SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS BY THE DISCRETE CONVOLUTION METHOD

This report gives solution results and computer programs for the Discrete Convolution Method applied to scattering from a helix, to radiation from planar array antennas with antenna elements arranged in triangular patterns (solved using one expansion function per element), and to radiation from planar array antennas with antenna elements arranged in triangular patterns (solved using three expansion functions per element). It also gives a brief discussion on the spatial domain interpretation of the spectral domain iteration technique.
SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS

BY THE DISCRETE CONVOLUTION METHOD

by

Htay L. Nyo
Roger F. Harrington

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

Technical Report No. 23
February 1984

Contract No. N00014-76-C-0225

Approved for public release; distribution unlimited

Reproduction in whole or in part permitted for any purpose of the United States Government.

Prepared for

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
ARLINGTON, VIRGINIA 22217
SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS

BY THE DISCRETE CONVOLUTION METHOD

by

Htay L. Nyo
Roger F. Harrington

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

Technical Report No. 23
February 1984

Contract No. N00014-76-C-0225

Approved for public release; distribution unlimited

Reproduction in whole or in part permitted for any purpose of the United States Government.

Prepared for

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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. SAMPLE COMPUTATIONS AND COMMENTS</td>
<td>1</td>
</tr>
<tr>
<td>III. CONCLUSIONS</td>
<td>33</td>
</tr>
</tbody>
</table>

### APPENDIX

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. SPATIAL DOMAIN INTERPRETATION OF THE SPECTRAL ITERATION TECHNIQUES</td>
<td>35</td>
</tr>
<tr>
<td>B. COMPUTER PROGRAMS</td>
<td>43</td>
</tr>
<tr>
<td>I. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE HELIX PROBLEM</td>
<td>43</td>
</tr>
<tr>
<td>II. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEM</td>
<td>50</td>
</tr>
<tr>
<td>(SINGLE EXPANSION FUNCTION PER ANTENNA ELEMENT)</td>
<td></td>
</tr>
<tr>
<td>III. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEMS</td>
<td>58</td>
</tr>
<tr>
<td>(THREE EXPANSION FUNCTIONS PER ANTENNA ELEMENT)</td>
<td></td>
</tr>
</tbody>
</table>

REFERENCES                                                                 | 74   |
I. INTRODUCTION

In the first report [1] on discrete convolution method for solving some large moment equations, three types of one and two dimensional problems were solved. In this report, we solve three more types of one and two dimensional problems. They are

(i) scattering from a helix
(ii) planar arrays with antenna elements arranged in triangular pattern, solved using one expansion function per element
(iii) planar arrays with antenna elements arranged in triangular pattern, solved using three expansion functions per element

As in the first report, the computing time measurements (made on a KL/10 machine) and the number of iterations needed for the given accuracy are listed. In addition, the array factor of the planar arrays are given.

II. SAMPLE COMPUTATIONS AND COMMENTS

The "one" dimensional problem is scattering from a helix. Fig. 1 shows the problem of scattering from a helix. The MOM formulation of the helix problem requires that the helix be subsectioned into equal length segments. If we number the helix segments so that the segment numbers are in consecutive increasing order from top to bottom, then it is
Fig. 1. The helical wire scatterer
Fig. 2. The helical wire scatterer with gaps
easy to see that the mutual coupling between segments as given by the MOM impedance matrix $Z$ is

$$Z_{mn} = Z(m-n)$$ (1)

i.e., the value is dependent only on the difference between segment numbers. Therefore the $Z$ matrix will be Toeplitz if the helix is a complete helix as shown in Fig. 1. The $Z$ matrix will be non-Toeplitz if the helix has gaps in between as shown in Fig. 2. It is apparent from the discussions of other one dimensional problems that both problems can be solved using the one dimensional DCM technique. For the helix with gaps all we need is to insert phantom elements as shown in Fig. 2.

Table 1 gives the number of iterations needed to get the required degree of accuracy for the helix problem. We can see that the number of iterations needed is practically independent of the length of the helix.

The problem of a planar array with antennas arranged in triangular patterns instead of rectangular, can also be formulated as a two dimensional convolution equation by adding phantom elements (as shown in Fig. 4), to make a parallelogram. The triangular pattern arrangement is shown in Fig. 3.

The MOM formulation using one expansion per antenna then gives 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 phantom
Table 1. Results for some helical scatterer problems

<table>
<thead>
<tr>
<th>Radius</th>
<th>Pitch</th>
<th>Number of turns</th>
<th>Number of segments</th>
<th>Number of iterations</th>
<th>Field</th>
<th>Last error</th>
<th>Field change</th>
<th>Last current change</th>
</tr>
</thead>
<tbody>
<tr>
<td>.125</td>
<td>.30</td>
<td>6</td>
<td>120</td>
<td>22</td>
<td>.693</td>
<td>.823</td>
<td>.0498</td>
<td>.1</td>
</tr>
<tr>
<td>.125</td>
<td>.30</td>
<td>6</td>
<td>120</td>
<td>23</td>
<td>1.01</td>
<td>1.11</td>
<td>.141</td>
<td>.113</td>
</tr>
<tr>
<td>.15</td>
<td>.50</td>
<td>6</td>
<td>120</td>
<td>8</td>
<td>.812</td>
<td>1.39</td>
<td>.0504</td>
<td>.202</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0562</td>
<td>.0958</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0036</td>
<td>.0142</td>
</tr>
<tr>
<td>.15</td>
<td>.30</td>
<td>6</td>
<td>120</td>
<td>10</td>
<td>1.56</td>
<td>1.17</td>
<td>.177</td>
<td>.248</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0668</td>
<td>.0496</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0076</td>
<td>.0109</td>
</tr>
<tr>
<td>.15</td>
<td>.30</td>
<td>8</td>
<td>160</td>
<td>12</td>
<td>.568</td>
<td>1.21</td>
<td>.0375</td>
<td>.0778</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0697</td>
<td>.153</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0046</td>
<td>.0097</td>
</tr>
<tr>
<td>.15</td>
<td>.30</td>
<td>8</td>
<td>160</td>
<td>13</td>
<td>.453</td>
<td>.883</td>
<td>.0572</td>
<td>.0467</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0934</td>
<td>.184</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0118</td>
<td>.0097</td>
</tr>
</tbody>
</table>
Table 1. continued

<table>
<thead>
<tr>
<th>Radius</th>
<th>Pitch</th>
<th>Number of turns</th>
<th>Number of segments</th>
<th>Number of iterations</th>
<th>Field change (%)</th>
<th>Last error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.15</td>
<td>0.30</td>
<td>8</td>
<td>160</td>
<td>11</td>
<td>1.07</td>
<td>0.785</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.104</td>
<td>0.141</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0821</td>
<td>0.0602</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0082</td>
<td>0.0114</td>
</tr>
<tr>
<td>0.15</td>
<td>0.30</td>
<td>16</td>
<td>320</td>
<td>13</td>
<td>0.395</td>
<td>0.659</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0154</td>
<td>0.0295</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0848</td>
<td>0.139</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0033</td>
<td>0.00634</td>
</tr>
<tr>
<td>0.25</td>
<td>0.50</td>
<td>8</td>
<td>160</td>
<td>11</td>
<td>0.537</td>
<td>1.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0282</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0495</td>
<td>0.153</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0026</td>
<td>0.0184</td>
</tr>
</tbody>
</table>

Here, Radius is the radius of the helix in wavelengths
Pitch is the pitch of the helix in wavelengths
the wire radius in all cases is .006 wavelengths
first entries for each problem in the last two columns
are maximum values and second entries are average values
* indicates that polarity of impressed E field is not as
shown in Fig. 10 but is z directed
* indicates that the impressed field is not from the
direction shown in Fig. 10 but is from z direction
Fig. 3. The triangular pattern arrangement.

Fig. 4. A planar array with antennas arranged in triangular pattern.
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 gives a much more accurate result and needs five times more computing time for the DCM. With LU decomposition method computing time will go up twenty seven times the already large value.

Table 2 lists the computing time and number of iterations needed for the DCM solution using one expansion function per antenna element. Table 3 lists the computing time and number of iterations needed for the DCM solution using three expansion functions per antenna element. All the problems are for planar arrays with 0.48 wavelength antennas one-quarter wavelength in front of the infinite ground plane. The separation between antennas is one-half wavelength in either direction.

The graphs given in Figs. 6, 7, 8, 9, 10, and 11 are the arrays factors for the planar arrays with triangular pattern arrangement solved by the DCM using one expansion function per antenna element. The array factors are computed in the plane perpendicularly bisecting the array as shown in Fig. 5. Angle measurements are as shown. In all the figures, the solid lines give the array factors for the solutions which take the mutual coupling between antennas into account and the dashed lines are for the idealized solutions which do not take the mutual coupling into
Table 2. Results for some planar arrays with triangular pattern arrangement

<table>
<thead>
<tr>
<th>N</th>
<th>Excitation</th>
<th>I</th>
<th>Computing Time (secs)</th>
<th>Field Error (%)</th>
<th>Current Change (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Uniform</td>
<td>5</td>
<td>4</td>
<td>.1789</td>
<td>.664</td>
</tr>
<tr>
<td></td>
<td>Beam steer</td>
<td>6</td>
<td>4</td>
<td>.0867</td>
<td>.378</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.1124</td>
<td>.595</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0554</td>
<td>.173</td>
</tr>
<tr>
<td>76</td>
<td>Uniform</td>
<td>6</td>
<td>20</td>
<td>.1447</td>
<td>.489</td>
</tr>
<tr>
<td></td>
<td>Beam steer</td>
<td>6</td>
<td>20</td>
<td>.0283</td>
<td>.116</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.2768</td>
<td>.796</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0333</td>
<td>.098</td>
</tr>
<tr>
<td>372</td>
<td>Uniform</td>
<td>5</td>
<td>105</td>
<td>.4950</td>
<td>1.674</td>
</tr>
<tr>
<td></td>
<td>Beam steer</td>
<td>6</td>
<td>105</td>
<td>.0467</td>
<td>.174</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.3636</td>
<td>.863</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0163</td>
<td>.0456</td>
</tr>
</tbody>
</table>

Here, N 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.
Table 3. Results for some planar arrays with triangular pattern arrangement (Multiple Expansion Solutions)

