems in Table 2 are linear arrays with halfwave antennas one-quarter wavelength in front of an infinite ground plane. The seperation between antennas is one-half wavelength also. Uniform excitation is used for all cases given in Table 2. Other excitations were tried and number of iterations needed (and hence computing times) were found to be practically independent of the type of excitation.

We see that, for very large arrays, DCM is faster. For 1000 antenna elements it is nearly 4 times faster. Breakeven seems to occur at about 300 antenna elements. Comparisons between the solutions given by DCM and Toeplitz algorithms were made for all problems except the 1000 antennas problem, and the agreement was to within .1% maximum difference in current in all cases.

Table 2. Comparison of DCM and Toeplitz methods

N_e	I	Setup Time (sec)	DCM Time (sec)	Toeplitz Time (sec)	Field Error (%)
11	4		.57	.37	
44	3		1.55	1.23	
88	3	1.51	2.43	1.57	.0222
					.000971
250	3	3.97	9.8	9.79	.0224
					.000349
352	3	5.55	19.48	18.53	.022
					.000935
1000	3	11.57	34.22	135.44	.0218
					.000423
1000	4	-	45.63	-	.00466
					.000365

Here N_e is the number of antennas in the array

I is the number of iterations needed for the given field error and the last current change. For both field error and last current change, the upper entry is maximum error and the lower entry is average error.

Computing time savings are even more dramatic for antenna arrays with some antennas missing, i.e., gaps. The matrix produced by MOM in this case is no longer Toeplitz, and LU decomposition (needing $1/3\ N^3$ multiplicative operations) is usually used instead of Toeplitz methods (needing $2N^2$ multiplicative operations). But, as explained in section II, we can add dummy segments and still use DCM to solve the problem. The number of iterations needed increased by only 1 over that needed for the same problem without gaps in each of the cases that were tried. The results are given in Table 3.

Table 3. Results for array problems with gaps

N_e	N_g	Gap(s)	I	Field error(%)
44	1	17-28	4	.0166
				.00175(avg)
-	2	12-16	4	.0384
		28-32		.00759(avg)
-	3	12-15	4	.0330(max)
		21-26		.00542
		33-37		

Here N_g is the number of gaps

Gap(s) gives the start and end segment numbers of the gaps

I is the number of iterations needed for the given field error

With the LU decomposition method, the computing time would be 10 times larger even for the 44 antenna problem.

For linear antenna arrays which are lined up at an angle as shown in Fig. 9, then one expansion per antenna would no longer be enough since the current will not be symmetric anymore.

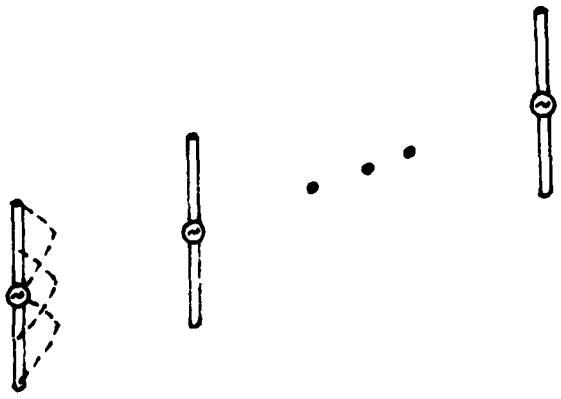


Fig. 9 Linear antenna array lined up at an angle

The MOM formulation will no longer give an impedance matrix that is Toeplitz but will give an impedance matrix that is block Toeplitz. This is equivalent to the two dimensional discrete convolution. We can solve by using the two dimensional DCM technique but since one of the dimensions will have only three points, it is not worthwhile. However, we can look at the matrix equation as three one dimensional convolution equations similar to what is done in (17) to (21). However, convolutions would be one dimensional here, instead of three dimensional as in (21). If we call the

first, second and the third expansions of each antenna, a,b, and c respectively, then the equivalent set of equations is

$$\begin{aligned} \vec{Z}^{aa} * \vec{I}^{a} + \vec{Z}^{ab} * \vec{I}^{b} + \vec{Z}^{ac} * \vec{I}^{c} &= \vec{0} \\ \vec{Z}^{ba} * \vec{I}^{a} + \vec{Z}^{bb} * \vec{I}^{b} + \vec{Z}^{bc} * \vec{I}^{c} &= \vec{V} \\ \vec{Z}^{ca} * \vec{I}^{a} + \vec{Z}^{cb} * \vec{I}^{b} + \vec{Z}^{cc} * \vec{I}^{c} &= \vec{0} \end{aligned} \qquad (34)$$

The solution of (34) using DCM would require nine times more multiplicative operations than the solution of (9). However, since the block Toeplitz solution technique will also need nine times more than the Toeplitz solution technique, the timing comparisons of Table 2 will remain unchanged. For problems with gaps, if three expansions per antenna are used, DCM will still need only nine times more multiplicative operations per iteration. But LU decomposition, being $1/3\ N^3$, will need twenty seven times more multiplicative operations.

Fig. 3 shows the problem of radiation from a planar antenna array. If only one expansion function per antenna is used, then the MOM formulation gives a matrix equation which is block Toeplitz. This is equivalent to a two dimensional convolution equation. The formulation is as given in [5]. Since the matrix equation is block Toeplitz, it can be solved using faster (N^a order where $a \approx 2.5$) algorithm for block Toeplitz matrices as given in [5]. Table 4 lists the computing time requirements for the DCM solution of some planar array problems. The computing times

given include the setup times. All the problems are for planar array problems with halfwave antennas one-quarter wavelength in front of an infinite ground plane. The seperation between antennas is one-half wavelength in either direction.

