p.

Sensor and Simulation Notes

Note 96

A NUMERICAL METHOD FOR COMPUTING THE PROPAGATION OF AN ELECTROMAGNETIC PULSE GUIDED OVER A MATERIAL INTERFACE

W. E. Page
D. H. Peterson
The Dikewood Corporation

January 1970

ABSTRACT

The formulation of a numerical calculation of the propagation of an electromagnetic pulse guided over material interface is described. The problem is formulated in two-dimensional Cartesian coordinates and time. Provision is made for conducting boundaries to guide the pulse. The numerical calculation employs a mesh that moves with the pulse wave front.

FOREWORD

This note describes numerical methods that have been applied in several different computer codes. These methods are illustrated by describing a code used to analyze some aspects of a ground transmission line simulator. Results of the calculation will be presented separately. We would like to thank Dr. Carl E. Baum of AFWL for helpful suggestions and discussions concerning this problem.

I. INTRODUCTION

This note describes the formulation of a finite difference calculation to analyze the propagation of an electromagnetic pulse guided over an air-ground interface by metallic conductors. The geometrical arrangement is shown in Figure 1 and is designed to approximate an EMP simulator.

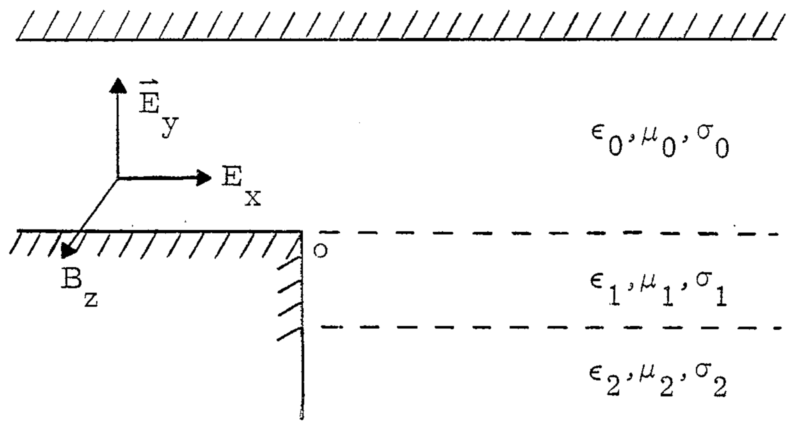


Figure 1

It is assumed that a plane wave pulse is initially established in the left section of the array. This initial plane wave is assumed to have the polarization shown in Figure 1 with $E_x = 0$. The numerical computation begins at t = 0 as the wave front crosses the origin o where the material interface begins. The three field components are then computed throughout the array as functions of time as the wave front propagates to the right. Layered media with different electrical properties can be included as indicated in the figure.

The problem is formulated in two-dimensional Cartesian coordinates so there is no variation in the z direction. The conducting boundaries are shown as straight in Figure 1; they can, however, be slanted or curved.

This note describes the numerical formulation developed to treat the early time behavior of a propagating pulse in the geometry of Figure 1. This is done by using a mesh that contains the wave front and extends to the left of it a distance ct_P where t_P is a specified problem time. As the wave front propagates to the right in Figure 1, zones are added at the "front" of the mesh and deleted at the back so that the mesh effectively moves with the pulse. The behavior of the field components at each mesh point are thus determined out to a certain retarded time τ. This type of calculation was employed since the geometrical complexity of Figure 1 negates most of the computational advantages of transforming the equations to compute directly in terms of τ.

This note describes the numerical techniques used for the computation and describes the computer code developed but does not detail the application and results of the technique. These results will be documented separately.

II. FORMULATION

We take both the air and the earth to be homogeneous, linear media with no sources so that Maxwell's equations are:

$$\nabla \times \vec{E} = -\dot{\vec{B}} \tag{1}$$

$$\nabla \cdot \vec{E} = 0 = \nabla \cdot \vec{B} \tag{2}$$

$$\nabla \times \vec{B} = \mu\epsilon\dot{\vec{E}} + \mu\sigma\vec{E} \tag{3}$$

where

$$D = \epsilon\vec{E}, \qquad \vec{J} = \sigma\vec{E},$$

$$B = \mu\vec{H}, \qquad \epsilon, \mu, \sigma \longrightarrow \text{constant} \tag{4}$$

The fields are determined by these equations plus the equation of continuity

$$\nabla \cdot \vec{J} = 0 \tag{5}$$

The fields can be represented by a Hertz vector $\vec{\Pi}$ which obeys the equation

$$\nabla^2\vec{\Pi} - \epsilon\mu\ddot{\vec{\Pi}} - \mu\sigma\dot{\vec{\Pi}} = 0 \tag{6}$$

where

$$\vec{E} = \nabla \times \nabla \times \vec{\Pi}, \qquad \vec{B} = \mu\epsilon\nabla \times \dot{\vec{\Pi}} + \sigma\mu\nabla \times \vec{\Pi} \tag{7}$$

It is convenient to define a new vector

$$\vec{\Phi} = \nabla \times \vec{\Pi} \tag{8}$$

in terms of which Eqs. (6) and (7) become

$$\nabla^2\vec{\Phi} - \mu\epsilon\ddot{\vec{\Phi}} - \mu\sigma\dot{\vec{\Phi}} = 0 \qquad (9)$$

and

$$\vec{E} = \nabla \times \vec{\Phi}, \qquad \dot{\vec{B}} = \mu\epsilon\dot{\vec{\Phi}} + \sigma\mu\vec{\Phi} \qquad (10)$$

Restricting ourselves to Cartesian coordinates in two spatial dimensions with the initial polarization shown in Figure 1, we have only x and y components of $\vec{E}$ and a z component of $\vec{B}$. We can then take

$$\Phi = (0)\vec{i} + (0)\vec{j} + \phi\vec{k} = \phi(x, y, t) \qquad (11)$$

so that we are concerned with the scaler equation

$$\nabla^2\phi - \mu\epsilon\ddot{\phi} - \mu\sigma\dot{\phi} = 0 \qquad (12)$$

Introducing a fundamental length L and a fundamental time T = cL, Eq. (12) can be written in dimensionless form

$$\nabla^2\phi + C1\ddot{\phi} + C2\dot{\phi} = 0 \qquad (13)$$

where

$$C1 = -c^2\mu\epsilon, \qquad C2 = \frac{c\mu\sigma}{L} \qquad (14)$$

The field components are determined from the potential as

$$E_x = \frac{\partial\phi}{\partial y}, \qquad E_y = -\frac{\partial\phi}{\partial x} \qquad (15)$$

and

$$\dot{B}_z = \nabla \times \vec{E} = -\nabla^2\phi, \qquad B(t) = -\int_0^t (\nabla^2\phi)\,dt \qquad (16)$$

III. DIFFERENCE EQUATIONS

The differential equation (13) can be expanded using Taylor's theorem giving

$$\frac{1}{h^2}\left\{\phi(x+h,y,t) - 2\phi(x,y,t) + \phi(x-h,y,t) + \phi(x,y+h,t) - 2\phi(x,y,t)\right.$$

$$\left. + \phi(x,y-h,t)\right\} + \frac{C1}{k^2}\left\{\phi(x,y,t+k) - 2\phi(x,y,t) + \phi(x,y,t-k)\right\}$$

$$+ \frac{C2}{k}\left\{\phi(x,y,t+k) - \phi(x,y,t)\right\} = \frac{h^2}{12}\left\{\frac{\partial^4\phi}{\partial x^4}(x+\alpha_1 h,y,t) + \frac{\partial^4\phi}{\partial y^4}(x,y+\alpha_2 h,t)\right\}$$

$$+ \frac{k^2 C1}{12}\frac{\partial^4\phi}{\partial t^4}(x,y,t+\alpha_3 k) + \frac{kC2}{2}\frac{\partial^2\phi}{\partial t^2}(x,y,t+\alpha_4 k) \tag{17}$$

where $|\alpha| < 1$

If the higher order derivatives on the right side of Eq. (17) are bounded we can choose h and k small enough that the terms on the right-hand side are negligible.

Approximating the left-hand side of Eq. (17) by a difference equation valid at space and time points that are integral multiples of h and k we have the recursion relation

$$\phi(\ell h, mh, (n+1)k) = A_1\left\{\phi((\ell+1)h, mh, nk) + \phi((\ell-1)h, mh, nk)\right.$$

$$\left. + \phi(\ell h, (m+1)h, nk) + \phi(\ell h, (m-1)h, nk) - 4\phi(\ell h, mh, nk)\right\} + A_2\phi(\ell h, mh, nk)$$

$$+ A_3\phi(\ell h, mh, (n-1)k) \quad \ell, m, n \rightarrow \text{integers} \tag{18}$$

This algorithm gives the values of ϕ at time $(n+1)k$ in terms of those at times nk and $(n-1)k$. The bracketed term on the right in Eq. (18) is a finite difference approximation to the Laplacian $\nabla^2\phi$ at time nk. This quantity can be used to determine the magnetic field from Eq. (16). Since the quantity B/μ is continuous across the material interfaces the Laplacian is also continuous. This boundary condition is used to couple the numerical calculation in the separate material regions of the problem.

The Laplacian calculation of Eq. (18) uses the mesh points shown in Figure 2A to determine the Laplacian at the center point of this array of points. By keeping more terms in the Taylor expansion of Eq. (17) more exact representations of $\nabla^2\phi$ can be devised that involve more points in the mesh and thus more time and complexity in the calculation. The array of Figure 2A corresponds to an only slightly more complex approximation to $\nabla^2\phi$ that has been found by experience to be more satisfactory in some calculations.