<table>
<thead>
<tr>
<th>N</th>
<th>Excitation</th>
<th>I</th>
<th>Computing Time (secs)</th>
<th>Field Error (%)</th>
<th>Current Change (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>76</td>
<td>Uniform</td>
<td>6</td>
<td>67</td>
<td>.0401</td>
<td>.3612</td>
</tr>
<tr>
<td></td>
<td>Beam steer</td>
<td>6</td>
<td>67</td>
<td>.0027</td>
<td>.0836</td>
</tr>
<tr>
<td></td>
<td>(xz 45°)</td>
<td></td>
<td></td>
<td>.078</td>
<td>.6898</td>
</tr>
<tr>
<td></td>
<td>Beam steer</td>
<td>6</td>
<td>67</td>
<td>.0032</td>
<td>.0704</td>
</tr>
<tr>
<td></td>
<td>(yz 135°)</td>
<td></td>
<td></td>
<td>.0251</td>
<td>.527</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0016</td>
<td>.0493</td>
</tr>
<tr>
<td>372</td>
<td>Uniform</td>
<td>5</td>
<td>165</td>
<td>.1450</td>
<td>1.298</td>
</tr>
<tr>
<td></td>
<td>Beam steer</td>
<td>5</td>
<td>165</td>
<td>.0048</td>
<td>.134</td>
</tr>
<tr>
<td></td>
<td>(xz 45°)</td>
<td></td>
<td></td>
<td>.2928</td>
<td>2.1100</td>
</tr>
<tr>
<td></td>
<td>Beam steer</td>
<td>5</td>
<td>165</td>
<td>.0047</td>
<td>.0994</td>
</tr>
<tr>
<td></td>
<td>(yz 135°)</td>
<td></td>
<td></td>
<td>.0706</td>
<td>1.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.0029</td>
<td>.0748</td>
</tr>
</tbody>
</table>

Here, N 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.
Fig. 5. The relative position of the plane in which the array factors are computed.
Fig. 6 Array factor of the twelve antenna planar array with uniform excitation
Fig. 7 Array factor of the twelve antenna planar array with excitation to give a 45 degrees scan.
Fig. 9 Array factor of the seventy six antenna planar array with uniform excitation
Fig. 10 Array factor of the three hundred and seventy two antenna planar array with uniform excitation
Fig. 11 Array factor of the three hundred and seventy-two antenna planar array with excitation to give a 45 degrees scan
As we can see from the graphs, for larger arrays, the main beam is not affected by the mutual coupling but the side lobes and the nulls are affected strongly.

The graphs given in Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, and 24 are the array factors for the planar arrays with triangular pattern arrangement solved by the DCM using three expansion functions per antenna element. The array factors are computed in the xz and yz planes perpendicularly bisecting the array as shown in Fig. 12. Angle measurements are as shown. In all the figures, the solid lines give the array factors for the solutions which take the mutual coupling between antennas into account and the dashed lines are for the idealized solutions which do not take the mutual coupling into account.

The array factor is defined as the far field pattern divided by the element factor. Therefore for the DCM solution using one expansion function per element, the element factor is the far field pattern of the expansion function and so the array factor can be and is computed directly from the solved currents. However, with three expansions per antenna element, the current distribution over each element is different from the others. Therefore, since our main purpose in computing array factors is to compare the far field patterns, we define a normalized array factor as the total far field pattern divided by the far
Fig. 12. The relative position of the planes in which the array factors are computed.
Fig. 13 Array factor in the xz plane of the seventy six antenna planar array with uniform excitation
Fig. 14 Array factor in the yz plane of the seventy-six antenna planar array with uniform excitation
Fig. 15 Array factor in the xz plane of the seventy six antenna planar array with excitation to give a 45 degrees scan in the xz plane
Array factor in the $yz$ plane of the seventy-six antenna planar array with excitation to give a 45 degrees scan in the $xz$ plane.
Fig. 17 Array factor in the xz plane of the seventy six antenna planar array with excitation to give a 135 degrees scan in the yz plane.
Fig. 18 Array factor in the yz plane of the seventy six antenna planar array with excitation to give a 135 degrees scan in the yz plane.
Fig. 19 Array factor in the xz plane of the three hundred and seventy two antenna planar array with uniform excitation
Fig. 20 Array factor in the yz plane of the three hundred and seventy-two antenna planar array with uniform excitation.
Fig. 21 Array factor in the $xz$ plane of the three hundred and seventy two antenna planar array with excitation to give 45 degrees scan in the $xz$ plane.
Fig. 22 Array factor in the yz plane of the three hundred and seventy two antenna planar array with excitation to give 45 degrees scan in the xz plane
Fig. 23 Array factor in the xz plane of the three hundred and seventy two antenna planar array with excitation to give 135 degrees scan in the yz plane
Fig. 24 Array factor in the yz plane of the three hundred and seventy two antenna planar array with excitation to give 135 degrees scan in the yz plane.
field pattern of an element with only one expansion function. This is the array factor used in the graphs discussed above.

\[
(\text{NAF})_n = \frac{(\text{FP})_n}{(\text{EP})_1}
\]

Here \((\text{NAF})_n\) is the normalized array factor for the DCM solution using \(n\) expansions per antenna element. \((\text{FP})_n\) is the far field pattern for the DCM solution using \(n\) expansion functions per antenna element. \((\text{EP})_1\) is the element pattern or the far field pattern of an element with only one expansion function per element. Since both \((\text{FP})_n\) and \((\text{EP})_1\) goes to zero when the angles are either 0 or 180 degrees, \((\text{NAF})_n\) is indeterminate at those angles.

As we can see from the graphs, for larger arrays, the main beam is not effected by the mutual coupling but the side lobes and the nulls are effected strongly. In addition we can see that the difference between the actual and the ideal array factors are more pronounced for the 45 degrees beam steering in the xz plane. This is expected since the coupling between the antenna elements that are adjacent in the x-direction is stronger than the coupling between the antenna elements that are adjacent in the y-direction. Also, we can see that the array factors computed from the solutions using one expansion function per antenna and the
array factors computed from the solutions using three expansion functions per antenna are surprisingly close. This indicates that at least for the given spacings and antenna sizes, the solutions using only one expansion function per antenna may be good enough for most purposes.

III. CONCLUSIONS

Three more types of problems suggested in the first report on DCM [1] are solved and the number of iterations needed are found to be reasonably small for the antenna array problems. The helix problems required more iterations but the number of iterations needed for the required degree of accuracy is found to be practically independent of the length of the helix.

Three other major types of problems formulated or suggested in [1] still remains to be solved. They are

(i) scattering from a arbitrarily shaped planer conducting surfaces or dielectric sheets

(ii) scattering from a imperfectly conducting or dielectric bodies

(iii) planar array antenna backed by a finite sized ground plane

The major question here is wherether or not convergence will be achieved for these types of problems. The condition for convergence is that the absolute value of the largest
eigenvalue of $\mathcal{L}$ must be less than 1 as shown in [1]. There are some practical applications where the solutions of these types of problems are useful. Therefore, solving some representative problems are suggested as possible future extension to this report.
APPENDIX A
SPATIAL DOMAIN INTERPRETATION OF THE SPECTRAL ITERATION TECHNIQUES

Both the spectral solution techniques (STD, SIT etc.) [2], [3], [4], [5] and the Discrete Convolution Method (DCM) [1] use the discrete Fourier Transform to solve electromagnetic fields problems. The following spatial domain interpretation of the spectral solution techniques may give a better understanding of the differences between the methods.

The electric field integral equation for a perfectly conducting scatterer or radiator is [2]

\[( \mathbf{G} * \mathbf{J} )_\mathbf{r} = -\mathbf{E}_\mathbf{t} \quad r \in S \quad (A1) \]

Here \( \mathbf{G} \) is the Green's Dyadic, \( \mathbf{J} \) is the surface current density on surface \( S \) and \( \mathbf{E}_\mathbf{t} \) is the tangential component of the impressed field. We can extend the field equation over all space so that (A1) becomes

\[ \mathbf{G} * \mathbf{J} = -\mathbf{E}_\mathbf{i} + \mathbf{F} \quad (A2) \]
where \( \vec{F} \) is the field outside the surface \( S \). The Fourier transformed version of (A2) reads

\[
\tilde{G}(\Omega) \tilde{J}(\Omega) = -\tilde{E}_I(\Omega) + \tilde{F}(\Omega)
\]

(A3)

\( \Omega = (\omega^x, \omega^y, \omega^z) \) in general

A formal solution to (A3) is

\[
\tilde{J}(\Omega) = \tilde{G}^{-1}(\Omega) \tilde{E}(\Omega)
\]

(A4)

where \( \tilde{E}(\Omega) \) is \(-\tilde{E}_I(\Omega) + \tilde{F}(\Omega)\).

The spectral solution techniques essentially consist of iterative techniques that find \( \vec{F} \) and thus \( \vec{J} \). However since \( \vec{F} \) cannot be found in closed form, the solution is a numerical solution. Therefore (A4) and hence (A3) is solved only for a finite number of specific discrete frequencies. The solution we get is an exact solution of (A3) and hence of (A1) only if \( \vec{F} \) is known exactly and if the solution is for all frequencies. In the following discussion we will set aside the question of inaccuracies due to inexact knowledge of \( \vec{F} \) because it is a question of convergence. Here we are discussing spatial domain interpretation rather than the convergence of the solution. Therefore

(a) since (A3) is satisfied only for discrete frequencies, we are sampling in frequency domain and we are solving the following set of equations
instead of (A3).

(b) Since we cannot solve for infinite number of discrete frequencies, we solve for only a finite number. This amounts to truncating in frequency domain. If we denote the truncating function by \( \tilde{H}(\Omega) \), we are solving the following set of equations

\[
\tilde{H}(\Omega) \tilde{G}(\Omega) \tilde{J}(\Omega) \delta(\Omega - \Omega_n) = \tilde{H}(\Omega) \tilde{E}(\Omega) \delta(\Omega - \Omega_n)
\]  

(A6)

instead of either (A3) or (A5).

Before we give the spatial domain interpretation of (A6), we will first examine the errors due to above approximations from a frequency domain perspective. It is well known \([6]\) that sampling in the spatial domain will lead to aliasing in frequency domain if the function sampled is not limited in frequency. By reason of symmetry, we can easily show that sampling in frequency domain will lead to aliasing in the spatial domain if the function sampled is not limited in space. In solving (A3), we used sampling on both sides of the equation. The surface current \( \mathbf{J} \) is indeed limited in space (being confined to conductor surface \( \mathbf{S} \) but
the field $\vec{E}$ is not. Therefore, there will be aliasing irrespective of the sampling rate. However, since the field $\vec{E}$ decays outside the surface, aliasing effect can be reduced by a high enough sampling rate so that the overlap occurs in the region where $\vec{E}$ has already decayed to a small value. Truncation of the frequency range by $\tilde{\mathcal{H}}(\mathcal{K})$ (the windowing function) on both sides of the equation leads to Gibb's phenomenon. This windowing error can be reduced by using different windowing strategies discussed in [6]. A good example is the Hamming windowing function.