Table 4. Results for some planar array problems

N_e	Excitation	I	Computing Time(sec)	Field Error (%)	Current Change (%)
16	Uniform	4	2	.05	.25
(4x4)				.03	.14
36	Uniform	4	3	.056	.25
(6x6)				.025	.1
	Exponential	4	3	.055	.27
	Taper			.021	.087
	Beam Steer	4	3	.034	.62
	(45°-45°)			.008	.056
	Beam Steer	4	3	.02	.18
	(30°-20°)			.007	.05
	Progressive	4	3	.074	.3
	Phase shift			.032	.1
	(30°-20°)				
121	Uniform	4	13	.07	.29
				.03	.08
	Exponential	4	5	.068	.24
				.016	.05

N_e	Excitation	I	Computing Time(sec)	Field Error (%)	Current Change(%)
1849	Uniform	4	255	.07	.28
				.009	.023
	Beam Steer (30°-30°)	4	255	.028	.27
				.0014	.01

Here N_e is the number of antennas in the array

I is the number of iterations needed to get the given accuracy. For both field error and (last) current change, the upper entry is the maximum and the lower entry is the average.

We see that even for very large arrays DCM is quite fast. The problem with 1849 antennas takes only 4 minutes computing time. Just as for the linear arrays, problems with gaps could also be treated and would take comparable amounts of time.

For more accurate planar array solutions, three expansions per array should be used since the current on each antenna is not symmetric. This would increase the required computing time by a factor of nine as before. Notice also, the independence of the number of iterations required to the size of the array.

To check the accuracy of the DCM technique, the solution of the 36 antenna planar array was made using matrix inversion routine LINEQ from [3], and the agreement between the currents were found to be better than .1% maximum difference.

The problem of a planar array with antennas arranged in diamond patterns instead of rectangular, can also be formulated as a two dimensional convolution equation by adding dummy elements (as shown in Figure 11), to make a parallelogram. The diamond pattern arrangement is shown in Fig. 10.

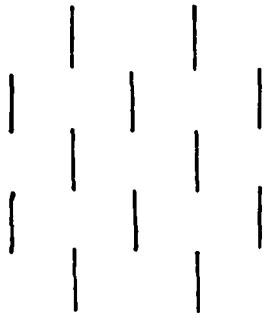


Fig. 10 The Diamond pattern arrangement

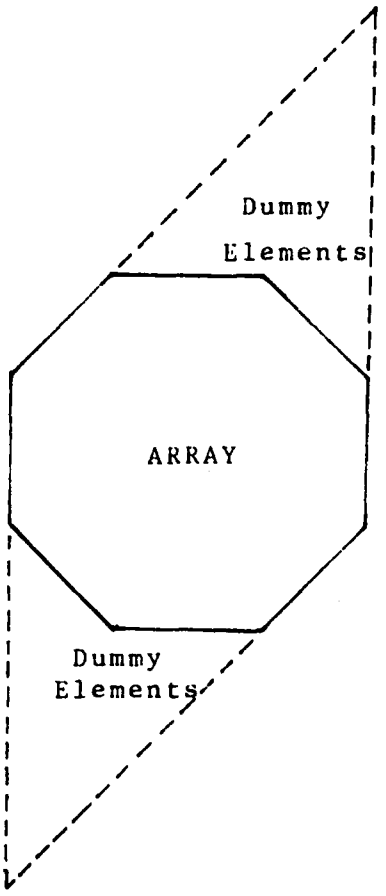


Fig. 11 A planar array with antennas arranged in diagonal patterns

The MOM formulation using one expansion per antenna will then give a block Toeplitz matrix which can be solved using two dimensional DCM. However, it cannot be solved using the block Toeplitz method since the field on the dummy elements are unknown. Therefore, only LU decomposition or two dimensional DCM can be used. For a large array, DCM will be considerably faster. Since the current on each antenna is not symmetric, using three expansion functions per antenna will give a much more accurate result and will need nine times more computing time for the DCM. With LU decomposition method computing time will go up twenty seven

times the already large value.

V. DISCUSSIONS

Discrete Convolution Method for solving the matrix equation set up by the MOM, is found to be accurate and much faster than either the Gaussian Elimination (LU decomposition) or the Toeplitz and block Toeplitz methods given in [4] and [5]. Also DCM can solve a wider range of problems than the Toeplitz and the block Toeplitz methods. The number of iterations needed by DCM for a given accuracy is also found to be practically independent of size. However, it is dependent on other factors. For instance, the number of iterations needed to solve an array problem is found to be dependent on antenna length, antenna seperation and the ground plane distance.

With careful formulation, the problem of radiation from a planar array backed by a finite ground plane can be solved with the DCM. Therefore DCM may prove to be useful in designing array antennas.

The other numerical technique using FFT and IFFT to solve iteratively, electrically large problems is the Spectral Theory of Diffraction (STD). This technique is well known and a number of papers and reports [6], [7],[8] etc., have been published about STD. The difference between

STD and DCM is that STD solves the problems in the spectral domain and DCM solves the problems in the spatial domain. Therefore, for certain types of problems STD may feel more natural and for other types of problems (for example planar arrays), DCM may feel more natural. Also errors in each technique have different causes.