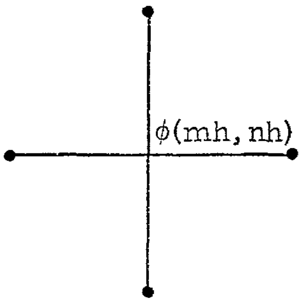


Figure 2A

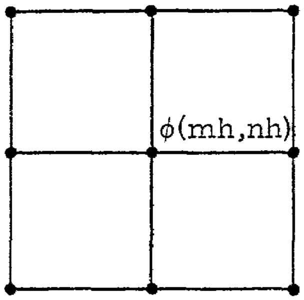


Figure 2B

Using the Laplacian calculation corresponding to Figure 2B, the algorithm of Eq. (18) becomes

$$\begin{aligned}\phi(\ell h, mh, (n+1)k) = A_1 \Big\{ &\Big[\phi((\ell+1)h, (m+1)h, nk) + \phi((\ell-1)h, (m-1)h, nk) \\ &+ \phi((\ell+1)h, (m-1)h, nk) + \phi((\ell-1)h, (m+1)h, nk)\Big] + 4\Big[\phi((\ell+1)h, mh, nk) \\ &+ \phi(\ell h, (m+1)h, nk) + \phi((\ell-1)h, nh, nk) + \phi(\ell h, (m-1)h, nk)\Big] - 20\Big[\phi(\ell h, mh, nk)\Big]\Big\} \\ &+ A_2\phi(\ell h, mh, nk) + A_3\phi(\ell h, mh, (n-1)k) \qquad (19)\end{aligned}$$

$$\ell, m, n \rightarrow \text{integers}$$

The geometrical diagram corresponding to Eq. (19) is shown in Figure 3. This illustrates the basic mesh point cell used in the body of the numerical calculation.

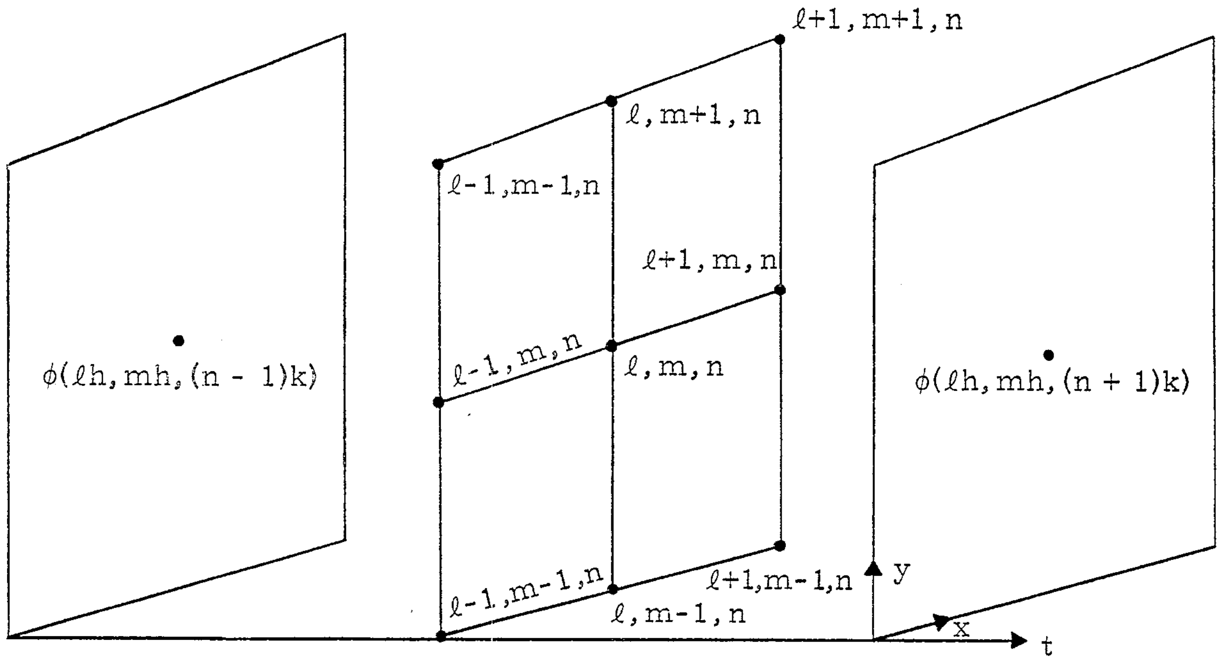


Figure 3

IV. BOUNDARY CONDITIONS

Three types of boundaries occur in the calculation that require modification of the basic computational algorithm given by Eq. (19).

CONDUCTING BOUNDARIES

Conducting boundaries are treated as perfect conductors by demanding that tangential E be zero at the boundary surface, this implies

$$\frac{\partial \phi}{\partial n} = 0 \qquad (n \rightarrow \text{inward normal}) \tag{20}$$

Figure 4 illustrates a curved boundary C cutting through the problem mesh. The inward normal to C at the boundary point ϕ_B will cut through the interior mesh, which has a sufficiently small uniform spacing h, at ϕ_A as shown. The boundary value ϕ_B is determined numerically from the interior points in accordance with Eq. (20) by the equation

$$\phi_B = (\gamma_2 \phi_1 + \gamma_1 \phi_2) \tag{21}$$

Having determined values of ϕ_B wherever the boundary intersects mesh lines it still remains to find $\nabla^2 \phi$ at interior points adjacent to the boundary. If the boundary cuts through the mesh then the array of points available to compute $\nabla^2 \phi$ will not be uniformly spaced.

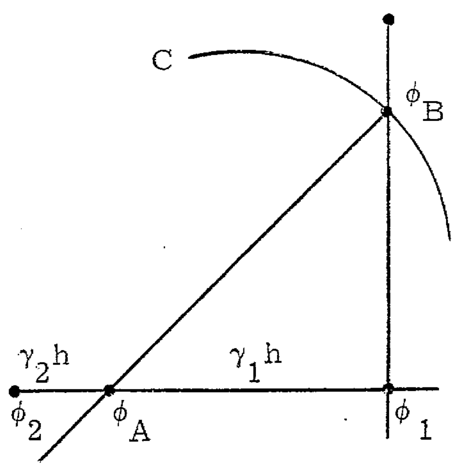


Figure 4

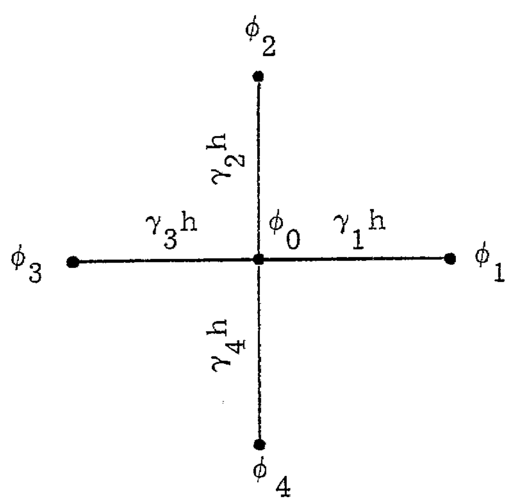


Figure 5

Figure 4 shows an array corresponding to that of Figure 2A but having arbitrary unequal spacing. A Taylor expansion of $\nabla^2\phi$ about $\phi(mh, nk)$ gives the difference approximation

$$\nabla^2\phi = \frac{1}{h^2}\left\{\frac{2\phi_1}{\gamma_1(\gamma_1+\gamma_3)} + \frac{2\phi_2}{\gamma_2(\gamma_2+\gamma_4)} + \frac{2\phi_3}{\gamma_3(\gamma_3+\gamma_1)} + \frac{2\phi_4}{\gamma_4(\gamma_4+\gamma_2)} - 2\phi_0\left(\frac{\gamma_1\gamma_3+\gamma_2\gamma_4}{\gamma_1\gamma_2\gamma_3\gamma_4}\right)\right\} \tag{22}$$

ANALYTIC BOUNDARIES

During the course of the computation part of the problem mesh corresponds to regions where trivial plane wave solutions of the differential equation (13) exist. These regions are determined analytically and not computed numerically. The boundary between these regions is one where $\phi(x, y, t)$ can be specified and used as a boundary condition for the numerical computation of ϕ.

The initial incident plane wave can have any time wave form. One of particular interest is a unit pulse with a linear rise time. This can be used to approximate a unit step input wave form. This incident pulse propagates as a plane wave until disturbed by refraction from the ground-air interface. The solution of Eq. (13) for this case is trivial representing an undisturbed propagation with velocity c. Referring to Figures 8 and 9, the electric field outside the shaded areas will consist of only the component E_y with the wave form indicated in Figure 6. The rise time is T_R and the arrival time of the wave front at the spatial point for which E_y is being determined is T_A. The magnetic field in this region is given by $B = E/c$.

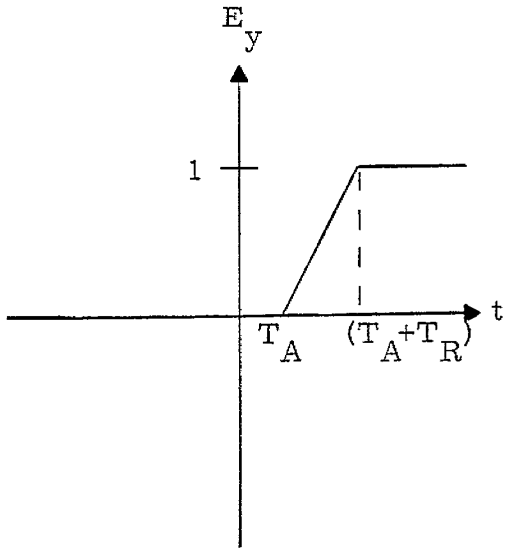


Figure 6

If it is assumed that at time zero the wave front is at the origin (0,0) then the wave front will arrive at any mesh point (x,y) at time t = x/c. Therefore, from Eq. (15) and Figure 4

$$E_y(x,t) = 0 \qquad \text{for} \qquad t < \frac{x}{c} \tag{23}$$

$$E_y(x,t) = \frac{1}{T_R}\left(t - \frac{x}{c}\right) \qquad \text{for} \qquad \frac{x}{c} \leq t < \min\left[\left(\frac{x}{c} + T_R\right), T_x\right] \tag{24}$$

$$E_y = 1 \qquad \text{for} \qquad \min\left[\left(\frac{x}{c} + T_R\right), T_x\right] \leq t \leq T_x \tag{25}$$

where T_x is the time at which the refracted wave from the air-ground interface first arrives at the point x.