We will now look at the above errors from a spatial domain perspective. In spectral solution techniques, solution to (A6) is found by using the Discrete Fourier Transform. This is possible if $\Omega_n$'s are chosen at equally spaced intervals. It can be proved [6] that (A6) in this case is equivalent to the Discrete Convolution equation

$$\tilde{G}(K) \ast \tilde{J}(K) = \tilde{E}(K) \quad K = (k_x, k_y, k_z)$$

$$k_x = 0, 1, \ldots, L$$
$$k_y = 0, 1, \ldots, M$$
$$k_z = 0, 1, \ldots, N$$

Up to this point, we have been discussing the general multi-dimensional case. From here on however, we will confine our discussion to the one dimensional case. Discussion for two or three dimensional problems would be similar. (We will use lower case indices instead of higher
case, to indicate one dimensionality.) Since $\bar{E}_1$ is known exactly and $\bar{F}$, once we achieve convergence, is also known exactly, $\bar{E}(k)$ represents the sampled value at $k^{th}$ point of the exact field $\bar{E}$ [2], [5]. Also, $\bar{J}$ is the required solution and hence $\bar{J}(k)$ represents the sampled value at $k^{th}$ point of the current $\bar{J}$. But $\bar{G}(k)$ is not the sampled value of $\bar{G}$. Instead, $\bar{G}(k)$ is the inverse Discrete Fourier Transform (based on $N+1$ points) of $\bar{G}(\omega)$.

$$\bar{G}(k) = \sum_{n=-N/2}^{N/2} \bar{G}(n) e^{j\frac{2\pi n}{N+1} k n} \quad (A8)$$

Therefore $\bar{G}(k)$ is the inverse Fourier Transform of the sampled and truncated (or in general windowed) $\bar{G}(\omega)$.

It is well known [6] that sampling in spatial domain will lead to a function in frequency domain which consists of periodic repetitions of the original frequency domain function. By reason of symmetry between the domains, we can easily show that sampling in frequency domain will lead to a function in spatial domain which consists of periodic repetitions of the original spatial domain function. Therefore, sampling of $\bar{G}(\omega)$ will cause the inverse of the sampled function $\bar{G}_s$ to consist of periodic repetitions of the original spatial domain function $\bar{G}$. This will cause overlaps as shown in Fig. 25 for the one dimensional case. But $G$ is the Green's Dyadic, the field due to a unit impulse
Fig. 25  Relationship between $\frac{\partial \overline{G}}{\partial S}$ and $\overline{G}$ for the one dimensional case
current at the origin. Therefore $\tilde{Q}_s$ is the overlapped approximate field of a unit impulse current at the origin. Using higher sampling rates will cause the rate of repetition to decrease, i.e., increase $X$. Since we are interested in solving (A7) for only a finite region in space, error due to overlap will be insignificant if the sampling rate is high enough.

However, in addition to sampling, we also truncate the frequency function. This truncation will, in general, be done by using a windowing function $\tilde{H}(\omega)$. Therefore $\tilde{G}(k)$ can be thought of as the inverse Fourier Transform of the sampled version of the function $\tilde{G}(\omega)\tilde{H}(\omega)$. $\tilde{G}(\omega)$ is the Fourier Transform of the Green's Dyadic $G$, the field due to a unit current impulse at the origin. Therefore $\tilde{G}(\omega)\tilde{H}(\omega)$ is the Fourier Transform of the field due to a current distribution $\tilde{H}$, where $\tilde{H}$ is the inverse Fourier Transform of the windowing function $\tilde{H}(\omega)$. Therefore $\tilde{G}_j(k)$ is the sampled value at $k^{th}$ point of the field due to the current distribution $\tilde{H}$. Therefore, the result of the convolution product $\tilde{G}_j * \tilde{J}$ is the (overlapped) approximate field due to the sampled values of the current $\tilde{J}$ multiplied by the current distribution $\tilde{H}$. Since a convolution product produces a finite set of values, each one corresponding to a point in space, each value can be interpreted as the approximate field at the corresponding point in space due to the current $\tilde{J}$, expanded in terms of the basis or expansion function $\tilde{H}$. Since solving (6) amounts to matching the left
hand side and the right hand side of (A7) [6], the solution of (A6) can now be seen as point matching the field E with the approximate field due to $\tilde{J}$ expanded in terms of the basis or expansion function $\tilde{H}$. If the error due to overlap is not significant, then the solution of (A6) is equivalent to point matching the known field E with the field due to $\tilde{J}$ expanded in terms of the basis or expansion function $\tilde{H}$. Therefore, the spectral iteration method is equivalent to the Method of Moments [7], if we use the inverse Fourier Transform of the windowing function $\tilde{H}(\omega)$ as expansion function and point match. The only difference is that with the Method of Moments, Gaussian elimination is normally used instead of iterative techniques.

The Discrete Convolution Method [1] on the other hand, is the iterative solution of the matrix equation formulated by the Method of Moments. Therefore the difference between the Discrete Convolution Method and the spectral iteration techniques is that with the former method, the expansion function is chosen explicitly whereas with the latter, the expansion function is chosen implicitly; i.e. the inverse Fourier Transform of the windowing function.
The computer programs given in this section are written to solve the three additional types of problems. Although they are not optimized in any sense of the word, they are written to avoid obvious wastes of computing time and memory space and is fairly efficient.

I. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE HELIX PROBLEM

Both the main program segment and the subroutine CALZ are from [8]. The slight modifications in the main program are self-explaining so no attempt will be made here to give a description. The subroutine HELIX on the other hand, is written to compute the co-ordinates of the helix points from the given radius, pitch, number of turns, and the number of points for which the co-ordinates must be found.
**THIS IS THE MAIN PROGRAM**

```fortran
C
COMPLEX Z(3600), U(400), C(400), E(3), EI(2), UV(4)
COMPLEX U, ZL(30), ZIN, YIN, V, ZC
COMPLEX TOE(1024), EX(400), CUR(1024), GUESS
COMMON /G/GUESS, CUR
COMMON XX(800), XY(800), XZ(800), TX(800), TY(800), TZ(800), AL(800)
COMMON T(1600), TP(1600), BK, RAD2(21), L(11), LR(21) /COA/C
DIMENSION PX(300), PY(300), PZ(300), LL(11), RAD(21), IFP(60)
DIMENSION LP(30)
PI=3.141592654
OPEN (UNIT=21, FILE='ABData.DAT')
1      BEAD (1, 101) NW, NP, HR, BK, IFLAG
WRITE (3, 102) NW, NP, NB, BK
IR(NR+1)=801
IR(NR+1)=801
IF(IFLAG.NE.0) GO TO 300
READ (1, 130) (PX(I), I=1, NP)
READ (1, 130) (PT(I), I=1, NP)
READ (1, 130) (PZ(I), I=1, NP)
GO TO 301
300 CONTINUE
IF(IFLAG.NE.1) GO TO 310
CALL HELIX(PX, PY, PZ, NP)
GO TO 301
310 CONTINUE
READ (1, 130) WIREL, SEGL
DO 501 I=1, NP
PX(I)=WIREL
WIREL=WIREL+SEGL
PY(I)=0.0
PZ(I)=0.0
501 CONTINUE
WRITE (3, 104) (PX(I), I=1, NP)
WRITE (3, 105) (PT(I), I=1, NP)
WRITE (3, 106) (PZ(I), I=1, NP)
READ (1, 107) (LL(I), I=1, NW)
WRITE (3, 108) (LL(I), I=1, NW)
READ (1, 107) (LR(I), I=1, NR)
WRITE (3, 109) (LR(I), I=1, NR)
READ (1, 109) (RAD(I), I=1, NR)
WRITE (3, 110) (RAD(I), I=1, NR)
301 CONTINUE
101 FORMAT(3I3, E14.7)
102 FORMAT(* ONW NP NR        BK/*3I3, E14.7)
103 FORMAT(* ONF O0.0)
104 FORMAT(* OX/(1X, 8F9.4))
105 FORMAT(* OY/(1X, 8F9.4))
106 FORMAT(* OZ/(1X, 8F9.4))
107 FORMAT(20I3)
108 FORMAT(* OLL/(1X, 10I4))
109 FORMAT(5E14.7)
103 FORMAT(* OLR/(1X, 10I4))
110 FORMAT(* ORAD/(1X, 8E0.0))
DO 46 I=1, NR
RAD2(I)=RAD(I)*RAD(I)
46 CONTINUE
J1=1
J2=2
```
J1 = J1 + 1
KK = 1

GO TO 30

29 UV(1) = UV(3)
UV(2) = UV(4)

3100 KK = 3

3200 DO 31 M = KK, 4

3300 J3 = J2 + M

3400 XDT = TX(J3) * S1 + TY(J3) * S2 - T2(J3) * ST

3500 XDP = TX(J3) * SP + TY(J3) * CP

3600 BRK = XX(J3) * BK1 + XY(J3) * BK2 + XZ(J3) * BK3

3700 UV(M) = (XDT * EI(1) + XDP * EI(2)) * (COS(BKR) + U1 * SIN(BKR))

3800 CONTINUE

3900 J3 = (J - 1) * 4

4000 J4 = J3 + 1

4100 J5 = J4 + 1

4200 J6 = J5 + 1

4300 J7 = J6 + 1

4400 U(J) = T(J4) * UV(1) + T(J5) * UV(2) + T(J6) * UV(3) + T(J7) * UV(4)

4500 J2 = J2 + 2

4600 CONTINUE

4900 WRITE(21, 113) (U(I), I = 1, N)

5000 WRITE(3, 112) 'TYPE 1 TO STOP, ELSE 0 AND RETURN'

5100 READ(1, 101) IFLAG

5200 IF (IFLAG.EQ.0) GO TO 1

5300 CLOSE(UNIT = 21)

5400 STOP

5500 END

C THIS IS SUBROUTINE #1

SUBROUTINE CALZ(N, N1, Z)

COMPLEX Z(3600), PSI(3200), U, U1, U2, U3, U4, U5, U6

COMMON XX(800), XY(800), XZ(800), TX(800), TY(800), TZ(800), AL(800)

COMMON T(1600), TP(1600), BK, RAD2(21), L(11), LR(21)

DIMENSION DC(3200)

U = (0., 1.)

PI = 3.141593

ETA = 376.7307

C1 = .125/PI

C2 = .25/PI

J1 = 1

J2 = -2

DO 1 J = 1, N

1 IF(L(J1) - J) 3, 4, 3

4 J2 = J2 + 2

3 J3 = (J - 1) * 4

5 J4 = J3 + 1

6 J5 = J4 + 1

7 J6 = J5 + 1

8 J7 = J6 + 1

9 K4 = J2 + 1

10 K5 = K4 + 1

11 K6 = K5 + 1

12 K7 = K6 + 1

13 S1 = AL(K4) + AL(K5)

C THIS IS SUBROUTINE #1

SUBROUTINE CALZ(N, N1, Z)

COMPLEX Z(3600), PSI(3200), U, U1, U2, U3, U4, U5, U6

COMMON XX(800), XY(800), XZ(800), TX(800), TY(800), TZ(800), AL(800)

COMMON T(1600), TP(1600), BK, RAD2(21), L(11), LR(21)

DIMENSION DC(3200)

U = (0., 1.)