Since both STD and DCM are iterative techniques using FFT and IFFT, they both have numerical inaccuracies associated with

(i) the use of FFT and IFFT

(ii) taking only a finite number of iterations based on some criterion

However, as shown in the Appendix, for DCM these errors are insignificant. But since DCM is the iterative solution of the MOM formulation of the original problem, DCM will have in addition the inaccuracies associated with the MOM formulation (but not the matrix inversion). On the other hand, STD has the following numerical problems [6],

(i) windowing and

(ii) the need to take sufficient number of points to make sure that the aliasing effect is small

Thus the numerical errors of DCM and STD are of different natures. However, since the MOM formulation has been in wide use for a considerable period of time, the numerical errors associated with the MOM are familiar through experience.

APPENDIX

I. INDEPENDENCE OF CONVERGENCE ON STARTING POINT

We prove here, the independence of convergence on starting point for the one dimensional case. Proofs for higher dimensions will be similar. Consider the one dimensional problem given in (9). The discrete convolution equation to be solved is,

$$\vec{Z} * \vec{J} = \vec{V} \qquad \text{(A1)}$$

But $\vec{V}$ is unknown, only $\Theta[\vec{V}]$ is known. Here the function $\Theta[\ ]$ means truncate values outside region S and replace with zeros. We also know that $\vec{J}$ is confined to S, i.e.

$$\vec{J} = \Theta[\vec{J}] \qquad \text{(A2)}$$

The general solution technique is,

$$\vec{Z} * \vec{J}_{n+1} = \Theta[\vec{V}] + \hat{\Theta}[\vec{Z} * \Theta[\vec{J}_n]] \qquad \text{(A3)}$$

where $\hat{\Theta}[\ ]$ is the complement of $\Theta[\ ]$,

$\vec{J}_n$ is the approximation of $\vec{J}$ at the n^{th} iteration

The correct solution is,

$$\vec{Z} * \vec{J} = \Theta[\vec{V}] + \hat{\Theta}[\vec{V}] \qquad \text{(A4)}$$

Therefore from (A3) and (A4), we get

$$\vec{Z} * (\vec{J}-\vec{J}_{n+1}) = \hat{\Theta}[\vec{V}-\vec{Z} * \Theta[\vec{J}_n]] \qquad \text{(A5)}$$

Using (A1), the equation above becomes,

$$\vec{Z} * (\vec{J}-\vec{J}_{n+1}) = \hat{\Theta}[\vec{Z}*\vec{J} - \vec{Z}*\Theta[\vec{J}]] \qquad \text{(A6)}$$

But by (A2),

$$\vec{Z} * (\vec{J}-\vec{J}_{n+1}) = \hat{\theta}[\vec{Z}* \theta[\vec{J}-\vec{J}_n]] \qquad (A7)$$

Therefore, if we denote the error in the approximate solution at n^{th} step, $(\vec{J}-\vec{J}_n)$ as $\vec{\mathcal{E}}_n$,

$$\vec{Z} * \vec{\mathcal{E}}_{n+1} = \hat{\theta}[\vec{Z}* \theta[\vec{\mathcal{E}}_n]] \qquad (A8)$$

We now prove the independence of convergence to the starting point, in the sense that if the convergence is achieved for a certain starting point, then the convergence is achieved for other starting points nearer or further than that. Suppose that we achieve convergence if we start with the error $\vec{\mathcal{E}}_o$. If instead we start at a different starting point with the error, $\vec{\delta}_o = \alpha \vec{\mathcal{E}}_o$, then

$$\vec{Z} * \vec{\delta}_1 = \hat{\theta}[\vec{Z}* \theta[\alpha \vec{\mathcal{E}}_o]] \qquad (A9)$$

$$\vec{Z} * \vec{\delta}_1 = \alpha \hat{\theta}[Z* \theta[\vec{\mathcal{E}}_o]] \qquad (A10)$$

$$\vec{Z} * \vec{\delta}_1 = \alpha \vec{Z} * \vec{\mathcal{E}}_1 \qquad (A11)$$

Therefore,

$$\vec{Z} * (\vec{\delta}_1 - \alpha \vec{\mathcal{E}}_1) = \vec{0} \qquad (A12)$$

From the fact that the original convolution equation is the Method of Moments formulation of the physical problem which can have no currents for zero excitation, (A12) can be interpreted as indicating,

$$\vec{\delta}_1 = \alpha \vec{\mathcal{E}}_1 \qquad (A13)$$

and

$$\vec{\delta}_n = \alpha \vec{\mathcal{E}}_n \qquad (A14)$$

Since the convergence is achieved when we start with the error $\vec{\varepsilon}_o$, in the limit as n approaches ∞, $\vec{\varepsilon}_n$ approaches zero. Therefore,

$$\lim_{n\to\infty} \vec{\delta}_n = \vec{0} \qquad \text{(A15)}$$

II. CONDITION FOR CONVERGENCE

The condition for convergence for the one dimensional case is given here. Multi-dimensional cases will have similar conditions for convergence. The general solution technique as given by (A3) for the one dimensional case is,

$$\vec{Z} * \vec{J}_{n+1} = \theta[\vec{V}] + \hat{\theta}[\vec{Z} * \theta[\vec{J}_n]] \qquad \text{(A16)}$$

However, it is apparent that since the convolution can be written as a matrix multiplication operation,

$$[\mathcal{Z}]\,\vec{J}_{n+1} = \theta[\vec{V}] + \hat{\theta}[[\mathcal{Z}]\,\theta[\vec{J}_n]] \qquad \text{(A17)}$$

Here $[\mathcal{Z}]$ is the circulant matrix produced from $\vec{Z}$ and not the same as [Z].