Substituting $-\partial\phi/\partial x$ for E_y in Eq. (24) and solving the resulting differential equation for ϕ gives

$$\phi(x,t) = \frac{1}{T_R}\left(\frac{x^2 + c^2t^2}{2c} - tx\right) \qquad \text{for} \qquad \frac{x}{c} \leq t < \min\left[\left(\frac{x}{c} + T_R\right), T_x\right] \tag{26}$$

Equation (25) can be solved by ϕ by first substituting $-\partial\phi/\partial x$ for E_y and then integrating to obtain

$$\phi(x,t) = \left[-x + c\left(t - \frac{T_R}{2} \right) \right] \qquad \text{for} \quad \min\left[\left(\frac{x}{c} + T_R \right),\ T_x \right] \leq t \leq T_x \tag{27}$$

At each time cycle Eqs. (26) and (27) are used to set values for ϕ at all required points outside the shaded area of Figures 8 and 9.

MATERIAL INTERFACES

Material interfaces are taken to lie along coordinate lines, y = constant, corresponding to a mesh line in the calculation. The interface is defined by two sets of mesh, which are spatially superposed but taken to belong to the different regions of the problem. Figure 7 illustrates a section of the mesh at such an interface.

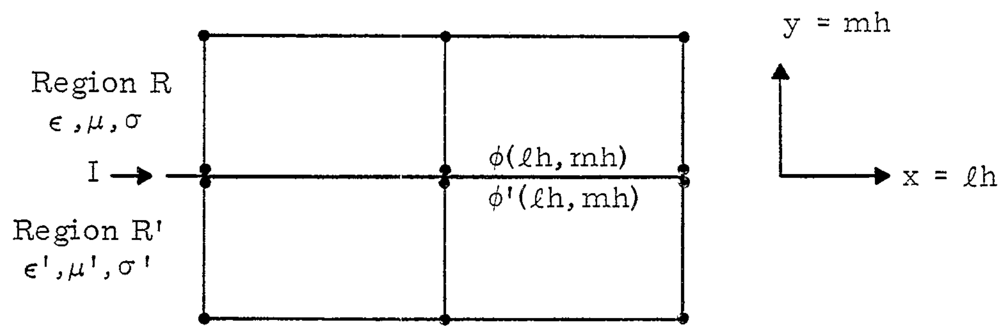


Figure 7

The two material regions correspond to different potential functions ϕ and ϕ' which are solutions of the differential equation (13) with the appropriate values of ϵ, σ, and ϕ. In order to compute values of ϕ and ϕ' numerically, we must evaluate $\nabla^2\phi$ and $\nabla^2\phi'$ for the interface points in the two regions. Since the quantity $1/\mu\nabla^2\phi$ is continuous at the interface we can evaluate it using one-sided differences in each region and average the result to obtain centered difference expressions for the quantity. This procedure gives

$$\frac{1}{\mu} \nabla^2 \phi(\ell h, mh) = \frac{1}{\mu'} \nabla^2 \phi'(\ell h, mh) = \frac{1}{6h^2} \Big\{ \mu \Big[\phi((\ell + 1)h, (m + 1)h)$$

$$+ \phi((\ell + 1)h, (m - 1)h) + 2\phi((\ell + 1)h, mh) + 2\phi((\ell - 1)h, mh) + 4\phi(\ell h, (m + 1)h)$$

$$- 10\phi(\ell h, mh) \Big] + \mu' \Big[\phi'((\ell - 1)h, (m - 1)h) + \phi'((\ell - 1)h, (m + 1)h$$

$$+ 2\phi'((\ell + 1)h, mh) + 2\phi'((\ell - 1)h, mh) + 4\phi'(\ell h, (m - 1)h) - 10\phi'(\ell h, mh) \Big] \Big\} \quad (28)$$

Equation (28) allows us to determine $\nabla^2 \phi$ for interface points in both regions. We can then use this quantity in the algorithm for computing the potential at these points.

V. LOGICAL STRUCTURE OF PROGRAM MOVMSH2

Several different computer programs have been written to apply the procedures discussed above to particular problems. The program MOVMSH2 was written to carry out a parametric set of calculations on the geometry shown in Figure 1. Only two material layers are included. This code is described here to illustrate the application of the numerical technique.

The application of the finite difference formulas given above can be visualized by referring to Figures 8 through 11. Figure 8 represents the computational mesh and boundaries at an early time t when the incident plane wave front W has progressed a distance $\overline{o\ell} = ct$ to the right past the origin o. The diffraction effects in the air arising from propagation in the ground are constrained by causality to the interior of the circle B whose radius is $\overline{o\ell}$. Outside this circle the incident pulse continues to propagate as a plane wave so that the trivial solution can be used to determine ϕ on this boundary. Below the ground the wave traveling with velocity v has penetrated to a maximum depth $\overline{od} = vt$. The line $\overline{d\ell}$ is the wave front in the ground and is thus a boundary on which $\phi = 0$. The numerical computation is then limited to mesh points inside the shaded area of Figure 8, with the appropriate boundary condition enforced on the boundaries of this region. A problem time t_P is selected which determines the extent of the computational mesh and limits the retarded time that the computation is carried to at any mesh point. Figure 9 shows the configuration of the mesh at a later time $t > t_P$. The mesh has in this case reached its maximum extent determined by ct_P and vt_P. As the computation is carried further the shaded mesh can be visualized as moving to the right along with the wave front. The boundary B will continue to approach the wave front W.

The line b is an artificial mesh boundary where the calculation is cut off. These boundary points are determined separately from the numerical calculation of interior mesh points by simply extrapolating from neighboring points. This approximation is less accurate than the numerical algorithm

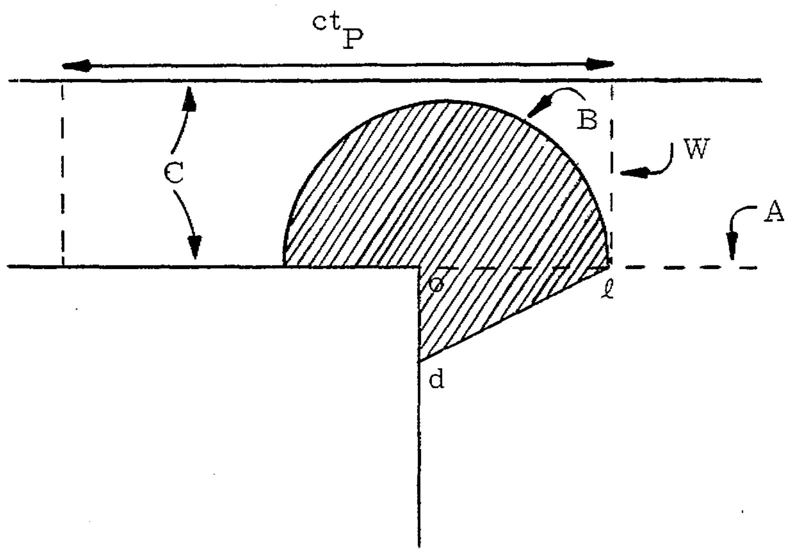


Figure 8

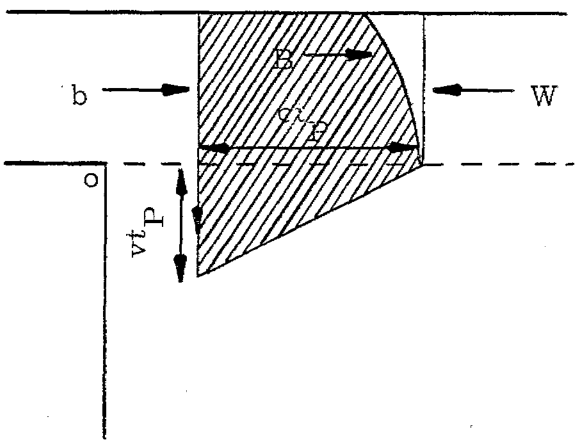


Figure 9

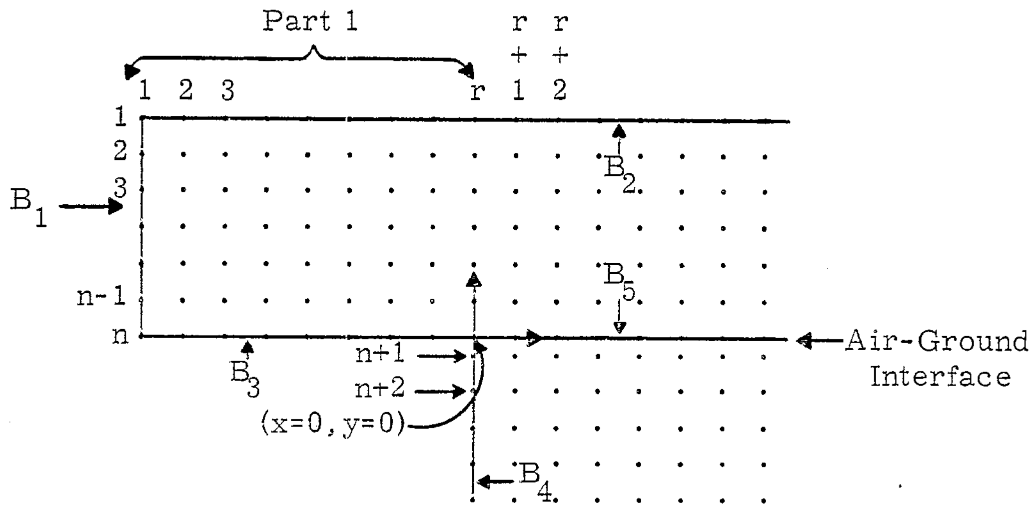


Figure 10

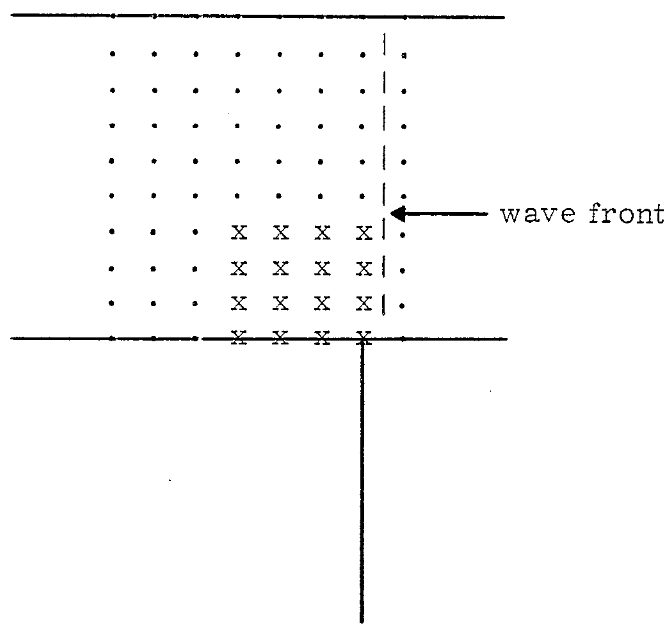


Figure 11

Note: The x's indicate those points at which initial values for ϕ are set analytically.

but since the mesh effectively moves with the wave front with velocity c, these perturbations will not propagate away from the artificial boundary.