PI = 3.141593

ETA = 376.7307

C1 = .125/PI

C2 = .25/PI

J1 = 1

J2 = -2

DO 1 J = 1, N

1 IF(L(J1) - J) 3, 4, 3

4 J2 = J2 + 2

3 J3 = (J - 1) * 4

5 J4 = J3 + 1

6 J5 = J4 + 1

7 J6 = J5 + 1

8 J7 = J6 + 1

9 K4 = J2 + 1

10 K5 = K4 + 1

11 K6 = K5 + 1

12 K7 = K6 + 1

13 S1 = AL(K4) + AL(K5)
S2 = AL(K6) + AL(K7)
T(J4) = AL(K4) * 5 * AL(K4) / S1
T(J5) = AL(K5) * (AL(K4) + 5 * AL(K5)) / S1
T(J6) = AL(K6) * (AL(K7) + 5 * AL(K6)) / S2
T(J7) = AL(K7) * 5 * AL(K7) / S2
TP(J4) = AL(K4) / S1
TP(J5) = AL(K5) / S1
TP(J6) = -AL(K6) / S2
TP(J7) = -AL(K7) / S2
J2 = J2 + 2
CONTINUE
n3 = 0 * BK * ETA
U4 = -0 / BK * ETA
BK2 = BK * BK / 2.
BK3 = BK2 * BK / 3.
N9 = 0
N2 = 1
N0 = 1
N3 = -2
DO 10 NS = 1, N
IP(L(N2) - NS) 12, 11, 12
10
KK = 1
11
N3 = N3 + 2
N2 = N2 + 1
GO TO 13
13
CONTINUE
12
K = KK, 4
14
MF = 1, N1
N4 = NF + N1
N5 = NA * N1
N6 = N5 + N1
DC(NF) = DC(N5)
DC(N4) = DC(N6)
PSI(NF) = PSI(N5)
PSI(NA) = PSI(N6)
CONTINUE
15
DO 15 K = KK, 4
N7 = N3 * K
K1 = (K - 1) * N1
16
DO 16 NF = 1, N1
IF(NF - LR(N0)) 5, 6, 5
5
AA = RAD2(N0)
N0 = N0 + 1
6
N8 = NF + K1
S1 = XX(N7) - XX(NF)
S2 = XY(N7) - XY(NF)
S3 = XZ(N7) - XZ(NF)
R2 = S1 * S1 + S2 * S2 + S3 * S3 + AA
R = SORT(R2)
RT = ABS(S1 * TX(N7) + S2 * TY(N7) + S3 * TZ(N7))
RT2 = RT * RT
R = (R2 - RT2)
ALP = 5 * AL(N7)
AR = ALP / R
S1 = BK * R
U2 = COS(S1) - U * SIN(S1)
IP(AR - 1) 22, 22, 21
CONTINUE
21
U2 = U2 * C1 / ALP
S1 = BT-ALP
S2 = ST + ALP
S3 = SQRT(S1 * SH - RH)
S4 = SQRT(S2 * S2 + BH)

AI1 = ALOG((S2 + S4) * (-S1 + S3) / RH)
GO TO 20

AI3 = (S2 * S4 - S1 * S3 + RH * AI1) / 2.
AI4 = AI2 * (RH + ALP * ALP / 3. * RT2)
S3 = AI1 * R

S1 = AI1 - BK2 * (AI3 - R * (2. * AI2 - S3))
S2 = - BK * (AI2 - S3) - BK3 * (AM - 3. * AT3 * B * I12 * (3. * AI2 - S3))

AI2 = ALN7
AI3 = (S2 * S' » - S1 * S3 * aH * A11) / 2.
AIU = AI2 * (RH + ALP * ALP / 3. * HSRT2)
S3 = AI1 * R
S1 = AI1 - BK2 * (AI3 - H * (2. * AI2 - S3))

GO TO 28

U2 = U2 * C2 / R
BA = BK * ALP
BA2 = BA * BA
AR2 = AR * AR
AR3 = AR2 * AR
ZB = ZRT / R
ZB2 = ZB * ZB
ZB3 = ZB2 * ZB
ZB4 = ZB3 * ZB
A0 = 1. + AR * A1
A4 = ZB4 / 120.
S1 = AO + BA2 * (A2 + BA2 * A4)
S2 = BA * (A1 + BA2 * A3)
PSI (N8) = 02 * (S1 + N * S2)

CONTINUE

N3 = N3 + 2
J3 = (NS - 1) * 4
4000 J7 = - 2
4100 J9 = 1
4200 DO 25 NF = 1, N
4300 J1 = (NF - 1) * 4
4400 IF (L (J9) - NF) 26, 27, 26
4500 J9 = J9 + 1
4600 J7 = J7 + 2
4700 N9 = N9 + 1
4800 U5 = 0.
4900 U6 = 0.
5000 J5 = 0
5100 DO 23 JS = 1, 4
5200 J4 = J3 + JS
5300 J8 = J5 + J7
5400 DO 24 JF = 1, 4
5500 J6 = J8 + JF
5600 J2 = J1 + JF
5700 U5 = T (J2) * T (J4) * DC (J6) * PSI (J6) + U5
5800 U6 = TP (J2) * TP (J4) * PSI (J6) + U6
5900 CONTINUE
6000 J5 = J5 + N1
6100 CONTINUE
Z(N9)=U5*U3+U6*U4
J7=J7+2
IF (N9.GE.N) RETURN
CONTINUE
RETURN
CONTINUE
END

SUBROUTINE HELIX(X, Y, Z, NP)
REAL X(1), Y(1), Z(1), RAD, PITCH, PHI, DPHI
INTEGER TURNS
READ (1, 100) RAD, PITCH, TURNS
WRITE (3, 110) RAD, PITCH, TURNS
100 FORMAT (2F0.0, 10)
110 FORMAT (1H 'RADIUS=', E14.7/1H 'PITCH=', E14.7/
$ 1H 'NO OF TURNS=', I7)
PITCH=PITCH/6.2831853
PHI=0.0
DPHI=6.2831853*TURNS/FLOAT(NP-1)
DO 10 I=1, NP
X(I)=RAD*COS(PHI)
Y(I)=RAD*SIN(PHI)
Z(I)=PITCH*PHI
PHI=PHI+DPHI
10 CONTINUE
RETURN
END
II. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEMS (SINGLE EXPANSION FUNCTION PER ANTENNA ELEMENT)

Subroutine SHAPE zerorize the corners i.e. it zerorize the phantom antenna element currents. Subroutine PATTRN computes the array factor for the given set of array currents. The main program segment solves the planar array problems using the solution routine SOLVE and other routines not listed in this section. They are listed in section III since these routines are common to both single and multiple expansion solution programs.

The flow chart for the main program segment is as follows.
Get the dimensions of the planar array

Compute the basis (L and M) for the DFT and IDFT using the routine SIZE

Compute \( \mathbf{Z} \)

Read CODE and FLAG (FLAG indicates if the same array is to be solved for more than the excitation which is current)

Depending on CODE, compute the excitation voltages \( \mathbf{V} \)

Solve the Discrete Convolution Equation \( \mathbf{Z} \ast \mathbf{J} = \mathbf{V} \)

Ask if array factors are to be computed

\( \text{YES?} \)

\( \text{N} \)

\( \text{Y} \)

Compute the array factor of the solved currents

Compute the array factor of the ideal solution

\( \text{Y} \)

more excitations to be solved?

\( \text{N} \)

STOP
PROGRAM TO COMPUTE MUTUAL, IMAGE AND TOTAL IMPEDANCES

LOGICAL FXCITE
INTEGER CODE, FLAG, FCSPTe, FLAG2
COMPLEX MUTUAL, IMAGE, ZMNG, V(1849), Z(16384), CZERO, Y(1849)
READ (1, 100) NROWS - NUMBER OF ROWS OF ANTENNA ELEMENTS
NCOLS - NUMBER OF COLUMNS
DX - SEPERATION BETWEEN ELEMENTS IN X DIRECTION
DY - DISTANCE OF ELEMENTS FROM GROUND PLANE
DZ - SEPERATION BETWEEN ELEMENTS IN Z DIRECTION

FXCITE = .TRUE.
TWOPI = 6.2831853
CZERO = (0.0, 0.0)
OPEN (UNIT=21, FILE='PATTERN.DAT')
READ (1, 100) NSIDE, CODE, DX, DY, DZ, WLENGTH, RAD
NT = NSIDE*3 - 2
NROWS = NT
NCOLS = NT
CALL SIZE (NCOLS, NROWS, L, M, N)
DC 1 I = 1, N
CONTINUE
DX = DX*TWOPI
DY = DY*TWOPI
DZ = DZ*TWOPI
WLENGTH = WLENGTH*TWOPI
RAD = RAD*TWOPI
NELEM = NBOWS*NCOLS
ZFLPM = 0.0
DY2SC = 4.0*DY*DY
RAD2 = EAD*RAD
WLBY2 = WLENGTH/2.0
POSPTR = 1
DO 20 I = 1, 2*NT - 1
XCUR = (I - 1)*DX
ZCUB = (6*NSIDE - 5 - I)*DZ
DO 10 J = 1, 2*NT - 1
RB = XCUR*XCUR*DY2SC
IMAGE = ZMNG(ZELEM, ZCUR, WLBY2, WLEY2, RR)
RR = RB - RAD2
MUTUAL = ZMNG(ZELEM, ZCUR, WLBY2, WLEY2, RR)
Z(POSPTR) = MUTUAL - IMAGE
POSPTR = POSPTR + 1
XCUR = XCUR - DX
ZCUR = ZCUR - DZ
CONTINUE
POSPTR = POSPTR + L - NT - NT + 1
READ (1, 100) CODE, FLAG
IF (CODE.EQ.1) CALL VCLTU (NCOLS, NROWS, V)
IF (CODE.EQ.2) CALL VCLTC (NCOLS, NROWS, V)
IF (CODE.EQ.3) CALL TAP (NCOLS, NROWS, DX, DZ, V)
IF (CODE.EQ.4) CALL VCLTK (NCOLS, NROWS, V)
IF (CODE.EQ.5) CALL STEER (NCOLS, NROWS, DX, DZ, V)
IF (CODE.EQ.6) CALL PHASE (NCOLS, NROWS, V)
CALL SOLVE (Z, V, Y, NCOLS, NROWS, NELEM, L, M, N, FXCITE)
FXCITE = .FALSE.
WRITE (3, 300)
IF (FLAG2 .NE. 0) GO TO 25
CALL PATTRN(NCOLS,NROWS,NELEM,Y,DX,DZ)
CALL SHAPE(V,NSIDE,NT)
CALL PATTRN(NCOLS,NROWS,NELEM,V,DX,DZ)