The trunction operator $\theta[\ ]$ can also be represented as [T], a diagonal matrix with 1's at places on the diagonal corresponding to region S and zeros elsewhere. Similarly, it is easy to see that the operator $\hat{\theta}[\ ]$ can be represented as $[\hat{T}]$, a diagonal matrix with 1's at places on the diagonal corresponding to region S (i.e. the region outside S) and zeros elsewhere. Therefore (A17) can be rewritten as,

$$[\mathcal{Z}]\,\vec{J}_{n+1} = [T]\,\vec{V} + [\hat{T}][\mathcal{Z}][T]\,\vec{J}_n \qquad \text{(A18)}$$

$$\vec{J}_{n+1} = [\mathcal{Z}]^{-1}[T]\,\vec{V} + [\mathcal{Z}]^{-1}[\hat{T}][\mathcal{Z}][T]\,\vec{J}_n \qquad \text{(A19)}$$

Since, $[\mathcal{Z}]$ is the circulant matrix with Z as its rows, it has an inverse if $\vec{Z}$ has a DFT [9]. Let

$$[Q] = [\mathcal{Z}]^{-1}[T] \tag{A20}$$

$$[R] = [\mathcal{Z}]^{-1}[T][\mathcal{Z}][T] \tag{A21}$$

Therefore,

$$\vec{J}_{n+1} = [Q]\,\vec{V} + [R]\,\vec{J}_n \tag{A22}$$

It is easy to see from (A2) and (A19) that the exact solution is given by

$$\vec{J} = [Q]\,\vec{V} + [R]\,\vec{J} \tag{A23}$$

Equation (A22) can be rewritten as,

$$\vec{J}_{n+1} = [Q]\,\vec{V} + [R][Q]\,\vec{V} + [R]^2\,\vec{J}_n \tag{A24}$$

Repeated application of (A22) gives

$$\vec{J}_{n+1} = ([I]+[R]+[R]^2+\ldots)[Q]\,\vec{V} + [R]^{(n+1)}\,\vec{J}_n \tag{A25}$$

If the maximum eigenvalue of [R], λ_{max} is such that

$$|\lambda|_{max} < 1 \tag{A26}$$

Then [9],

$$[I]+[R]+[R]^2+\ldots = [I-R]^{-1} \tag{A27}$$

and

$$[R]^{(n+1)} = 0 \tag{A28}$$

as n approaches ∞.

Therefore

$$\vec{J}_{n+1} = [I-R]^{-1}[Q]\,\vec{V} \tag{A29}$$

$$[I-R]\,\vec{J}_{n+1} = [Q]\,\vec{V} \tag{A30}$$

$$\vec{J}_{n+1} = [Q]\,\vec{V} + [R]\vec{J}_{n+1} \tag{A31}$$

as n approaches ∞.

Therefore from (A23) and (A31),

$$\vec{J}_{n+1} = \vec{J} \qquad \text{(A32)}$$

as n approaches ∞, if

$$|\lambda|_{max} < 1 \qquad \text{(A33)}$$

However, the usefullness of the above condition is limited as the computation of λ_{max} requires N^2 order complex multiplications and so will take longer than the solution itself, although not as long as using Gaussian Elimination which requires complex multiplications of the order N^3.

III. ESTIMATION OF NUMERICAL ERRORS

The Discrete Convolution Method (DCM) is an iterative technique for solving the matrix equation,

$$[Z]\ \vec{J} = \vec{V} \qquad \text{(A34)}$$

formulated by the Method of Moments. This is done by looking at the above equation as a convolution equation,

$$\vec{Z} * \vec{J} = \vec{V} \qquad \text{(A35)}$$

Now, it is apparent that if we are given the answer for (A34), $\vec{J}_a$ (say), then we can use $\vec{J}_a$ to take the matrix multiplication with [Z] and get $\vec{V}_a$, which we can then check against $\vec{V}$ to see if the answer given, $\vec{J}_a$ is correct or not. Also, instead of taking FFT and IFFT, if we actually convolve, by using the relationship

$$V_m = \sum_{n=1}^{N} Z_{(m-n)}\, J_n \qquad \text{(A36)}$$

to compute $\vec{V}_a$ from $\vec{J}_a$, then the computations involved in using (A35) is identical to that involved in using (A34).

Therefore, if the matrix equation solution of (A34) using Gaussian Elimination is unique (in the sense that up to the desired precision point there are no two answers to A34, although there may be many beyond that precision point), then trying out various J_a's (chosen randomly, found by iteration or any method) in (A34), would also give us the same unique answer. Hence, trying out various J_a's in (A35), provided we actually convolve (i.e. use A36), will give the same answer. So, any difference in DCM and matrix inversion solutions will come only through

(a) the use of FFT and IFFT

(b) the fact that the iteration is carried out only up to a certain accuracy based on some criterion.

Numerical errors due to (a) can be analyzed in the following way. By [2], the use of FFT introduces the output error, the expected value of which is

$$E(\text{error}) = \sqrt{N/3}\ 2^{-b} \qquad \text{(A37)}$$

where b is the number of machine precision bits

N is the basis on which FFT is made and

E(error) is the expected value of normalized error

Since IFFT needs identical computations as FFT, IFFT causes error, the expected value of which is

$$E(\text{error}) = \sqrt{N/3}\ 2^{-b} \qquad \text{(A38)}$$

Even if the worst case occurs and errors do not cancel at all in taking FFT and IFFT, then the total convolution error is,

$$E(\text{error}) = \sqrt{N/3}\ 2^{-b+1}\ 100\% \qquad \text{(A39)}$$

To give a numerical example, for the IBM mainframe computers, b=23. For N=100000,

$$E(\text{error}) = \sqrt{100000/3} \times 2^{-22} \times 100\% \qquad \text{(A40)}$$

$$= .004\%$$

Therefore, if we check the answer by comparing $\vec{V}$ with $\vec{V}_a$ computed from (A35) by using FFT and IFFT, the numerical error in $\vec{V}_a$ would be in the fifth precision position i.e. insignificant.