Program MOVMSH2 operates on a moving array of points taken from the equally spaced mesh shown in Figure 10. At the start of the calculation the wave front lies between the rth and the r + 1st column of Figure 10. Part 1 of Figure 10 is then treated as the source of a plane electromagnetic pulse polarized as shown in Figure 1 such that, for each mesh point in Part 1, E_y increases as a linear function of time until a specified maximum is reached.* E_y then remains constant and E_x is continuously zero until both quantities are perturbed by a reflection from the pulse after the wave has moved out over the air-ground interface.

The boundaries B_2, B_3, and B_4, in Figure 10 are assumed to be conducting boundaries such that at any time t'

$$\phi(x_j, y_1, t') = \phi(x_j, y_2, t') \qquad \text{for all } j \tag{29}$$

$$\phi(x_j, y_n, t') = \phi(x_j, y_{n-1}, t') \qquad \text{for } j \leq r+1 \text{ or } j \geq r+1 \tag{30}$$

$$\phi(x_{r+1}, y_i, t') = \phi(x_{r+2}, y_i, t') \qquad \text{for } i \geq n+1 \tag{31}$$

The quantities $\phi^-_{i,j}$, $\phi_{i,j}$, and $\phi^+_{i,j}$ refer to values for $\phi(x,y,t)$ at that space point located at the intersection of the ith row and jth column of the mesh of Figure 10. These values correspond to a monotonically increasing sequence of time values $t - \Delta t$, t, $t + \Delta t$ where Δt is the time step to be used in updating the function ϕ.

The calculations used to update ϕ values at points interior to the mesh and inside the shaded areas of Figures 8 and 9 are of the form

$$\phi^+_{i,j} = A_1 L_{i,j} + A_2 \phi_{i,j} + A_3 \phi^-_{i,j} \tag{32}$$

---

* The restriction that E_y be a linear function of time can easily be removed by a simple change of the functional form for $\phi(x,y,t)$ in the Subroutine PHI.

for $i \leq n$, and

$$\phi^{+}_{i,j} = G_1 L_{i,j} + G_2 \phi_{i,j} + G_3 \phi^{-}_{i,j} \tag{33}$$

for $i > n$. In these expressions for ϕ, the quantities A_1, A_2, A_3, G_1, G_2, and G_3 represent constants, and $L_{i,j}$ represents the Laplacian $\nabla^2 \phi$. Points lying above ground but outside the shaded areas of Figures 8 and 9 are determined according to Eqs. (26) and (27).

For any column of the mesh the two interior points lying in the nth and n + 1st rows are always considered to occupy essentially the same point in space. Of these two points, the one from the nth row will be treated as a point lying above ground. The one from the n + 1st row will be assumed to lie beneath the ground. Since in this code $\mu(\text{air}) = \mu(\text{ground})$, $\nabla^2 \phi$ is required to be continuous across the air-ground boundary the Laplacian calculation for points in these two rows must be such that for each j

$$L_{n,j} = L_{n+1,j}$$

where $L_{n,j}$ is computed according to Eq. (28).

When all of the required interior points have been updated those points on the boundaries B_2, B_3, and B_4, when included in the traveling mesh, are updated according to Eqs. (29), (30), and (31), respectively. The boundary B_1 is not a physical boundary but represents the termination of the calculation. Points on B_1 are determined at each time step by linearly extrapolating those values lying in the two columns immediately to the right of B_1.

Those columns from Figure 10, which comprise the traveling mesh, are adjusted with time to insure that the wave front always lies between the two columns at the extreme right of the mesh. This process requires the periodic addition of a new column on the right-hand side of the traveling mesh accompanied by a corresponding deletion of the leftmost column of the mesh.

VI. FUNCTIONAL DESCRIPTION OF PROGRAM MOVMSH2

MOVMSH2 MAIN PROGRAM

The MOVMSH2 main program is used to read input values, set program constants, and to provide a mechanism for transferring from the linear storage blocks, established during set-up procedures, to the two and three dimensional arrays used for referencing Laplacian and ϕ values during program execution.

SUBROUTINE MAIN

The basic purpose of this subroutine is to control the sequence of operations performed by the other program subroutines. A flow diagram for Subroutine MAIN can be found on pages 22 through 25.

SUBROUTINE START

This subroutine uses Eqs. (26) and (27) to set initial values for ϕ for those points shown in Figure 11 at that time when the wave front has just started to move out over the air-ground interface.

Subroutine START is also used to compute the depth of penetration of the pulse into the ground. This depth is then indexed in terms of the number of mesh rows penetrated as a function of column number in the traveling mesh.

SUBROUTINE POINTS

This subroutine is used to compute the mesh row numbers which correspond to the input array of y values for plot point coordinates. Since the mesh travels with time, the relationship between mesh column numbers and the x coordinates of plot points varies with time. Therefore, this relationship is established in the Subroutine UPTAPE which is called each time output information is requested.

SUBROUTINE CMPLMTS

This subroutine is used to determine those points, for the current time cycle, which must be updated according to the finite difference equations (32) and (33). It also determines which points should be set analytically by using Eqs. (26) and (27) in order to have ϕ values available to compute the Laplacians required for updating an expanded array of ϕ values at the next time cycle.

SUBROUTINE LAPLACE

This subroutine uses Eqs. (18), (22), and (28) to calculate Laplacians.

SUBROUTINE PHI

This subroutine uses Eqs. (26), (27), (32), and (33), as appropriate, to update ϕ values at all interior points of the traveling mesh.

SUBROUTINE BNDRY

This subroutine is used to update ϕ values along the boundaries B_1, B_2, B_3, and B_4 of Figure 10.

SUBROUTINE UPTAPE

This subroutine is used to compute E_x and E_y. It also stores these values at all mesh points at which output has been requested.

SUBROUTINE FINISH

This subroutine is used to compute values for the B field and to print and plot values for E_x, E_y, $\sqrt{E_x^2 + E_y^2}$, B, and $\dot{B}$.

PROGRAM INPUTS, STORAGE, AND SET UP PROCEDURES

In order to make a run with Program MOVMSH2, the user must first decide on the relative position of the boundaries B_2, B_3, and B_4 (Figure 10), the (x, y) coordinates of those points at which output will be requested, and the period of time over which output values are to be observed. The size of the mesh step, the number of update time cycles per mesh step and the overall width of the traveling mesh must also be specified. The names and descriptions of the variables which control these and other program processes are as follows.

Input Variables

Variable	Description and Comments
DELX	The distance (in meters) between adjacent mesh points.
EYMAX	The maximum value the y component of the electromagnetic field is required to reach (see Figure 6).
MPMOD	The modulus used to determine the rate at which plot information will be saved (i.e., MPMOD = 1 implies data will be saved at each time cycle, MPMOD = 2 implies every other time cycle, etc.).
NC	The number of columns in the traveling mesh. (Note that this quantity limits the time at which information can be observed at any space point.)
NCYCLES	The number of time cycles the program is required to run. (This value determines how far the wave front will advance beyond the origin of the (x, y) coordinate system.)
NPTS	The number of mesh points at which output data will be saved.
NR1	The number of rows in Part 1 of the mesh (see Figure 10).
NR2	The number of rows in Part 2 of the mesh. This value depends on the width of the traveling mesh and should be set equal to $\left[NR1 + NC * (VLG/VLA) + 1\right]$.
NTSPSS	The number of update time cycles per space step.
TRISE	The rise time of the input pulse in seconds (see Figure 6).
VLA	The velocity of light in the upper medium (air) in meters/sec.
VLG	The velocity of light in the lower medium (ground) in meters/sec.

Variable	Description and Comments
XX(J)	The x coordinate in meters of the Jth point at which output is requested J = 1, NPTS.
YY(J)	Similar to XX(J) but refers to the y coordinate.

In addition to specifying values for the above set of inputs, the user may be required to adjust the dimensions of certain blocks in the MOVMSH2 main program. In particular, if the block PH is dimensioned N, then N must be at least as large as NR2 * NC * 2. The minimum dimension for the block XLP would be NR2 * NC.

An approximation for the amount of central processor time required for a given run can be obtained by the formula

CPU Time (seconds) = .000064(NCYCLES)(NC)(NR1 + NC/6)