50  CONTINUE
IF(FLAG.EQ.0) GO TO 30
STOP

100 FORMAT(2I0,5F0.0,10)
200 FORMAT(1H,'4E12.4')
300 FORMAT(1H,'TYPE 0 TO PRINT ARRAY FACTOR, ELSE 1')

END

C
C ROUTINE TO COMPUTE THE FAR FIELD PATTERN
SUBROUTINE PATTRN(I0,MO,NO,Y,DX,DZ)
INTEGER IC,i0,NC,FTE
REAL DX,DZ,ALPHA,CCSTHE,MAGE,DANG,T1,T2,T3,DEL
COMPLEX Y(1),AF,XPHASE,ZPHASE,DXPH,DZPH,CMPLX
REAL PAT(361)

WRITE(3,1000)
READ(1,1100) NPTS,DANG
AIPHA=0.0

DO 300 II=1,NPTS
COSTHE=COS(ALPHA*.17453293E-01)
T3=DX*COSTHE
T1= COS(T3)
T2= SIN(T3)
DXPH=CMPLX(T1,T2)
DZPH=DXPH
AF=(0.0,0.0)
ZPHASE=(1.0,0.0)

DO 200 II=1,MO
T=(I-1)*LO
XPHASE=(1.0,0.0)
DO 10 J=1,LO
AF=AF+Y(T)*XPHASE*ZPHASE
XPHASE=XPHASE*DXPH
100 CONTINUE
ZPHASE=ZPHASE*DZPH
200 CONTINUE
MAGE=ABS(AF)
PAT(II)=MAGE
WRITE(3,1200) ALPHA,MAGE,AF
AIPHA=AIPHA+DANG

300 CONTINUE
WRITE(21,1300) (PAT(II),II=1,NPTS)
1000 FORMAT(1H,'NPTS AND ANGLE INCREMENT?')
1100 FORMAT(1H,'4E12.4')
1200 FORMAT(8E11.4)
1300 FORMAT(1H,'4E12.4')
1400 RETURN

C
C ROUTINE TO ZEROIZE CORNERS
SUBROUTINE SHAPE(Y,NS,LO)
COMPLEX Y(1),CZERO
CZERO=(0.0,0.0)
DO 30 I=1,NS-1
IPI= (I-1)*LO
DO 10 J=1,NS-I
Y(IPI*J)=CZERO
100 CONTINUE

11200 C
11300 C
11400 C
11500 C
11600 C
11700 C
11800 C
11900 C
12000 C
DO 20 J = 2*NS+I-1, 3*NS-2
Y(IPTR+J) = CZERO
CONTINUE
20  CONTINUE
DO 60 I = 2*NS, 3*NS-2
IPTR = (I-1)*LO
DO 40 J = 1, I-2*NS+1
Y(IPTR+J) = CZERO
CONTINUE
40  CONTINUE
DO 50 J = 5*NS-I-2, 3*NS-2
Y(IPTR+J) = CZERO
CONTINUE
50  CONTINUE
CONTINUE
CONTINUE
RETURN
END
SUBROUTINE SOLVE(A,B,LO,MO,NO,LM,MM,L,M,N,FXCITE)
ROUTINE TO SOLVE THE MATRIX EQUATION A X = B

LOGICAL FXCITE

COMPLEX CTEMP,CZERO,A(1),X(4096),V(4096),B(1),Y(7396)
INTEGER FLAG1,FLAG2,COUNT
REAL CHNAV,G,CHMAX,CHANGE

CZERO=(0.0,0.0)

CZEROIZE Y AND V (EXPENDED B)

DO 10 I=1,N
  V(I)=CZERO
  CONTINUE

DO 20 I=1,NO
  Y(I)=CZERO
  CONTINUE

IF(.NOT.FXCITE) GO TO 65

DO 40 J=1,MO
  JPTR=(T-1)*L
  JPTR=JPTR+LO+LO-1
  A(JPTR)=A(J)
  JPTR=JPTR-1
  CONTINUE

DO 60 I=1,H0-1
  IPTR=(I-1)*L
  JPTR= (HO+MO-I-1)*L
  DO 50 J=1,L0*L0-1
    A (JPTR+J) =A (IPTR+J)
    CONTINUE

CALL TWODF(A,N,M,I,MH,LM)

65      CONTINUE

CALL ITER(A,B,X,V,Y,LO,M0,NO,LM,MM,L,M,N)

RETURN

END


INTEGER COUNT,FLAG1,FLAG2
COMPLEX CZERO,CTEMP,A(1),B(1),X(1),V(1),Y(1)
REAL CHNAV,G,CHMAX,CHANGE

COUNT=0

JPTR=1

DO 90 I=1,MO
  IPTR=L*(MO-2+I)+LO
  DO 80 J=1,LO
    V(IPTR)=B(JPTR)
    JPTR=JPTR+1
  CONTINUE

FIND V TRANSFORMED AND COMPUTE X TRANSFORMED

CALL TWODP(V,N,M,L,MM,L)

DO 100 I=1,H
  X(I)=V(IPTR)/A(I)
  IPTR=IPTR+1
  CONTINUE

GET X FROM X TRANSFORMED

CALL ITWODF(X,N,M,L,MM,L)

TRUNCATE X AND SAVE X AFTER COMPUTING THE CONVERGENCE CRITERION
CHNAVG=0.0
CHNMAX=0.0
DO 120 I=1,N
IPTL=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*L+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(NO)
CHNMAX=CHNMAX*100.0
WRITE(3,1200) CHNAVG,CHNMAX,COUNT
FIND THE TRANSFORM OF TRUNCATED X
CALL TWODF(X,N,H,L,HH,LM)
COMPUTE V TRANSFORMED
DO 130 1=1,N
V(I)=A(I)*X(I)
130 CONTINUE
GET V FROM V TRANSFORMED
CALL ITWODF(V,N,M,L,MM,LM)
COMPUTE THE ERROR CRITERION
CHNAVG=0.0
CHNMAX=0.0
JPTR=1
DO 150 I=1,NO
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
150 CONTINUE
ASK WHETHER OR NOT TO STOP AFTER REPORTING % FIELD ERROR
CHNAVG=(CHNAVG*100.0)/FLOAT(NO)
CHNMAX=CHNMAX*100.0
WRITE(3,1300) CHNMAX,CHNAVG
READ(1,1100) FLAG1
IF(FLAG1.EQ.0) GO TO 70
WRITE(3,1400) (V(I),I=1,N)
RETURN
1000 FORMAT(10EO.O)
1100 FORMAT(110)
1200 FORMAT(1H,'AVG CURRENT CHANGE=','E14.7,'%,')
1300 FORMAT(1H,'MAX CURRENT CHANGE=','E14.7,'%,')
1400 FORMAT(1H,'AFTER',I4,' ITERATIONS')
$ 1H 'AVG FIELD ERROR = ','E15.7,' %'
12300 $ 1H 'CONTINUE ITERATIONS? 0 FOR YES, 1 FOR NO, AND RETURN'/)
12400 1400 FORMAT(1H 'CURRENTS'//(1H ,10E11.4))
12500 1500 FORMAT(1H 'PRINT FIELDS? 0 FOR YES, 1 FOR NO, THEN RETURN'/)
12600 1600 FORMAT(1H 'RESULTANT FIELDS'//(1H ,10E11.4))
12700 END
III. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEMS (THREE EXPANSION FUNCTIONS PER ANTENNA ELEMENT)

The difference between the main program segment for three expansion functions per antenna element and the main program segment for one expansion function per antenna element is that there are five distinct mutual impedance vectors $\vec{Z}$ to be computed for the three expansion functions per antenna element solution. Therefore the computation of each $\vec{Z}$ is done by the separate routine FILLZ. Similarly, MSOLVE and SOLVE routines differ mainly in that there are three distinct vectors each for the generalized voltage $\vec{V}$ and $\vec{J}$ to be computed by MSOLVE. SOLVE on the other hand, computes only one vector each of $\vec{V}$ and $\vec{J}$.

Routine ZMNG computes the mutual impedance between two parallel segments of thin wire dipoles of same length which may or may not be offset from each other along one or more axes. SICI, VOLTU, VOLTC TAP, VOLTK, and PHASE are all from [9] and hence no description of them will be given here. FFT, IFFT, TWODF, and ITWODF are fast fourier transform and inverse transform routines for one and two dimensional discrete fourier transforms, respectively. PATT3E is the routine to compute the array factor from the current distribution solutions obtained by using three expansion functions per antenna element.
The last main program segment computes the array factors from the current distribution solutions obtained by using three expansion functions per antenna element and from the ideal solutions which ignore the mutual coupling between antenna elements.
PROGRAM FOR TRIANGULAR PATTERN WITH THREE EXPANSIONS PER ELEMENT

INTEGER CODE,FLAG,NRCWS,NCOLS,NELEM
REAL TWOPI
COMPLEX V2(4096),ZA(4096),ZB(4096),ZC(4096),ZD(4096),ZE(4096)
COMPLEX XI(4096),X2(4096),X3(4096)

FXCITE=.TRUE.
TWOPI=6.2831853
CZERO=(0.0,0.0)
OEEN(UNIT=21,FILE='MXCUR.CAT')
READ(1,100) NSIDE,DX,DY,DZ,WLNGTH,RAD
NROWS=NSIDE*3-2
NCOLS=NROWS
CALL SIZE(NCOLS,NROHS,LM,MH,L,H,N)
DX=DX*TWOPI
DY=DY*TWOPI
DZ=DZ*TWOPI
WLBY2=WLNGTH*TWOPI/4.0
ZLEMI=WLBY2
RAD=RAD*TWOPI
NELEM=NEOWS*NCCLS
DY2SQ=4.0*DY*EY
RAD2=BAD*SAD
ZCFF=0.0
CALL F1LLZ(ZA,ZOFF,DX,DZ,DY2SQ,WLBY2,RAD2,NROHS,L,N)
ZOFF=-ZELEM
CALL FILLZ(ZB,ZCFF,DX,DZ,DY2SQ,HLEY2,FAn2,NHCiS,L,S)
ZOFF=-ZELEK-ZELEn
CALL FILLZ(ZC,ZOFF,DX,DZ,DY2SQ,WLBY2,BA')2,NPCWS,L, 8)
CALL   FILLZ(ZD,ZELEK,DX,DZ,DY2SQ,HLEY2,FAD2,NBOHS,L,N)
ZOFF=ZELEM+ZELEM
CALL   FILIZ(ZE,ZCFF,EX,DZ,CY2SC,iLBY2,BA')2,NBCWS,L,N)
10 READ(1,200) CODE,FLAG
IF{CODE.EQ.1) CALL VCLTU(NC0LS,NS0WS,V2)
IF(C0DE.EQ,2)   CALL   VCITC(NCCLS,NBOiS,
IF{CODE.EQ.3)
IF(C0DE.EQ.4)
IF(C0DE.EQ.5)
IF(CODE.EQ.6)   CALL   PHASE(NCOLS,NEOWS,V2)
CALL HSOLVE(ZA,ZB,ZC,ZD,ZE,V2,X1,X2,X3,NCOLS,NRO»S,
FXCITE=.,FALSE.
IF(FLAG.EQ.0) GO TO 10
STOP
FORMAT(IO,5FO.O,IC)
200 FORMAT (210)
END