Errors due to (b) will be small, provided that the problem is well behaved (i.e. the condition number of the impedance matrix [Z] is small) and that the iterations are carried out far enough so that the error in $\vec{V}$ is small. In a practical problem like the antenna array problem, the change in current for each iteration drops off very sharply, indicating that the actual error in the current is probably less than the last change. The following qualitative argument can be given in support of the above claim. From (A22),

$$\vec{J}_{n+1} = [Q]\,\vec{V} + [R]\,\vec{J}_n \qquad \text{(A41)}$$

$$\vec{J}_n = [Q]\,\vec{V} + [R]\,\vec{J}_{n-1} \qquad \text{(A42)}$$

Subtracting (A42) from (A41), we get

$$\vec{\delta}_{n+1} = [R]\,\vec{\delta}_n \qquad \text{(A43)}$$

where $\vec{\delta}_n = \vec{J}_n - \vec{J}_{n-1}$ is the change in current at n^{th} iteration. Therefore

$$|\vec{\delta}_{n+1}| \leq \|R\| |\vec{\delta}_n| \qquad \text{(A44)}$$

Now, if $\vec{\delta}_{n+1}$ is much less than $\vec{\delta}_n$ for n=1,2,3,...,N, then it is very unlikely that $\vec{\delta}_{n+1}$ would be larger than $\vec{\delta}_n$ for n>N.

IV. COMPUTER PROGRAMS AND SUBROUTINES

The computer programs and subroutines given in this section are written to verify that DCM works properly and to measure the number of iterations needed for some sample problems. No attempts have been made to optimize the computer code. In fact, the fast fourier transform and inverse fast fourier transform (FFT and IFFT) routines given are for 2^N points. This means that more points than are strictly necessary has to be taken. However, even with the relatively unoptimized code, DCM proves to be faster than other techniques for large problems.