PROGRAM OUTPUT

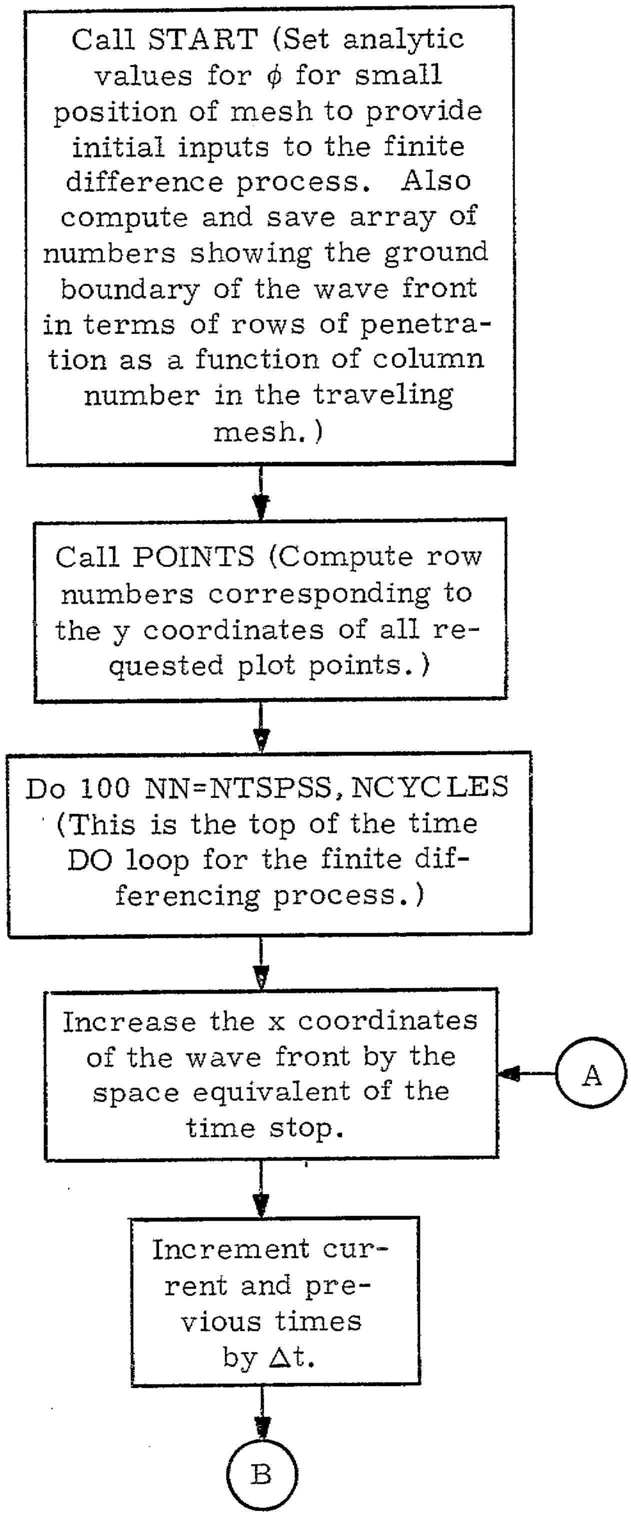
Printed and Tape Output

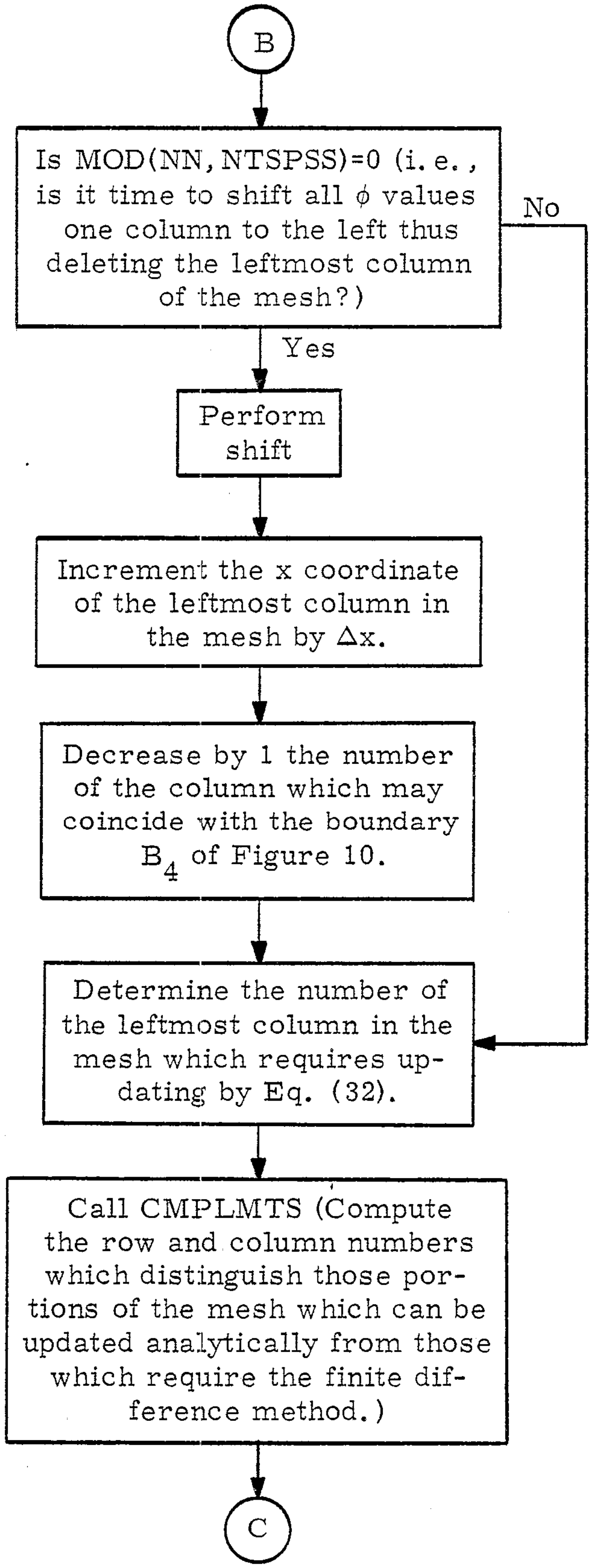
The printed output from Program MOVMSH2 includes all of the input variables listed above. It also includes a time history of E_x, E_y, $\sqrt{E_x^2 + E_y^2}$, B, and $\dot{B}$ at each of the mesh points at which output data has been requested.

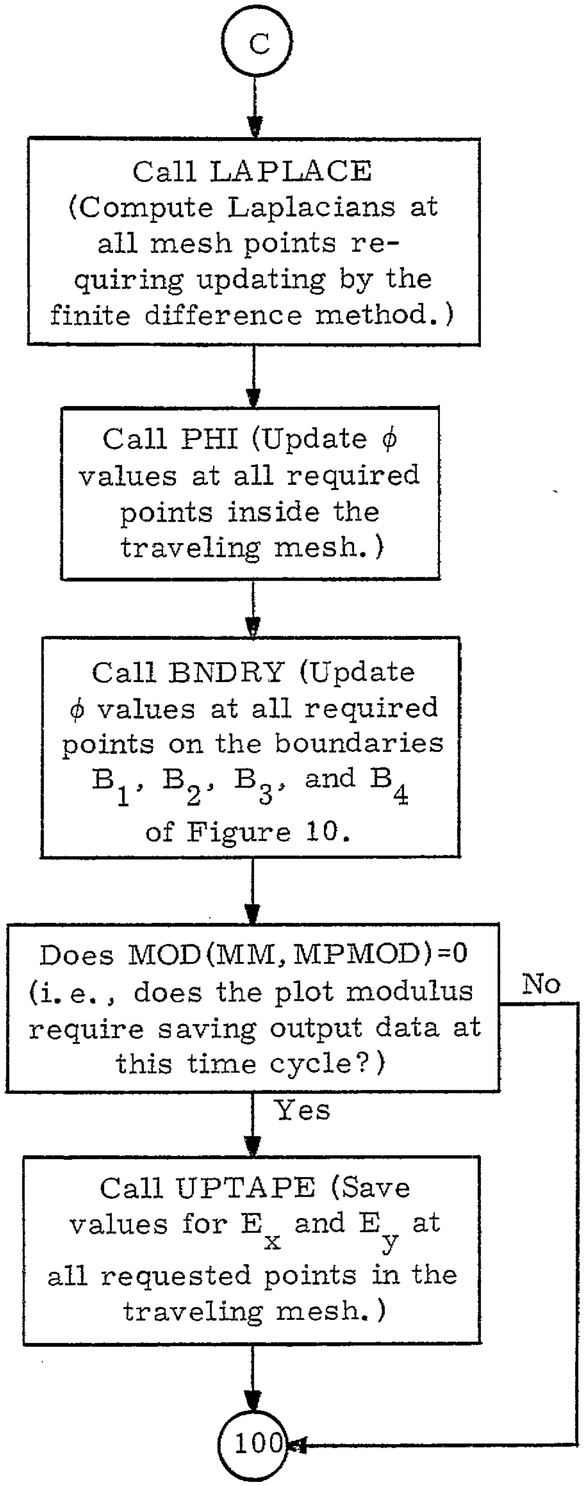
Microfilm Output

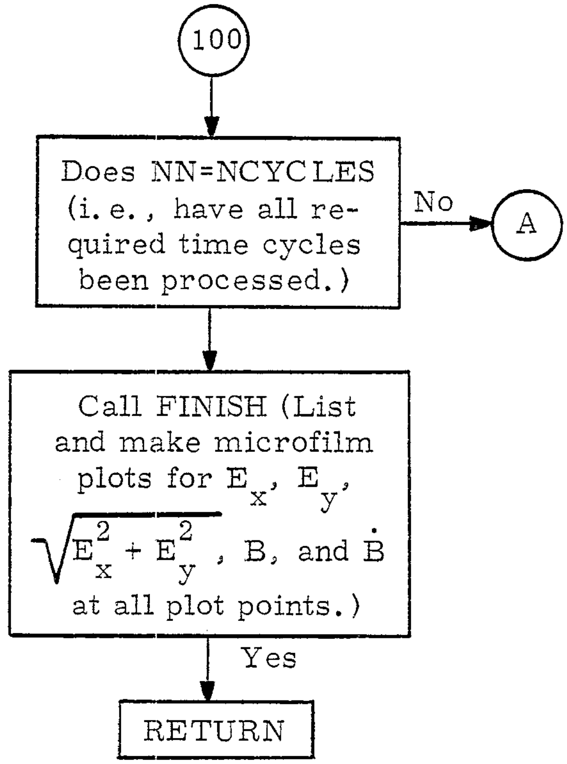
The program produces microfilm output showing each of the quantities E_x, E_y, $\sqrt{E_x^2 + E_y^2}$, B, and $\dot{B}$ plotted against time. Separate plots are produced for each of these functions at each of the mesh points at which output has been requested.

FLOWCHART FOR SUBROUTINE MAIN




B
Is MOD(NN, NTSPSS)=0 (i. e., is it time to shift all ϕ values one column to the left thus deleting the leftmost column of the mesh?)
No
Yes
Perform shift
Increment the x coordinate of the leftmost column in the mesh by Δx.
Decrease by 1 the number of the column which may coincide with the boundary B4 of Figure 10.
Determine the number of the leftmost column in the mesh which requires updating by Eq. (32).
Call CMPLMTS (Compute the row and column numbers which distinguish those portions of the mesh which can be updated analytically from those which require the finite difference method.)
C


C
Call LAPLACE (Compute Laplacians at all mesh points requiring updating by the finite difference method.)
Call PHI (Update ϕ values at all required points inside the traveling mesh.)
Call BNDRY (Update ϕ values at all required points on the boundaries B_1, B_2, B_3, and B_4 of Figure 10.
Does MOD(MM, MPMOD)=0 (i.e., does the plot modulus require saving output data at this time cycle?)
No
Yes
Call UPTAPE (Save values for E_x and E_y at all requested points in the traveling mesh.)
100


100
Does NN=NCYCLES (i.e., have all required time cycles been processed.)
No
A
Call FINISH (List and make microfilm plots for E_x, E_y, $\sqrt{E_x^2 + E_y^2}$, B, and $\dot{B}$ at all plot points.)
Yes
RETURN

VII. GLOSSARY FOR PROGRAM MOVMSH2

The following variables appear in the MOVMSH2 main program, or in common storage. (For dimensioned variables an M and N in a subscript position will indicate that the dimensions of the block must be set by the user.)