ROUTINE TO FILL THE Z MATRIX FOR TRIANGULAR ARRAYS

SUBROUTINE FILLZ(Z,ZOFF,DX,DZ,DY2SQ,WLBY2,RAD2,NROHS,L,N)
INTEGER POSPTR,NRCWS,LM
REAL DX,DZ,DY2SQ,ZELEM,XCUR,ZCUR,REW,WLBY2
COMPLEX Z(1),MUTUAL,IMAGE,CZERO,ZMNG
ZC=0.0
PCSPTR=1
DO 10 I=1,N
Z(I)=CZERO
CONTINUE
NTOTAL=NRWS+NRCWS-1
DO 30 I=1,NTOTAL
XCUR=(I-1)*DX
ZCUR=(NTOTAL-I)*DZ+ZCFF
DO 20 J=1,NTOTAL
RR=XCUR*XCUR+DY2SC
IMAGE=ZMNG(ZELEM,ZCUR,WLEY2,RR)
RR=RR-DY2SQ
IF(RR.EQ.0.0)RR=RR+RAD2
MUTUAL=ZMNG(ZELEM,ZCUR,WLEY2,RR)
Z(POSPTR)=MUTUAL-IMAGE
PCSPTR=PCSPTR+1
XCUR=XCUR-DX
ZCUR=ZCUR-DZ
20 CONTINUE
POSPTS=POSPTR*-L-NTOTAL
30 CONTINUE
RETURN
END

C ROUTINE TO SOLVE THE CONVOLUTION EQUATIONS

SUBROUTINE MSOLVE(ZA,ZB,ZC,ZD,ZE,B,X1,X2,X3,LO,NO,LM,MM,L,M,N,FXCITE)

INTEGER COUNT
LOGICAL FXCITE
COMPLEX CTEXP1,CTEMP2,CTEMP3,ZA(1),ZB(1),ZC(1),ZD(1),ZE(1),B(1),X1(1),X2(1),X3(1)
V1(1),V2(1),V3(1)
T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,Y1(484),Y2(484),Y3(484)
CZERO=(0.0,0.0)
DO 10 I=1,NO
Y1(I)=CZERO
Y2(I)=CZERO
Y3(I)=CZERO
10 CONTINUE
NS=(IC+2)/3
EEACT=3.0*FLOAT(NO-2*(NS-1)*NS)
DO 20 I=1,N
V1(I)=CZERO
V2(I)=CZERO
V3(I)=CZERO
20 CONTINUE
IF(.NOT.FXCITE)GO TO 30
CALL TDODF(ZA,N,M,L,MM,LM)
CALL TDODF(ZB,N,M,L,MM,LM)
CALL TDODF(ZC,N,M,L,MM,LM)
CALL TDODF(ZD,N,M,L,MM,LM)
CALL TDODF(ZE,N,M,L,MM,LM)
30 ICENTB=L*(NO-1)+LC-1
40 CONTINUE
IF(.NOT.FXCITE)GO TO 30
CALL TDODF(ZA,N,M,L,MM,LM)
CALL TDODF(ZB,N,M,L,MM,LM)
CALL TDODF(ZC,N,M,L,MM,LM)
CALL TDODF(ZD,N,M,L,MM,LM)
CALL TDODF(ZE,N,M,L,MM,LM)
1040 30 ICENTB=L*(NO-1)+LC-1
1050 COUT=1
1060 40 CONTINUE
1080 DO 60 I=1,NS-1
1090 IPTR=(I-1)*L+ICENTR
1100 JPTR=(I-1)*LO
1110 DO 50 J=NS-I+1,2*NS+I-2
1120 V1(IPTR+J)=CZERO
1130 V2(IPTR+J)=B(JPTR+J)
1140 V3(IPTR+J)=CZERO
1150 50 CONTINUE
11600  60  CONTINUE
11700  DO  80  I=NS,2*NS-1
11800  IPTR= (I-1) *L+ICENTR
11900  JPTR= (I-1) *LO
12000  DO  70  J=1,3*NS-2
12100  V1(IPTR+J)=CZERO
12200  V2(IPTR+J)=B(IPTR+J)
12300  V3(IPTR+J)=CZERO
12400  70  CONTINUE
12500  80  CONTINUE
12600  DC  100  I=2*NS,3*NS-2
12700  IPTR= (I-1) *L+ICENTR
12800  JPTR= (I-1) *LO
12900  DO  90  J=I*2-2*NS,5*NS-I-3
13000  V1(IPTR+J)=CZERO
13100  V2(IPTR+J)=B(IPTR+J)
13200  V3(IPTR+J)=CZERO
13300  90  CONTINUE
13400  100  CONTINUE
13450  105  CONTINUE
13500  C  FIND V1, V2, V3 TRANSFORMED AND COMPUTE X1, X2, X3
13600  CALL TWODF(V1,N,M,I,Mf1,LM)
13700  CALL TWODF(V2,N,M,I,Mf1,LM)
13800  CALL TWODF(V3,N,M,I,f'H,LM)
13900  DC  110  I=1,N
14000  T1=ZD(I)/ZA(I)
14100  T2=ZE(I)/ZA(I)
14200  T3=ZA(I)-T1*ZB(I)
14300  T4=ZB(I)-T1*ZC(I)
14400  T5=ZA(I)-T2*ZC(I)
14500  T6=ZD(I)-T2*ZB(I)
14600  T7=V2(I)-V1(I)*T1
14700  T8=V3(I)-V1(I)*T2
14800  T9=T5-T4*T6/T3
14900  T10=T8-T7*T6/T3
15000  X3(I)=T10/T9
15100  X2(I)=(T7-T4*X3(I))/T3
15200  X1(I)=(V1(I)-ZB(I)*X2(I)-ZC(I)*X3(I))/ZA(I)
15300  110  CONTINUE
15400  CALL ITWODF(X1,N,M,L,MM,LM)
15500  CALL ITWODF(X2,N,M,L,MM,LM)
15600  CALL ITWODF(X3,N,M,L,MM,LM)
15700  DC  140  I=1,NS-1
15800  IPTR= (I-1) *L
15900  DC  120  J=1,NS-I
16000  X1(IPTR+J)=CZERO
16100  X2(IPTR+J)=CZERC
16200  X3(IPTR+J)=CZERO
16300  120  CONTINUE
16400  DO  130  J=2*NS+I-1,3*NS-2
16500  X1(IPTR+J)=CZERC
16600  X2(IPTR+J)=CZERC
16700  X3(IPTR+J)=CZERO
16800  130  CONTINUE
16900  140  CONTINUE
17000  DO  170  I=2*NS,3*NS-2
17100  IPTR= (I-1) *L
17200  DO  150  J=1,I-2*NS+1
17300  X1(IPTR+J)=CZERO
17400  X2(IPTR+J)=CZERC
17500 X3(IPTR+J)=CZERC
17600 CONTINUE
17700 DO 160 J=5*NS-1,3*NS-2
17800 X1(IPTR+J)=CZERC
17900 X2(IPTR+J)=CZERC
18000 X3(IPTR+J)=CZERC
18100 CONTINUE
18200 CONTINUE
18300 C
18400 C COMPUTE THE CRITERIANS AND REPORT
18500 CHNAVG=0.0
18600 CHNMAX=0.0
18700 DC 190 I=1,N
18800 IPTR=I/L
18900 JPTR=I-IPTR*L
19000 IF (JPTR.LE.0. AND. JPTR.NE.0. AND. IPTR.LT.MC) GO TO 180
19100 X1(I)=CZERO
19200 X2(I)=CZERO
19300 X3(I)=CZERO
19400 GC TO 190
19500 CONTINUE
19600 IPTR=IPTR*LO+JPTR
19700 CTEMP1 =X1(I)
19800 CTEMP2 =X2(I)
19900 CTEMP3 =X3(I)
20000 IF (CTEMP1.EQ.CZERO) GO TO 190
20100 CHNGE1=CAES(CTEMP1-Y1(IPTR))/CAES(CTEMP1)
20200 CHNGE2=CAES(CTEMP2-Y2(IPTR))/CAES(CTEMP2)
20300 CHNGE3=CAES(CTEMP3-Y3(IPTR))/CAES(CTEMP3)
20400 Y1(IPTR)=CTEMP1
20500 Y2(IPTR)=CTEMP2
20600 Y3(IPTR)=CTEMP3
20700 CHNAVG=CHNAVG+CHNGE1+CHNGE2+CHNGE3
20800 IF (CHNGE1.GT.CHNMAX) CHNMAX=CHNGE1
20900 IF (CHNGE2.GT.CHNMAX) CHNMAX=CHNGE2
21000 IF (CHNGE3.GT.CHNMAX) CHNMAX=CHNGE3
21100 CONTINUE
21200 CHNAVG=(CHNAVG*100.0)/EFACT
21300 CHNMAX=(CHNMAX*100.0)
21400 WRITE (3,1200) CHNAVG,CHNMAX,COUNT
21500 C
21600 C GET THE TRANSFORM OF THE CURRENTS
21700 CALL TWODF (X1,N,M,L,MM,LM)
21800 CALL TWODF (X2,N,M,L,MM,LM)
21900 CALL TWODF (X3,N,M,L,MM,LM)
22000 DO 200 I=1,N
22100 V1(I)=ZA(I)*X1(I)+ZB(I)*X2(I)+ZC(I)*X3(I)
22200 V2(I)=ZD(I)*X1(I)+ZA(I)*X2(I)+ZB(I)*X3(I)
22300 V3(I)=ZE(I)*X1(I)+ZD(I)*X2(I)+ZA(I)*X3(I)
22400 CONTINUE
22500 C
22600 C GET THE FIELD AND COMPUTE % ERRORS
22700 CALL ITWODF (V1,N,M,L,MM,LM)
22800 CALL ITWODF (V2,N,M,L,MM,LM)
22900 CALL ITWODF (V3,N,M,L,MM,LM)
23000 CHNAVG=0.0
23100 CHNMAX=0.0
23200 DO 220 I=1,NS-1
23300 IPTR=(I-1)*L+ICENTF
23400 JPTR=(I-1)*LO
DO 210 J = NS-I+1, 2*NS+I-2
CTEMP1=E(JPTR+J)
CHANGE=CABS(CTEMP1-V2(IPTR+J))/CABS(CTEMP1)
IF (CHANGE.GT.CHNMAX) CHNMAX=CHANGE
CHNAVG=CHNAVG+CHANGE
V2(IPTR+J)=CTEMP1
V1(IPTR+J)=CZERO
V3(IPTR+J)=CZERO
210 CONTINUE
DO 220 I=NS, NS*2-1
IPTe= (.I-1)*L+ICENTR
JPTR= (I-1)*LO
DO 230 J=1, 3*NS-2
C'HEP1=B{JPTa+J)
CHANGE=CABS(CTEMP1-V2(IPTR+J))/CABS(CTEHP1)
IF (CHANGE.GT.CHNHAX) CHNHAX=CHANGE
CHNAVG=CHNAVG+CHANGE
V1(IPTR+J)=CZEBC
V2(IPTR+J)=CTEME1
V3(IPTR+J)=CZEBC
230 CONTINUE
220 CONTINUE
DO 240 I=2*NS, 3*NS-2
IPTB=(I-1)*L+ICENTR
JFB=(I-1)*LO
DO 250 J=I+2-2*NS, 5*NS-I-3
CTEMP1=E{JPTR*J)
CHANGE=CABS(CTEHP1-V2(IPTR+J))/CABS(CTEHP1)
IF (CHANGE.GT.CHNHAX) CHNHAX=CHANGE
CHNAVG=CHNAVG+CHANGE
V2(IPTR+J)=CTEMP1
V1(IPTR+J)=CZEBC
V3(IPTR+J)=CZEBC
250 CONTINUE
240 CONTINUE
ASK WHETHER OR NOT TO STOP AFTER REPORTING % FIELD ERROR
CHNAVG=(CHNAVG*100.0)/EFACCT
CHNMAX=CHNt1AX*100.0
WRITE (3,1300) CHNAVG, CHNMAX
READ (1,1100) FLAG1
COUNT=COUNT+1
IF (FLAG1.EQ.0) GO TO 105
WRITE (3,1400) (Y1(I), I=1,NO)
WRITE (21,1700) (Y1(I), I=1,NO)
WRITE (3,1400) (Y2(I), I=1,NO)
WRITE (21,1700) (Y2(I), I=1,NO)
WRITE (3,1400) (Y3(I), I=1,NO)
WRITE (21,1700) (Y3(I), I=1,NO)
ASK IF FIELD SHOULD BE PRINTED OUT ALSO
WRITE (3,1500)
READ (1,1100) FLAG2
IF (FLAG2.NE.0) RETURN
WRITE (3,1600) (Y2(I), I=1,N)
RETURN
1000 FORMAT (10E0.0)
1100 FORMAT (4I0)
1200 FORMAT (1H 'AVG CURRENT CHANGE=', E14.7, ' ')
8700 $ 1H 'MAX CURRENT CHANGE=', E14.7, ' /
8900 $ 1H 'AFTER', I4, ' ITERATIONS' /)
29000 1300 FORMAT(1H,'AVG FIELD ERROR = ',E15.7, '/
29100  $  1H,'MAX FIELD ERROR = ',E15.7, '/
29200  1H,'CONTINUE ITERS: 0 FOR YES, 1 FOR NO, AND RETURN'/)
29300 1400 FORMAT(1H,'CONTINDENITCLES? 0 FOR YES, 1 FOR NO, THEN RETURN'/)
29400 1500 FORMAT(1H,'PRINT FIELDS? 0 FOR YES, 1 FOR NO, THEN RETURN'/)
29500 1600 FORMAT(1H,'RESULTANT FIELDS'/1H,10E11.4)
29550 1700 FORMAT(8E14.6)
29600 END
29700 C COMPUTE MUTUAL IMPEDANCE BETWEEN TWO PARALLEL SEGMENTS
29800 C OF THIN WIRE DIPOLIES OF SAME LENGTH
30000 FUNCTION ZMNG(Z1,Z2,LIBY2,RSQ)
30100 REAL LBY2
30200 COMPLEX ZMNG,CMFLX
30300 DZ=ABS(Z1-Z2)
30400 CC=2.0*COS(LBY2)
30500 CSQ=CC*CC
30600 D1=DZ
30700 D2=DZ+LIBY2
30800 D3=DZ-LIBY2
30900 D4=D2+LIBY2
31000 D5=D3-LIBY2
31100 U1=SQRT (RSQ+D1*D1)/D1
31200 U2=SQRT (RSQ+D2*D2)/D2
31300 U3=SQRT (RSQ+D3*D3)/D3
31400 U4=SQRT (RSQ+D4*D4)/D4
31500 U5=SQRT (RSQ+D5*D5)/D5
31600 V1=RSQ/U1
31700 V2=RSQ/U2
31800 V3=RSQ/U3
31900 V4=RSQ/U4
32000 V5=RSQ/U5
32100 CALL SICI(SU1,CU1,U1)
32200 CALL SICI(SU2,CU2,U2)
32300 CALL SICI(SU3,CU3,U3)
32400 CALL SICI(SU4,CU4,U4)
32500 CALL SICI(SU5,CU5,U5)
32600 CALL SICI(SV1,CV1,V1)
32700 CALL SICI(SV2,CV2,V2)
32800 CALL SICI(SV3,CV3,V3)
32900 CALL SICI(SV4,CV4,V4)
33000 CALL SICI(SV5,CV5,V5)
33100 S1=SIN(D1)
33200 S2=SIN(D2)
33300 S3=SIN(D3)
33400 S4=SIN(D4)
33500 S5=SIN(D5)
33600 C1=COS(D1)
33700 C2=COS(D2)
33800 C3=COS(D3)
33900 C4=COS(D4)
34000 C5=COS(D5)
34100 RL=(2.0+CSQ)*(C1*(CU1+CV1)+S1*(SU1-SV1))
34200 $-2.0*CC*(C2*(CU2+CV2)+S2*(SU2-SV2)+C3*(CU3+CV3)
34300 $+S3*(SU3-SV3))+C4*(CU4+CV4)+S4*(SU4-SV4)
34400 $+C5*(CU5+CV5)+S5*(SU5-SV5)
34500 AG=(2.0+CSQ)*(S1*(SU1-CV1)-C1*(SU1+SV1))
34600 $-2.0*CC*(S2*(SU2-CV2)-C2*(SU2+SV2)+S3*(CU3-CV3)
34700 $-C3*(SU3-CV3))+S4*(CU4-CV4)-C4*(SU4+SV4)
34800 $+S5*(CU5-CV5)-C5*(SU5+SV5)
SUBROUTINE SICI(SI, CI, X)
    Z = ABS(X)
    IF(Z .LT. 4.) RETURN