Subroutines FFT and IFFT are for one dimensional FFT and IFFT. Subroutines TWODF and ITWODF are for two dimensional FFT and IFFT. The main program segment starting on page 47, solves the one dimensional convolution equation. This program is written to be able to solve problems with gaps. Subroutine SOLVE solves the two dimensional convolution equation. It is not written to solve the problems with gaps, however.

```
      SUBROUTINE FFT(X,N,M)
      INTEGER I,J,N,M,LE,LE1,NV2,NM1,K
      COMPLEX X(4096),U,W,T
      REAL PI
      PI=3.1415926535
      DO 20 L=1,M
      LE=2**(M+1-L)
      LE1=LE/2
      U=(1.0,0.0)
      W=CMPLX(COS(PI/FLOAT(LE1)),-SIN(PI/FLOAT(LE1)))
      DO 20 J=1,LE1
      DO 10 I=J,N,LE
      IP=I+LE1
      T=X(I)+X(IP)
      X(IP)=(X(I)-X(IP))*U
      X(I)=T
10    CONTINUE
      U=U*W
20    CONTINUE
      NV2=N/2
      NM1=N-1
      J=1
      DO 40 I=1,NM1
      IF(I.GE.J) GOTO 25
      T=X(J)
      X(J)=X(I)
      X(I)=T
25    K=NV2
26    IF(K.GE.J) GOTO 30
      J=J-K
      K=K/2
      GOTO 26
30    J=J+K
40    CONTINUE
      RETURN
      END
      SUBROUTINE IFFT(X,N,M)
      INTEGER I,J,N,M,LE,LE1,NV2,NM1,K
      COMPLEX X(4096),U,W,T
      REAL PI
      PI=3.1415926535
      DO 20 L=1,M
      LE=2**(M+1-L)
      LE1=LE/2
      U=(1.0,0.0)
      W=CMPLX(COS(PI/FLOAT(LE1)), SIN(PI/FLOAT(LE1)))
      DO 20 J=1,LE1
      DO 10 I=J,N,LE
      IP=I+LE1
      T=X(I)+X(IP)
      X(IP)=(X(I)-X(IP))*U
      X(I)=T
10    CONTINUE
      U=U*W
20    CONTINUE
      NV2=N/2
      NM1=N-1
      J=1
      DO 40 I=1,NM1
      IF(I.GE.J) GOTO 25
```

```
      T=X(J)
      X(J)=X(I)
      X(I)=T
   25 K=NV2
   26 IF(K.GE.J) GOTO 30
      J=J-K
      K=K/2
      GOTO 26
   30 J=J+K
   40 CONTINUE
      DO 50 I=1,N
      X(I)=X(I)/FLOAT(N)
   50 CONTINUE
      RETURN
      END
C**   DISCRETE CONVOLUTION METHOD TO SOLVE BOTH TOEPLITZ AND
C**   NON TOEPLITZ MATRIX EQUATIONS BY (N LOG N PROCESS) ITERATION
      INTEGER N,M
      COMPLEX X(1366),T(4096),CZERO,CUR(4096),A(4096),TSTORE
      INTEGER START(11),FINISH(10),ST,FI,REVS
      CZERO=(0.0,0.0)
    1 DO 10 I=1,1024
      A(I)=CZERO
      CUR(I)=CZERO
      T(I)=CZERO
   10 CONTINUE
      READ(1,100)N,M,NS,IFLAG,IREG,REVS
      WRITE(3,200)N,M,NS
      IF(IFLAG.NE.0) GO TO 11
      READ(1,300)  (X(I),I=1,NS)
      WRITE(3,400) (X(I),I=1,NS)
      READ(1,300)  (T(I),I=1,NS)
      WRITE(3,800) (T(I),I=1,NS)
      GO TO 13
   11 CONTINUE
      NSBY2=NS/2
      OPEN(UNIT=21,FILE='ARDATA.DAT')
      READ(21,101) (T(I),I=1,NS)
      READ(21,101) (X(I),I=1,NS)
  101 FORMAT(5E14.7)
      IF(REVS.EQ.0) GO TO 13
      DO 12 I=1,NSBY2
      TSTORE=T(I)
      IPTR=NS-I+1
      T(I)=T(IPTR)
      T(IPTR)=TSTORE
   12 CONTINUE
   13 CONTINUE
      IF(IREG.EQ.0) GO TO 14
      READ(1,100)(START(I),FINISH(I),I=1,IREG)
   14 CONTINUE
      NS2=NS+NS
      NSM1=NS-1
      DO 15 I=1,NS
      J=I+NSM1
      A(J)=X(I)
      T(NS2-I)=T(I)
   15 CONTINUE
      CALL FFT(T,N,M)
      ICOUNT=1
```

```
      IF(IREG.EQ.0) GO TO 18
      DO 16 I=1,IREG
      ST=START(I)
      FI=FINISH(I)
      DO 16 J=ST,FI
      A(J)=CZERO
   16 CONTINUE
   18 CONTINUE
   20 CONTINUE
      CALL FFT(A,N,M)
      DO 30 I=1,N
      CUR(I)=A(I)/T(I)
   30 CONTINUE
      CALL IFFT(CUR,N,M)
      WRITE(3,1000)ICOUNT
      DO 40  I=NS+1,N
      CUR(I)=CZERO
   40 CONTINUE
      IF(IREG.EQ.0) GO TO 46
      DO 45 I=1,IREG
      ST=START(I)
      FI=FINISH(I)
      DO 45 J=ST,FI
      CUR(J)=CZERO
   45 CONTINUE
   46 CONTINUE
      WRITE(3,600)(CUR(I),I=1,NS)
      CALL FFT(CUR,N,M)
      DO 50 I=1,N
      A(I)=CUR(I)*T(I)
   50 CONTINUE
      CALL IFFT(A,N,M)
      WRITE(3,500)(A(I),I=NS,NS+NSM1)
      CUMERR=0.0
      EMAX=0.0
      IF(IREG.NE.0) GO TO 52
      DO 51 I=1,NS
      J=I+NSM1
      TSTORE=X(I)
      ERROR=CABS(A(J)-TSTORE)/CABS(TSTORE)
      A(J)=TSTORE
      IF(ERROR.GT.EMAX) EMAX=ERROR
      CUMERR=CUMERR+ERROR
   51 CONTINUE
      GO TO 54
   52 CONTINUE
      ST=1
      DO 53 I=1,IREG+1
      IF(I.NE.1) ST=FINISH(I-1)+1
      FI=START(I)-1
      IF(I.EQ.IREG+1)FI=NS
      DO 53 J=ST,FI
      K=J+NSM1
      TSTORE=X(J)
      ERROR=CABS(A(K)-TSTORE)/CABS(TSTORE)
      IF(ERROR.GT.EMAX) EMAX=ERROR
      CUMERR=CUMERR+ERROR
      A(K)=TSTORE
   53 CONTINUE
   54 CONTINUE
```

```
      EMAX=EMAX*100.0
      CUMERR=100.0*CUMERR/FLOAT(NS)
      WRITE(3,900) EMAX,CUMERR
      WRITE(3,700)
      READ(1,100)IFLAG
      IF(IFLAG.EQ.0) GO TO 55
C      CALL IFFT(CUR,N,M)
C      WRITE(3,600)(CUR(I),I=1,NS)
C      WRITE(3,500)(A(I),I=NS,NS+NSM1)
      IF(IFLAG.EQ.1) STOP
      WRITE(3,1100)
      GO TO 1
   55 CONTINUE
      ICOUNT=ICOUNT+1
      GO TO 20
  100 FORMAT(10I0)
  200 FORMAT(1H ,' N M NS',3I7)
  300 FORMAT(10E0.0)
  400 FORMAT(1H ,'EXCITATION VECTOR'/(1H ,10E11.4))
  500 FORMAT(1H ,'RESULTANT FIELD'/(1H ,10E11.4))
  600 FORMAT(1H ,'CURRENTS'/(1H ,10E11.4))
  700 FORMAT(1H ,'CONTINUE ITERATIONS? 0 FOR YES,',
     $           ' 1 FOR NO AND RETURN')
  800 FORMAT(1H ,'GREEN"S FUNCTION'/(10E11.4))
  900 FORMAT(1H ,'MAX FIELD ERROR=',E10.3,'%'/
     $       1H ,'AVERAGE ERROR  =',E10.3,'%')
 1000 FORMAT(1H ,//1H ,'ITERATION NUMBER',I7//)
 1100 FORMAT(1H ,//1H ,'**********NEXT PROBLEM**********'//)
      END
```

```
      SUBROUTINE FFT(X,M,START,STEP)
      COMPLEX X(16384),U,W,T
      INTEGER START,STEP,SDIFF
      N=2**M
      SDIFF=STEP-START
      NV2  =N/2*STEP
      NM1  =(N-2)*STEP+START
      N    =(N-1)*STEP+START
      J    =START
      DO 8 I=START,NM1,STEP
      IF(I.GE.J) GO TO 5
      T    =X(J)