Variable	Description
AMU	The magnetic permeability.
C1A, C2A, C3A, C1G, C2G, C3G	Coefficients used in updating ϕ values according to Eqs. (32) and (33).
DELTIME	The time step used in updating ϕ values.
DELXSQ	The square of the space step between mesh points.
DT2	The square of the time step.
DXPTS	The space step corresponding to a time step.
EPSILON	The minimum distance permitted between the wave front and any column of the mesh.
JS	The number of time frames of output data presently stored in the block XOUT.
LR(100)	LR(J) is the row number corresponding to the y coordinate of the Jth plot point.
MXLP1A2	This quantity is a function of time and is the leftmost column of the mesh at which ϕ values require updating by the finite difference method.
NCLB	The number of the column in the traveling mesh which coincides with the boundary B_4 of Figure 10.
NCM1	NC1 - 1.
NDUMPS	The number of output files of plot information written on the disk.

Variable	Description
NFCI	The leftmost column of the mesh which intersects the circle of Figure 8.
NFR(1001)	NFR(K) is the row number of the last point in the Kth column that lies inside the circle of Figure 8.
NLCLI	The rightmost column of the mesh at which the pulse has penetrated at least one space step into the ground.
NLR(1001)	NLR(K) is the depth of penetration of the pulse at the Kth column measured in rows beneath the ground.
NR1M1	NR1 - 1.
NR1P1	NR1 + 1.
NR1P2	NR1 + 2.
PH(N)	Storage for ϕ values.
SIGA, SIGG	The conductivity of the air and ground, respectively.
TLAST	The time for the previous time cycle.
TNOW	The time for the current time cycle.
XLG	The x coordinate of the leftmost column in the traveling mesh.
XLP(M)	Storage for the array of $\nabla^2\phi$ values.
XOUT(5,100)	XOUT(1, J) associates this data frame with the x,y coordinates of a particular point at which output has been requested.

$$XOUT(2,J) = \text{Time}$$

$$XOUT(3,J) = \partial\phi/\partial y$$

$$XOUT(4,J) = \partial\phi/\partial x$$

$$XOUT(5,J) = \nabla^2\phi$$

Variable	Description
XWF	The x coordinate of the wave front.
Y1, Y2, Y3, Y4	Temporary storage.

DESCRIPTION OF VARIABLES FOR SUBROUTINE MAIN

Variable	Description
J	Run subscript.
JJ	Indicates if mesh columns are to be shifted.
K, L	Column and time subscripts.
MM	Used to assure that the mesh will be shifted after the first pass through the time DO loop.
NJ	Indicates if plot information is required for the current time cycle.
NN	Index for the time cycle DO loop.
PH(NR2, NC, 2)	PH(J, K, L) refers to the value of ϕ in the Jth row and Kth column of the traveling mesh. L = 1 implies ϕ is for current time cycle. L = 2 implies the previous time cycle.
XLP(NR2, NC)	XLP(J, K) corresponds to PH(J, K, 1) but refers to $\nabla^2 \phi$.

DESCRIPTION OF VARIABLES FOR SUBROUTINE START

Variable	Description
C2T2	Temporary storage.
J, K, L	Row, column, and time subscripts.

Variable	Description
N1, N2	Used to define the column and row bounds for setting initial values for ϕ.
PH(NR2, NC, 2)	Same as in Subroutine MAIN.
TIME	Used to define the times at which initial values for ϕ are set.
VAL	ϕ value.
X	Used to define x coordinates of points at which initial values for ϕ are required.
XDC	The x coordinate divided by the velocity of light.
XLP(NR2, NC)	Same as in Subroutine MAIN.

DESCRIPTION OF VARIABLES FOR SUBROUTINE POINTS

Variable	Description
DPTHG	The y coordinate of the air-ground boundary relative to a coordinate system where the zero value for y corresponds to the boundary B_2 of Figure 10.
J	Plot point subscript.
NR2	Same as input variable NR2.
NROW	Row indicator for the y coordinate of a plot point.
TMP	Temporary storage.
Y	The y coordinate of a plot point relative to the coordinate system described for DPTHG above.

DESCRIPTION OF VARIABLES FOR SUBROUTINE CMPLMTS

Variable	Description
DIST	A distance tested to determine if a column requires updating by the finite difference method.
DST	A distance used in determining the uppermost point in a given column which requires updating by the finite difference method.
DX	The distance from a column to the origin of the x,y system.
DXS	$(DX)^2$.
DY	A measure of distance from the Y axis.
K	Column index.
KK	Column index.
N1,N2	Temporary storage.
NSUB	The row number of the uppermost point in a given column lying inside the circle of Figure 8.
X	The distance used in determining if a given column intersects the circle of Figure 8.

DESCRIPTION OF VARIABLES FOR SUBROUTINE LAPLACE

Variable	Description
J,JM,JP	Row subscripts.
K,KM,KP	Column subscripts.
GAMMA	The distance from the wave front to the first column inside the wave front.
N1,N2	Row subscripts.

Variable	Description
PH(NR2, NC)	Same as PH(NR2, NC, 1) in Subroutine MAIN.
S1, S2, T1, T2, T3 TMP1, TMP2, TMP3	Temporary storage.
XLP(NR2, NC)	Same as in Subroutine MAIN.

DESCRIPTION OF VARIABLES FOR SUBROUTINE PHI

Variable	Description
EMDTR	EYMAX/TRISE.
J, J1	Row subscripts.
K, K1	Column subscripts.
N1, N2	Row subscripts.
NC	Same as input variable NC.
NR2	Same as input variable NR2.
NSB	The one dimensional equivalent of the subscript pair (J, K).
PH(NR2, NC)	Same as PH(NR2, NC, 1) in Subroutine MAIN.
PHM(NR2, NC)	Same as PH(NR2, NC, 2) in Subroutine MAIN.
TMP, VAL, VAL2	Temporary storage.
X	The x coordinate of a column.
XDC	The time required for the pulse to arrive at some point x.
XLP(NR2, NC)	Same as in Subroutine MAIN.

DESCRIPTION OF VARIABLES FOR SUBROUTINE BNDRY

Variable	Description
J	Row subscript.
K	Column subscript.
NC	Same as input variable NC.
NR2	Same as input variable NR2.
PH(NR2,NC)	Same as PH(NR2,NC,1) in Subroutine MAIN.

DESCRIPTION OF VARIABLES FOR SUBROUTINE UPTAPE

Variable	Description
DCOL	Floating point representation of the mesh column in which a plot point currently lies.
J,JR	Row subscripts.
NC	Same as input variable NC.
NCOL	DCOL truncated to an integer.
NR	Same as input variable NR2.
PH(NR,NC)	PH(J,K) corresponds to PH(J,K,1) in Subroutine MAIN.
XLP(NR,NC)	Same as in Subroutine MAIN.

DESCRIPTION OF VARIABLES FOR SUBROUTINE FINISH

Variable	Description
BSUM	B as defined in Eq. (4).
J	Frame counter for disk file of output points.

Variable	Description
JP	Frame counter for new file of plot values.
M	Plot point identifier.
N	Record count from disk file.
NPL	Plot point identifier from output array.
NPLTPTS	Maximum linear array of points which can be stored in output file.
PH(NPLTPTS, 6)	Used to store plot point file.
X2	Time
X3	E_x
X4	E_y
X5	$\nabla^2\phi$
X6	$\sqrt{E_x^2 + E_y^2}$

VIII. PROGRAM LISTING

```
      PROGRAM MOVMSH2 (INPUT,OUTPUT,TAPE1)
      DIMENSION PH(15000), XLP(7500)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      COMMON XX(100),YY(100),LR(100),XOUT(5,100),MPMOD
      COMMON /S/ JS,NDUMPS
      READ 6, TRUN
      READ 2, NPASS
      DO 1 NNN=1,NPASS
      JS=0
      NDUMPS=0
      REWIND 1
      READ 2, NTSPSS,NCYCLES,NR1,NR2,NC,NPTS,MPMOD
      PRINT 3, NTSPSS,NCYCLES,NR1,NR2,NC,NPTS,MPMOD
C     NTSPSS MUST BE GREATER THAN 1.
      READ 4, DELX,EYMAX,VLA,VLG,TRISE
      PRINT 5, DELX,EYMAX,VLA,VLG,TRISE
      READ 6, ((XX(J),YY(J)),J=1,NPTS)
      DXPTS=DELX/NTSPSS
      DELTIME=DELX/(VLA*NTSPSS)
      NCM1=NC-1
      NR1M1=NR1-1
      NR1P1=NR1+1
      NR1P2=NR1+2
      NR2M1=NR2-1
      DELXSQ=DELX*DELX
      EPSILON=.5*DELX/NTSPSS
      DS2=DELX*DELX
      DT2=DELTIME*DELTIME
      EP1=8.854E-12
      EP2=10.*EP1
      SIGA=0.
      SIGG=1.E-3
      PRINT 7, SIGG,SIGA
      SIG=SIGG
      AMU=1.257E-6
      Y1=2.*EP1*AMU
```

```
      Y3=2.*EP2*AMU
      Y2=DELTIME*SIGA*AMU
      Y4=DELTIME*SIGG*AMU
      C1A=2.*DT2/(DS2*(Y1+Y2))
      C1G=2.*DT2/(DS2*(Y3+Y4))
      C2A=2.*Y1/(Y1+Y2)
      C2G=2.*Y3/(Y3+Y4)
      C3A=(Y2-Y1)/(Y1+Y2)
      C3G=(Y4-Y3)/(Y3+Y4)
      PRINT 8, C1A,C2A,C3A,C1G,C2G,C3G
      CALL MAIN (PH,XLP,NR2,NC)
1     CONTINUE
      STOP 10
C
2     FORMAT (1H0,I9,7I10)
3     FORMAT (1H0///,11H NTSPSS = ,I4,13H,  NCYCLES = ,I4,9H,
     1NR1 = ,I4,9H,  NR2 = ,I4,8H,  NC = ,I4,10H,  NPTS = ,I4,11H,
     2MPMOD = ,I2,/)
4     FORMAT (5E15.3)
5     FORMAT (1H0/,9H DELX = ,E15.3,11H,  EYMAX = ,E15.3,9H,
     1VLA = ,E15.3,10H,  VLG = ,E15.3,11H,  TRISE = ,E15.3(/)
6     FORMAT (8F10.0)
7     FORMAT (9H SIGG = ,E15.3,10H,  SIGA = ,E15.3,/)
8     FORMAT (8H C1A = ,E20.3,9H,  C2A = ,E20.3,9H,  C3A = ,E20.3,/
     1,8H C1G = ,E20.3,9H,  C2G = ,E20.3,9H,  C3G = ,E20.3,/)
      END
```