    SI = X * (((1.753141E-9*Y + 1.568988E-7) * Y + 1.374168E-5) * Y + 6.939889E-4)
    CI = ((5.7721568E-1 + ALOG(Z)) / Z - (((1.386985E-10*Y + 1.58496E-8) * Y
          + 1.7525752E-6) * Y + 1.185999E-4) * Y + 4.990920E-3) * Y + 1.315308E-1)

RETURN

SUBROUTINE VOLTM23(M2, M3, V)
    COMPLEX V(1), CMPLX, VALUE
    M23 = M2 * M3

    READ (5, 1) AM, PH
    FORMAT (2F0.0)

    RAC = PH * 3.14159 / 180.
    VALUE = CMPLX(AM * COS(RAD), AM * SIN(RAD))

    DO 2 I = 1, M23
        V(I) = VALUE
    CONTINUE

WRITE (5, 3) AM, PH
    FORMAT (5, 3) AM, PH

SUBROUTINE VOLTC(M2, M3, V)
    READ IN COMPLEX NUMBERS AS VOLTAGE FOR EACH DIPOLE
    COMPLEX V(1)
    M23 = M2 * M3

    READ (5, 1) (V(I), I = 1, M23)

RETURN
C SUBROUTINE STEER(M2,M3,DX,DZ,PP)
C PROGRESSIVE PHASE SHIFT ON EACH DIPOLE -
C STEERING THE MAIN BEAM IN BOTH DIRECTIONS
C
READ (5,1) RZ,EX
1 FORMAT (2F0.0)
PI2=6.2831853
THX=(RX)*PI2/360.
THZ=(RZ)*PI2/360.
L=0
XK=-DX*COS(THX)
ZK=-DZ*COS(THZ)
DO 100 I=1,M3
DO 100 J=1,n2
L=I+1
PH=FLOAT(J-1)*XK+FLOAT(I-1)*ZK
PP(L)=1.0*CMPLX(COS(PH),SIN(PH))
100 CONTINUE
WRITE(5,20)RX,RZ
20 FORMAT (//1X,'BEAM STEERING =',F10.5,' DEGREES IN PHI ANGLE',//)
&/1X,14X,'=','F10.5,' DEGREES IN THE ANGLE'//)
RETURN
END
C
C SUBROUTINE TAP(M2,M3,EX,CZ,VV)
C MAGNITUDE TAPER OF EXCITATION IN BOTH DIRECTIONS
C
PI2=6.2831853
HFX=DX*(M2-1)*0.5
HFZ=DZ*(M3-1)*0.5
L=0
WRITE(5,2)
2 FORMAT (//1X,'EXPONENTIAL TAPERED IN MAGNITUDE'//)
DO 2 I=1,M3
Z=((I-1)*DZ-HFZ)/PI2
FUNZ=EXP(-ABS(Z))
DO 2 J=1,H2
L=I+1
X=((J-1)*DX-HFX)/PI2
VV(L)=EXP(-ABS(X))*FUNZ
20 CONTINUE
RETURN
END
C
C SUBROUTINE VOLTK(M2,M3,V)
C COMPLEX V(1),VK(20),VJ(20)
READ (5,1) (VK(I),I=1,M3)
READ (5,1) (VJ(I),I=1,M2)
1 FORMAT (10F0.0)
L=0
DO 2 I=1,M3
DO 2 J=1,M2
L=I+1
V(L)=VK(I)*VJ(J)
2 CONTINUE
WRITE(5,4)
SUPROUTINE PHASE(M2,M3,PP)

; PROGRESSIVE PHASE SHIFT ON EACH DIPOLE -

COMPLEX PP(1),CMPLX

READ (5,1)RZ,RX

1 FORMAT (2F0.0)

PI2=6.2831853

THX=(RX)*PI2/360.