      X(J) =X(I)
      X(I) =T
    5 K    =NV2
    6 IF(K-SDIFF.GE.J) GO TO 7
      J    =J-K
      K    =K/2
      GO TO 6
    7 J    =J+K
    8 CONTINUE
      PI   =3.141592653589793
      DO 20 L=1,M
      LE   =2**L
      LE1=LE/2
      LSTEP=LE1*STEP
      U    =(1.0,0.0)
      ANGLE=PI/FLOAT(LE1)
      W    =CMPLX(COS(ANGLE),-SIN(ANGLE))
      LE1  =LSTEP+START-STEP
      LE   =LE*STEP
      DO 20 J=START,LE1,STEP
      DO 10 I=J,N,LE
      IP   =I+LSTEP
      T    =X(IP)*U
      X(IP)=X(I)-T
      X(I) =X(I)+T
   10 CONTINUE
      U=U*W
   20 CONTINUE
      RETURN
      END
C
      SUBROUTINE TWODF(X,N,M,L,MM,LM)
      COMPLEX X(16384)
      INTEGER START,STEP
      START=1
      STEP =1
      DO 10 I=1,M
      CALL FFT(X,LM,START,STEP)
      START=START+L
   10 CONTINUE
      STEP =L
      DO 20 I=1,L
      START=I
      CALL FFT(X,MM,START,STEP)
   20 CONTINUE
      RETURN
      END
      SUBROUTINE IFFT(X,M,START,STEP)
```

```
      COMPLEX X(16384),U,W,T
      INTEGER START,STEP,SDIFF
      N=2**M
      SDIFF=STEP-START
      NV2  =N/2*STEP
      NN1  =(N-2)*STEP+START
      NEXP =(N-1)*STEP+START
      J    =START
      DO 8 I=START,NN1,STEP
      IF(I.GE.J) GO TO 5
      T    =X(J)
      X(J) =X(I)
      X(I) =T
    5 K    =NV2
    6 IF(K-SDIFF.GE.J) GO TO 7
      J    =J-K
      K    =K/2
      GO TO 6
    7 J    =J+K
    8 CONTINUE
      PI   =3.14159265358979
      DO 20 L=1,M
      LE   =2**L
      LE1=LE/2
      LSTEP=LE1*STEP
      U    =(1.0,0.0)
      ANGLE=PI/FLOAT(LE1)
      W    =CMPLX(COS(ANGLE),SIN(ANGLE))
      LE1  =LSTEP+START-STEP
      LE   =LE*STEP
      DO 20 J=START,LE1,STEP
      DO 10 I=J,NEXP,LE
      IP   =I+LSTEP
      T    =X(IP)*U
      X(IP)=X(I)-T
      X(I) =X(I)+T
   10 CONTINUE
      U=U*W
   20 CONTINUE
      RETURN
      END
C
      SUBROUTINE ITWODF(X,N,M,L,NN,LN)
      COMPLEX X(16384)
      INTEGER START,STEP
      START=1
      STEP =1
      DO 10 I=1,N
      CALL IFFT(X,LN,START,STEP)
      START=START+L
   10 CONTINUE
      STEP =L
      DO 20 I=1,L
      START=I
      CALL IFFT(X,NN,START,STEP)
   20 CONTINUE
      FN=FLOAT(N)
      DO 30 I=1,N
      X(I)=X(I)/FN
   30 CONTINUE
```

```
RETURN
END
```

```
      SUBROUTINE SOLVE(A,B,LO,MO,NO,LH,MM,L,M,N,FXCITE)
C     ROUTINE TO SOLVE THE MATRIX EQUATION A X = B
      LOGICAL FXCITE
      COMPLEX CTEMP,CZERO,A(16384),X(16384),V(16384),B(1849),Y(1849)
      INTEGER FLAG1,FLAG2,COUNT
      REAL CHNAVG,CHNMAX,CHANGE
      CZERO=(0.0,0.0)
C     ZEROIZE Y AND V (EXPENDED B)
      DO 10 I=1,N
      V(I)=CZERO
   10 CONTINUE
      DO 20 I=1,MO
      Y(I)=CZERO
   20 CONTINUE
      IF(.NOT.FXCITE) GO TO 65
      DO 40 I=1,MO
      IPTR=(I-1)*L
      JPTR=IPTR+LO+LO-1
      DO 30 J=IPTR+1,IPTR+LO-1
      A(JPTR)=A(J)
      JPTR=JPTR-1
   30 CONTINUE
   40 CONTINUE
C     FILL UP A ARRAY AND V ARRAY
      DO 60 I=1,MO-1
      IPTR=(I-1)*L
      JPTR=(MO+MO-I-1)*L
      DO 50 J=1,LO+LO-1
      A(JPTR+J)=A(IPTR+J)
   50 CONTINUE
   60 CONTINUE
      CALL TWODF(A,N,M,L,MM,LM)
   65 COUNT=0
   70 JPTR=1
      COUNT=COUNT+1
      DO 90 I=1,MO
      IPTR=L*(MO-2+I)+LO
      DO 80 J=1,LO
      V(IPTR)=B(JPTR)
      JPTR=JPTR+1
      IPTR=IPTR+1
   80 CONTINUE
   90 CONTINUE
C     FIND V TRANSFORMED AND COMPUTE X TRANSFORMED
      CALL TWODF(V,N,M,L,MM,LM)
      DO 100 I=1,N
      X(I)=V(I)/A(I)
  100 CONTINUE
C     GET X FROM X TRANSFORMED
      CALL ITWODF(X,N,M,L,MM,LM)
C     TRUNCATE X AND SAVE X AFTER COMPUTING THE CONVERGENCE CRITERION
      CHNAVG=0.0
      CHNMAX=0.0
      DO 120 I=1,N
      IPTR=I/L
      JPTR=I-IPTR*L
      IF(JPTR.LE.LO.AND.JPTR.NE.0.AND.IPTR.LT.NO) GO TO 110
      X(I)=CZERO
      GO TO 120
  110 IPTR=IPTR*LO+JPTR
```

```
      CTEMP=X(I)
      CHANGE=CABS(CTEMP-Y(IPTR))/CABS(CTEMP)
      Y(IPTR)=CTEMP
      CHNAVG=CHNAVG+CHANGE
      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
  120 CONTINUE
      CHNAVG=(CHNAVG*100.0)/FLOAT(MO)
      CHNMAX=CHNMAX*100.0
      WRITE(3,1200) CHNAVG,CHNMAX,COUNT
C     FIND THE TRANSFORM OF TRUNCATED X
      CALL TWODF(X,N,M,L,MM,LM)
C     COMPUTE V TRANSFORMED
      DO 130 I=1,N
      V(I)=A(I)*X(I)
  130 CONTINUE
C     GET V FROM V TRANSFORMED
      CALL ITWODF(V,N,M,L,MM,LM)
C     COMPUTE THE ERROR CRITERION
      CHNAVG=0.0
      CHNMAX=0.0
      JPTR=1
      DO 150 I=1,MO
      IPTR=L*(MO-2+I)+LO
      DO 140 J=1,LO
      CTEMP=B(JPTR)
      CHANGE=CABS(CTEMP-V(IPTR))/CABS(CTEMP)
      CHNAVG=CHNAVG+CHANGE
      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
      IPTR=IPTR+1
      JPTR=JPTR+1
  140 CONTINUE
  150 CONTINUE
C     ASK WHETHER OR NOT TO STOP AFTER REPORTING % FIELD ERROR
      CHNAVG=(CHNAVG*100.0)/FLOAT(MO)
      CHNMAX=CHNMAX*100.0
      WRITE(3,1300)CHNMAX,CHNAVG
      READ(1,1100)FLAG1
      IF(FLAG1.EQ.0) GO TO 70
      WRITE(3,1400) (Y(I),I=1,MO)
C     ASK IF FIELD SHOULD BE PRINTED OUT ALSO
      WRITE(3,1500)
      READ(1,1100)FLAG2
      IF(FLAG2.NE.0)RETURN
      WRITE(3,1600) (V(I),I=1,N)
      RETURN
 1000 FORMAT(10E0.0)
 1100 FORMAT(4I0)
 1200 FORMAT(1H ,'AVG CURRENT CHANGE=',E14.7,' %'/
     $       1H ,'MAX CURRENT CHANGE=',E14.7,' %'/
     $       1H ,'AFTER',I4,' ITERATIONS'/)
 1300 FORMAT(1H ,'MAX FIELD ERROR = ',E15.7,' %'/
     $    1H ,'AVG FIELD ERROR = ',E15.7,' %'/
     $    1H ,'CONTINUE ITERATIONS? 0 FOR YES, 1 FOR NO, AND RETURN'/)
 1400 FORMAT(1H ,'CURRENTS'//(1H ,10E11.4))
 1500 FORMAT(1H ,'PRINT FIELDS? 0 FOR YES, 1 FOR NO, THEN RETURN'/)
 1600 FORMAT(1H ,'RESULTANT FIELDS'//(1H ,10E11.4))
      END
```