SUBROUTINE MAIN (PH, XLP, NR2, NC)

```
      SUBROUTINE MAIN (PH,XLP,NR2,NC)
      DIMENSION LHEAD(100)
      DIMENSION PH(NR2,NC,2),  XLP(NR2,NC)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET / DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      COMMON XX(100),YY(100),LR(100),XOUT(5,100),MPMOD
      CALL SECOND (T1)
      CALL START (PH,XLP,NR2,NC)
      CALL POINTS
      DO 7 NN=NTSPSS,NCYCLES
      XWF=XWF+DXPTS
      TLAST=TNOW
      TNOW=TNOW+DELTIME
      JJ=MOD(NN,NTSPSS)
      IF (JJ) 3,1,3
1     DO 2 J=1,NR2
      DO 2 K=1,NCM1
      DO 2 L=1,2
2     PH(J,K,L)=PH(J,K+1,L)
      XLC=XLC+DELX
      NCLB=NCLB-1
3     MXLP1A2=MAX0((NCLB+1),2)
      CALL CMPLMTS
      CALL LAPLACE (PH,XLP,NR2,NC)
      CALL PHI (PH,PH(1,1,2),XLP,NR2,NC)
      CALL BNDRY (PH,NR2,NC)
      MM=NN-NTSPSS
      NJ=MOD(MM,MPMOD)
      IF (NJ) 5,4,5
4     CALL UPTAPE (PH,XLP,NR2,NC)
5     CONTINUE
      CALL SECOND (T2)
      IF (T2-T1-TRUN) 7,6,6
6     NCYCLES=NN
      PRINT 8, NCYCLES
7     CONTINUE
      PRINT 9, NCYCLES,T2
      NPLTPTS=NC*NTSPSS+10
      CALL FINISH (PH,NPLTPTS)
```

```
      RETURN
C
8     FORMAT (1H0///,35H TIME ESTIMATE EXCEEDED AT NCYCLES
     1=,I4,///)
9     FORMAT (1H0///,17H RUNNING TIME FOR,I5,9H CYCLES =,F8.3,
     15H SEC.,/)
      END
```

SUBROUTINE START (PH,XLP,NR2,NC)

```
      SUBROUTINE START (PH,XLP,NR2,NC)
      DIMENSION PH(NR2,NC,2), XLP(NR2,NC)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      DO 1 J=1,NR2
      DO 1 K=1,NC
      XLP(J,K)=0.0
      DO 1 L=1,2
1     PH(J,K,L)=0.0
      TNOW=(NTSPSS-1)*DELTIME+EPSILON/VLA
      TLAST=TNOW-DELTIME
      XWF=(NTSPSS-1)*DELX/NTSPSS+EPSILON
      XLC=-DELX*(NC-2)
      NCLB=NCM1
      NFCI=NCM1
      N1=NCM1-2
      N2=NR1-3
      TIME=TNOW+DELTIME
      DO 7 L=1,2
      TIME=TIME-DELTIME
      X=-3.*DELX
      DO 6 K=N1,NCM1
      X=X+DELX
      C2T2=(VLA*TIME)**2
      EMDTR=EYMAX/TRISE
      XDC=X/VLA
      IF (TIME-XDC-TRISE) 2,3,3
2     VAL=EMDTR*((X*X+C2T2)/(2.*VLA)-TIME*X)
      GO TO 4
3     VAL=EYMAX*(-X+VLA*(TIME-TRISE/2.))
4     CONTINUE
      DO 5 J=N2,NR1
5     PH(J,K,L)=VAL
6     CONTINUE
7     CONTINUE
      NLCLI=NCM1
      DO 9 K=2,NCM1
      NTMP=(VLG/VLA)*(NC-K)+EPSILON+NR1P1
      IF (NTMP-NR1P1) 8,8,9
```

```
8     NLCLI=K-1
      GO TO 10
9     NLR(K)=MIN0(NTMP, NR2M1)
10    CONTINUE
      N1=NLCLI+1
      DO 11 K=N1, NCM1
11    NLR(K)=NR1P1
      RETURN
      END
```

SUBROUTINE POINTS (IPTS)

```
      SUBROUTINE POINTS (IPTS)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      COMMON /S/ NDUMPS,JS
      COMMON XX(100),YY(100),LR(100),XOUT(5,100),MPMOD
      DPTHG=NR1M1*DELX
      NR2=NR2M1+1
      DO 15 J=1,NPTS
      TMP=XX(J)/DELX
      NTMP=TMP
      IF (TMP-NTMP-.5) 2,2,1
1     NTMP=NTMP+1
2     XX(J)=NTMP*DELX
      Y=YY(J)
      IF (Y) 6,3,6
3     IF (SIGN(1.0,Y)) 4,5,5
4     LR(J)=NR1+1
      GO TO 15
5     LR(J)=NR1
      GO TO 15
6     Y=DPTHG-Y
      TMP=Y/DELX
      NROW=TMP
      IF (TMP-NROW-.5) 8,8,7
7     LR(J)=NROW+2
      GO TO 9
8     LR(J)=NROW+1
9     IF (YY(J)) 10,11,11
10    LR(J)=LR(J)+1
11    IF (LR(J)) 12,13,13
12    LR(J)=1
      YY(J)=DPTHG
13    IF (LR(J)-NR2-1) 15,15,14
14    LR(J)=NR2-1
      YY(J)=-(NR2-NR1-2)*DELX
15    CONTINUE
      PRINT 16
      PRINT 17
      PRINT 18, ((J,XX(J),YY(J),LR(J)),J=1,NPTS)
```

```
      RETURN
C
16    FORMAT  (78H           THE FOLLOWING LIST CONTAINS ALL POINTS
     1AT WHICH OUTPUT WAS REQUESTED/)
17    FORMAT  (87H                   POINT NUMBER                    X
     1                  Y                             ROW,//)
18    FORMAT  (12X,I5,16X,F10.2,12X,F10.2,16X,I5)
      END
```

SUBROUTINE CMPLMTS

```
      SUBROUTINE CMPLMTS
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      IF (NFCI-2) 4,4,1
1     X=XLC+(NFCI-1)*DELX
      DO 3 KK=3,NFCI
      X=X-DELX
      DIST=SQRTF(X*X+DELXSQ)
      IF (DIST-XWF) 2,4,4
2     NFCI=NFCI-1
3     CONTINUE
      NFCI=MAX0(NFCI,2)
4     IF (NFCI-NCLB) 5,5,9
5     NSUB=NR1M1
      DO 8 K=NFCI,NCLB
      IF (NSUB-2) 8,8,6
6     DX=(NCLB-K)*DELX
      DXS=DX*DX
      N1=NSUB-2
      DO 7 J=1,N1
      DY=(NR1-NSUB+1)*DELX
      DST=SQRTF(DY*DY+DXS)
      IF (DST-XWF) 7,8,8
7     NSUB=NSUB-1
8     NFR(K)=NSUB
      N1=NCLB+1
      GO TO 10
9     N1=2
10    NSUB=2
      DO 15 K=N1,NCM1
      DX=(K-NCLB)*DELX
      DXS=DX*DX
      IF (NSUB-NR1M1) 11,11,14
11    DO 12 J=NSUB,NR1
      DY=(NR1-NSUB)*DELX
      DIST=SQRTF(DY*DY+DXS)
      IF (DIST-XWF) 13,12,12
12    NSUB=NSUB+1
13    NFR(K)=NSUB
      GO TO 15
```

```
14   NFR(K)=NR1
15   NFR(K)=MIN0(NFR(K),NR1)
     IF (NFCI-2) 18,18,16
16   N2=NFCI-1
     DO 17 K=2,N2
17   NFR(K)=NR1
18   CONTINUE
     RETURN
     END
```