THZ=(RZ)*PI2/360.

DO 100 I=1,M3

I=I+1

PH=FLOAT (I-1) *THX+FLOAT (I-1) *THZ

PP (I)=1.0*CMPLX (COS (PH) ,SIN (PH))

100 CONTINUE

WRITE(5,20)RX,RZ

20 FORMAT (///1X,'PROGRESSIVE PHASE SHIFT = ',F10.5,

&DEGREES IN ROW DIRECTION',/25X,'= ',F10.5,

&DEGREES IN COLUMN DIRECTION',///)

RETURN

END

SUBROUTINE SIZE(L0,MC,LM,M,L,M,N)

L=L0*3-2

M=M0*3-2

LTEMP=L

MTEMP=M

L=0

M=0

1 L=L/2

LM=LM+1

IF (L.GT. 1) GO TO 1

2 M=M/2

MM=MM+1

IF (M.GT. 1) GO TO 2

L=2**IM

M=2**MM

IF (LTEMP.GT. LM) LM=LM+1

IF (LMTEMP.GT. L) L=L*2

IF (MTEMP.GT. MM) MM=MM+1

IF (MTTEMP.GT. M) M=M*2

N=M*L

RETURN

END

SUBROUTINE FFT(X,M,START,STEP)

COMPLEX X (16384),U,K,T

INTEGER START,STEP,SCIfF

N=2**H

SCIfF=STEP-START

NV2 =N/2*STEP

NH1 = (N-2)*STEP*START

N=(N-1)*STEP+START

J=START

DO 8 I=START,NM1,STEP

IF (I.GE.I) GC TO 5
69

53300  \( T = X(J) \)
53400  \( X(J) = X(I) \)
53500  \( X(I) = T \)
53600  K = NV2
53700  IF(K-SDIFF.GE.J) GC TO 7
53800  J = J - K
53900  K = K/2
54000  GC TO 6
54100  J = J + K
54200  CONTINUE
54300  PI = 3.14159265358979
54400  DO 20 L=1,M
54500  LE = 2**L
54600  LSTEP = LE/2
54700  ANGLE = PI/FLOAT(L*E1)
54900  W = COMPLEX(COS(ANGLE),-SIN(ANGLE))
55100  IF1 = LSTEP+START-STEP
55200  LE = LE*STEP
55300  DO 20 J=START,LE1,STEP
55400  DC 10 I=J,N,LE
55500  IF = I+LSTEP
55600  T = X(IP)*U
55700  X(IP) = X(I) - T
55800  X(I) = X(I) + T
55900  CONTINUE
56000  U = U*W
56100  CONTINUE
56200  RETURN
56300  END
56400  C
56500  SUBROUTINE TWODF(X,N,L,U,W,T)
56600  COMPLEX X(16384)
56700  INTEGER START,STEP
56800  START = 1
56900  STEP = 1
57000  DO 8 I=1,M
57100  CALL FFT(X,LM,START,STEP)
57200  START = START + 1
57300  CONTINUE
57400  STEP = L
57500  DO 20 I=1,L
57600  START = I
57700  CALL IFT(X,MM,START,STEP)
57800  CONTINUE
57900  RETURN
58000  END
58100  SUBROUTINE IFFT(X,M,START,STEP)
58200  COMPLEX X(16384),U,W,T
58300  INTEGER START,STEP,SLIFF
58400  N = 2**M
58500  SLIFF = STEP-START
58600  NV2 = N/2*STEP
58700  NM1 = (N-2)*STEP+START
58800  NEXP = (N-1)*STEP+START
58900  J = START
59000  DO 8 I=START,NM1,STEP
59100  IF(I.GE.J) GC TO 5
59200  T = X(J)
X(J) = X(I)
X(I) = T

5 K = NV2
6 IF(K-SDIFF.GE.J) GC TO 7
J = J-K
K = K/2
GC 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

SUBROUTINE ITWOE(X,N,M,L,MM,LM)
COMPLEX X(16384)
INTEGER START,STEP
START=1
STEP =1
DO 10 I=1,M
CALL IFFT(X,L,M,START,STEP)
START=START+L
10 CONTINUE
STEP =L
DO 20 I=1,L
START=I
CALL IFFT(X,MM,START,STEP)
20 CONTINUE
FN = FLOAT(N)
DO 30 I=1,N
X(I) = X(I)/FN
30 CONTINUE
RETURN
END
SUBROUTINE PATTERN (LO, MO, NO, Y, DX, DZ)
INTEGER LO, MO, NO, PTR
REAL DX, DZ, ALPHA, COSTHE, MAGE, DANG, T1, T2, T3, T4, DEL
COMPLEX Y(1), AF, XPHASE, ZPHASE, DXPH, DZPH, CMPLX
REAL PAT(361)
WRITE((3,1000))
READ(1,1100) NPTS, DANG, IFLAG
ALPHA=0.0
DO 300 IT=1,flPTS
COSTHE=COS(ALPHA*.17453293E-01)
T3=DX*COSTHE
T1=COS(T3)
T2=SIN(T3)
T4=-T2
DXPH=CMPLX(T1, T2)
DZPH=DXPH
AF=(0.0,0.0)
ZPHASE=(1.0,0.0)
IF (IFLAG. NE. 0) DZPH=CMPLX(T1, T4)
DO 200 I=1,MO
PTR=(I-1)*LO
XPHASE=(1.0, 0.0)
DO 100 J=1,LO
AF=AF+Y(PTR+J)*XPHASE*ZPHASE
100 CONTINUE
ZPHASE=ZPHASE*DZPH
200 CONTINUE
HAGE=CA(E)(AF)
PAT(II)=HAGE
WRITE(3,1200) ALPHA, MAGE, AF
ALPHA = ALPHA*-DANG
300 CONTINUE
WRITE(21,1300) (PAT(II), II=1,NPTS)
RETURN
END

C ROUTINE TO ZEROIZE CORNERS
SUBROUTINE SHAPE(Y,NS,LO)
COMPLEX Y(1), CZERO
CZERO=(0.0,0.0)
DO 30 I=1,NS-1
IPTR=(I-1)*LO
DO 10 J=1,LO
Y(IPTR+J)=CZERO
10 CONTINUE
DO 20 J=2*NS*I-1,3*NS-2
Y(IPTR+J)=CZERO
20 CONTINUE
DO 60 I=2*NS,3*NS-2
IPTR=(I-1)*LO
Y(IPTR+J)=CZERO
60 CONTINUE
DO 60 I=2*NS,3*NS-2
IPTR=(I-1)*LO
Y(IPTR+J)=CZERO
60 CONTINUE
SUBROUTINE PATT3E (LO, MO, NO, Y, DX, DZ, WLBY2)

INTEGER LO, MO, NO, PTR, CPTR
REAL DX, DZ, ALPHA, COSTHE, MAGE, DANG, T1, T2, T3, T4, DEL

COMPLEX Y(1), AF1, AF2, AP3, XPHASE, ZPHASE, DXPH, DZPH, COMPLX, AP

REAL PAT (361)

WRITE (3, 1000)
READ (1, 1100) NPTS, DANG, IFLAG

ALPHA = 0.0

DO 300 I = 1, NPTS
COSTHE = COS(ALPHA * 1.7453293E-01)

T1 = DX * COSTHE
T2 = SIN(T1)
T3 = -T2

DXPH = CMPLX(T1, T2)
DZPH = DXPH

AF1 = (0.0, 0.0)
AF2 = (0.0, 0.0)
AF3 = (0.0, 0.0)
ZPHASE = (1.0, 0.0)

IF (IFLAG .NE. 0) DZPH = CMPLX(T1, T2)
IF (COSTHE .GE. 0.99999) GO TO 250

AF = AF1 * FAC1 + AF2 + AF3 * FAC3

IF (COSTHE .LE. 0.99999) GO TO 250

AF = AF * (T1 - COS(WLBY2 / 2.0)) / (COS(WLBY2 * COSTHE) - COS(WLBY2))

CONTINUE

BAGE = CABS(AF)

PAT (II) = BAGE
WRITE(3,1200) ALPHA, MAGE, AF
11300    ALPHA=ALPHA+DANG
11400    300 CONTINUE
11500    WRITE(21,1300) (PAT(II),II=1,NPTS)
11600    1000 FORMAT(1H, 'NPTS AND ANGLE INCREMENT?')
11700    1100 FORMAT(10, E0.0, I0)
11800    1200 FORMAT(1H, F8.1, 3E12.4)
11900    1300 FORMAT(0E11.4)
12000    RETURN
12100    END
12200    C
12300    C MAIN PROGRAM
12400    COMPLEX V(12300)
12450    INTEGER CODE, FLAG, LO, MO, NO, SKIP
12500    OPEN(UNIT=21, FILE='PATTRN.DAT')
12600    OPEN(UNIT=22, FILE='MXCUR.DAT')
12605    READ(1,200) LO, NO, NO, DX, DZ, WLBY2, SKIP
12610    IF(SKIP.LE.0) GO TO 20
12620    READ(22,300) (V(I), I=1,NO)
12630    READ(22,300) (V(I), I=1,NO)
12640    READ(22,300) (V(I), I=1,NO)
12650    SKIP=SKIP-1
12660    GC TO 10
12670    20 CONTINUE
12720    TWOPI=6.2831853
12760    DX=DX*TWOPI
12800    DZ=DZ*TWOPI
12820    WLBY2=WLBY2*TWOPI
12900    READ(22,300) (V(I), I=1,NO)
12930    READ(22,300) (V(I), I=NO+1,NO+NO)
12960    READ(22,300) (V(I), I=NO+NO+1,NO+NO+NO)
13000    CALL PATT3E(LO, NO, NO, V, DX, DZ, WLBY2)
13100    NSIDE=(LO+2)/3
13130    NCOLS=LO
13160    NROWS=MO
13200    READ(1,200) CODE, FLAG
13300    IF(CODE.EQ.1) CALL VOLTU(NCOLS, NROWS, V)
13400    IF(CODE.EQ.2) CALL VOLTC(NCOLS, NROWS, V)
13500    IF(CODE.EQ.3) CALL TAP(NCOLS, NROWS, DX, DZ, V)
13600    IF(CODE.EQ.4) CALL VOLTK(NCOLS, NROWS, V)
13700    IF(CODE.EQ.5) CALL STEER(NCOLS, NROWS, DX, DZ, V)
13800    IF(CODE.EQ.6) CALL PHASE(NCOLS, NROWS, V)
13900    CALL SHAPE(V, NSIDE, LO)
14000    CALL PATTYN(LO, MO, NO, V, DX, DZ)
14100    STOP
14200    200 FORMAT(3I0, 3F0.0, I0)
14300    300 FORMAT(8E14.6)
14400    END
REFERENCES