REFERENCES

[1] R. F. Harrington, "Field Computation by Moment Methods," The Macmillan Company, New York, 1968.

[2] A. V. Oppenhiem and R. W. Schafer, "Digital Signal Processing," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[3] D. C. Kuo and B. J. Strait, "Improved Programs for Analysis of Radiation and Scattering by Configurations of Arbitrarily Bent Thin Wires," Scientific Report No. 15 on Contract No. F19628-68-C-0180 with Air Force Cambridge Research Laboratories, Report AFCRL-72-0051, January 1972.

[4] J. Luzwick, E. Ngai, and A. T. Adams, "Analysis of a Large Linear Antenna Array of Uniformly-Spaced Thin-Wire Dipoles parallel to a Perfectly Conducting Plane," Scientific Report TR-80-2, Syracuse University, March 1980.

[5] E. Ngai and A. T. Adams, "A Computer Program for a Planar Array of Thin Wire Dipoles parallel to a Perfectly Conducting Plane," Scientific Report TR-80-3, Syracuse University, March 1980.

[6] W. L. Ko and R. Mittra, "A Method for Combining Integral Equation and Asymptotic Techniques for solving Electromagnetic Scattering Problems," Technical Report 76-6 on Grant No. DAHC04-74-G-0113 with U. S. Army Research Office and Grant No. N00014-75-C-0293 with Office of Naval Research, May 1976.

[7] R. Mittra, Y. Rahmat-Samii, and W. L. Ko, "Spectral Theory of Diffraction," Appl. Phys., vol. 10, pp. 1-13, 1976.

[8] Y. Rahmat-Samii and R. Mittra, "Spectral Analysis of high frequency diffraction of an arbitrary incident field by a half plane--Comparison with four asymptotic techniques," Rad. Sci., vol. 13, pp. 31-48, 1978.

[9] J. R. Westlake, "A Handbook of Numerical Matrix Inversion and Solution of Linear Equations," John Wiley & Sons, Inc., New York, 1968.

END

FILMED

1-84

DTIC