SUBROUTINE LAPLACE (PH,XLP,NR2,NC)

```
      SUBROUTINE LAPLACE (PH,XLP,NR2,NC)
      DIMENSION PH(NR2,NC), XLP(NR2,NC)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      DO 1 J=1,NR2
      DO 1 K=1,NC
    1 XLP(J,K)=0.0
      DO 6 K=NFCI,NCM1
      N1=NFR(K)
      IF (N1-NR1M1) 2,2,6
C     COMPUTE LAPLACIANS INSIDE CIRCLE AND ABOVE GROUND
    2 DO 5 J=N1,NR1M1
      JM=J-1
      JP=J+1
      KM=K-1
      KP=K+1
      IF (K-NCM1) 4,3,4
    3 GAMMA=XWF-(K-NCLB)*DELX
      S1=(PH(JM,K)+PH(JP,K)-2.*PH(J,K))/DELXSQ
      S2=2.*(GAMMA*PH(J,KM)-PH(J,K)*(DELX+GAMMA))/(DELX*GAMMA
     1*(GAMMA+DELX))
      XLP(J,K)=(S1+S2)*DELXSQ
      GO TO 5
    4 CONTINUE
      TMP1=PH(JM,KM)+PH(JM,KP)+PH(JP,KM)+PH(JP,KP)
      TMP2=PH(J,KM)+PH(J,KP)+PH(JM,K)+PH(JP,K)
      XLP(J,K)=(TMP1+4.*(TMP2-5.*PH(J,K)))/6.
    5 CONTINUE
    6 CONTINUE
C     COMPUTE LAPLACIANS ALONG AIR GROUND INTERFACE
      IF (MXLP1A2-NC) 7,15,15
    7 DO 11 K=MXLP1A2,NCM1
      T1=(PH(NR1,K-1)+PH(NR1P1,K-1))/2.
      T2=(PH(NR1,K)+PH(NR1P1,K))/2.
      T3=(PH(NR1,K+1)+PH(NR1P1,K+1))/2.
      IF (K-NCM1) 9,8,9
    8 GAMMA=XWF-(K-NCLB)*DELX
      S1=(PH(NR1M1,K)+PH(NR1P2,K)-2.*T2)/DELXSQ
      S2=2.*(GAMMA*T1-T2*(DELX+GAMMA))/(DELX*GAMMA*(GAMMA+
     1DELX))
```

```
      TMP3=(S1+S2)*DELXSQ
      GO TO 10
9     CONTINUE
      TMP1=PH(NR1M1,K-1)+PH(NR1M1,K+1)+PH(NR1P2,K-1)+PH(NR1P2,
     1K+1)
      TMP2=T1+T3+PH(NR1M1,K)+PH(NR1P2,K)
      TMP3=(TMP1+4.*(TMP2-5.*T2))/6.
10    XLP(NR1,K)=TMP3
11    XLP(NR1P1,K)=TMP3
C     COMPUTE LAPLACIANS BELOW GROUND
      IF  (MXLP1A2-NLCLI)  12,12,15
12    DO 14 K=MXLP1A2,NLCLI
      N2=NLR(K)
      DO 13 J=NR1P2,N2
      JM=J-1
      JP=J+1
      KM=K-1
      KP=K+1
      TMP1=PH(JM,KM)+PH(JM,KP)+PH(JP,KM)+PH(JP,KP)
      TMP2=PH(J,KM)+PH(J,KP)+PH(JM,K)+PH(JP,K)
13    XLP(J,K)=(TMP1+4.*(TMP2-5.*PH(J,K)))/6.
14    CONTINUE
15    CONTINUE
      RETURN
      END
```

SUBROUTINE FINISH (PH, NPLTPTS)

```
      SUBROUTINE FINISH (PH,NPLTPTS)
      DIMENSION PH(NPLTPTS,6)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      COMMON /S/ NDUMPS,JS
      COMMON XX(100),YY(100),LR(100),XOUT(5,100),MPMOD
      IF (JS) 4,4,1
1     JJS=JS*5
      NDUMPS=NDUMPS+1
      BUFFER OUT (1,1) (XOUT,XOUT(JJS))
2     IF (UNIT,1) 2,4,3,3
3     PRINT 18
4     CONTINUE
      DO 17 M=1,NPTS
      REWIND 1
      PRINT 19, M,XX(M),YY(M)
      JP=0
      BSUM=0.0
      PRINT 21
      DO 16 N=1,NDUMPS
      IF (N-NDUMPS) 7,5,7
5     IF (JS) 7,7,6
6     NNIND=JS
      BUFFER IN (1,1) (XOUT,XOUT(JJS))
      GO TO 8
7     NNIND=100
      BUFFER IN (1,1) (XOUT,XOUT(500))
8     IF (UNIT,1) 8,10,9,9
9     PRINT 20
      STOP
10    DO 15 J=1,NNIND
      NPL=XOUT(1,J)
      IF (NPL-M) 15,11,15
11    D=XX(M)**2+YY(M)**2
      X2=XOUT(2,J)
      X3=XOUT(3,J)
      X4=XOUT(4,J)
      X5=XOUT(5,J)
      X6=SQRTF(X3*X3+X4*X4)
```

```
      R=(VLA*X2)**2
      IF (R-D) 12,12,13
12    B1=BSUM
      BSUM=X4/VLA
      X5=(BSUM-B1)/DELTIME
      GO TO 14
13    BSUM=BSUM+XOUT(5,J)*DELTIME*MPMOD
14    PRINT 22,  X2,X3,X4,X6,BSUM,X5
      JP=JP+1
      PH(JP,1)=X2
      PH(JP,2)=X3
      PH(JP,3)=X4
      PH(JP,4)=X6
      PH(JP,5)=BSUM
      PH(JP,6)=X5
15    CONTINUE
16    CONTINUE
17    CONTINUE
      RETURN
C
18    FORMAT (35H          CANT WRITE ON OUTPUT FILE)
19    FORMAT (1H1,18H VALUES FOR POINT ,I2,6H,  AT (,F6.2,1H,,
     1F6.2,1H),//)
20    FORMAT (34H          CANT READ THE INPUT TAPE)
21    FORMAT (113H      TIME                  EX                 EY
     1          E-TOTAL                B                B-DOT,//)
22    FORMAT (2X,E12.4,5E20.4)
      END
```

SUBROUTINE BNDRY (PH, NR2, NC)

```
      SUBROUTINE BNDRY (PH,NR2,NC)
      DIMENSION PH(NR2,NC)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      DO 1 J=2,NR2M1
1     PH(J,1)=2.*PH(J,2)-PH(J,3)
      IF (NCLB) 5,5,2
2     DO 3 K=1,NCLB
3     PH(NR1,K)=PH(NR1M1,K)
      DO 4 J=NR1P1,NR2M1
4     PH(J,NCLB)=PH(J,NCLB+1)
5     DO 6 K=1,NCM1
6     PH(1,K)=PH(2,K)
      RETURN
      END
```

SUBROUTINE PHI (PH, PHM, XLP, NR2, NC)

```
      SUBROUTINE PHI (PH,PHM,XLP,NR2,NC)
      DIMENSION PH(NR2,NC), PHM(NR2,NC), XLP(NR2,NC)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /ITP/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
C     UPDATE PHI VALUES LYING ABOVE GROUND AND INSIDE CIRCLE.
      DO 1 K=NFCI,NCM1
      N1=NFR(K)
      DO 1 J=N1,NR1
      NSB=(K-1)*NR2+J
      TMP=C1A*XLP(NSB)+C2A*PH(NSB)+C3A*PHM(NSB)
      PHM(NSB)=PH(NSB)
1     PH(NSB)=TMP
C     UPDATE PHI VALUES ABOVE GROUND OUTSIDE CIRCLE FOR
C     COLUMNS IN VICINITY OF CIRCLE
      K1=MAX0(2,NFCI-2)
      DO 13 K=K1,NCM1
      J1=NFR(K)-1
      X=(K-NCLB)*DELX
      IF (J1-2) 13,2,2
2     EMDTR=EYMAX/TRISE
      XDC=X/VLA
      C2T2=(VLA*TNOW)**2
      IF (TNOW-XDC-TRISE) 3,4,4
3     VAL=EMDTR*((X*X+C2T2)/(2.*VLA)-TNOW*X)
      GO TO 5
4     VAL=EYMAX*(-X+VLA*(TNOW-TRISE/2.))
5     IF (VLA*TLAST-X) 6,6,7
6     VAL2=0.0
      GO TO 11
7     C2T2=(VLA*TLAST)**2
      IF (TLAST-XDC-TRISE) 8,9,9
8     VAL2=EMDTR*((X*X+C2T2)/(2.*VLA)-TLAST*X)
      GO TO 10
9     VAL2=EYMAX*(-X+VLA*(TLAST-TRISE/2.))
10    CONTINUE
11    CONTINUE
      DO 12 J=2,J1
      NSB=(K-1)*NR2+J
      PHM(NSB)=VAL2
```

```
12  PH(NSB)=VAL
13  CONTINUE
C   UPDATE PHI VALUES BELOW GROUND
    DO 15 K=MXLP1A2,NCM1
    N2=NLR(K)
    DO 14 J=NR1P1,N2
    NSB=(K-1)*NR2+J
    TMP=C1G*XLP(NSB)+C2G*PH(NSB)+C3G*PHM(NSB)
    PHM(NSB)=PH(NSB)
14  PH(NSB)=TMP
15  CONTINUE
    RETURN
    END
```

SUBROUTINE UPTAPE (PH, XLP, NR, NC)

```
      SUBROUTINE UPTAPE (PH,XLP,NR,NC)
      DIMENSION PH(NR,NC), XLP(NR,NC)
      COMMON /UPDATE/ XWF,XLC,TLAST,TNOW,NCLB,NFCI,MXLP1A2
      COMMON /SET/ DXPTS,DELTIME,NLCLI,NCM1,NR1M1,NR1P1,
     1NR1P2,NR2M1,DELXSQ,EPSILON
      COMMON /IPT/ NTSPSS,NCYCLES,DELX,VLA,VLG,EYMAX,NR1,
     1TRISE,NPTS,TRUN
      COMMON /ROWS/ NFR(1001),NLR(1001)
      COMMON /COEF/ C1A,C2A,C3A,C1G,C2G,C3G
      COMMON /S/ NDUMPS,JS
      COMMON XX(100),YY(100),LR(100),XOUT(5,100),MPMOD
      DATA (JS=0),(NDUMPS=0)
      DO 13 J=1,NPTS
      DCOL=(ABSF(XX(J))-XLC)/DELX+1.00001
      NCOL=DCOL
      IF (DCOL-NCOL-.5) 2,2,1
1     NCOL=NCOL+1
2     IF (NCOL-1) 13,13,3
3     IF (NCOL-NC) 4,13,13
4     JS=JS+1
      DIV=DELX
      IF (NCOL-NCM1) 6,5,6
5     XXD=(NCOL-1)*DELX+XLC
      DIV=XWF-XXD
6     CONTINUE
      XOUT(1,JS)=J
      XOUT(2,JS)=TNOW
      JR=LR(J)
      IF (JR-NR1) 7,7,8
7     XOUT(3,JS)=(PH(JR-1,NCOL)-PH(JR,NCOL))/DELX
      GO TO 9
8     XOUT(3,JS)=(PH(JR,NCOL)-PH(JR+1,NCOL))/DELX
9     XOUT(4,JS)=(-(PH(JR,NCOL+1)-PH(JR,NCOL)))/DIV
      XOUT(5,JS)=XLP(JR,NCOL)/DELXSQ
      IF (JS-100) 13,10,10
10    JS=0
      NDUMPS=NDUMPS+1
      BUFFER OUT (1,1) (XOUT,XOUT(500))
11    IF (UNIT,1) 11,13,12,12
12    PRINT 14
      STOP
13    CONTINUE
      RETURN
C
14    FORMAT (23H    CANT WRITE PLOT TAPE )
      END
```