NBSIR 84-2835

Free-Space Propagation of Light Pulses

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratories
Center for Manufacturing Engineering
Washington, DC 20234

May 1984

U.S. DEPARTMENT OF COMMERCE
ATIONAL BUREAU OF STANDARDS

NBSIR 84-2835

FREE-SPACE PROPAGATION OF LIGHT PULSES

Egon Marx

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratories
Center for Manufacturing Engineering
Washington, DC 20234

May 1984

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

Abstract

A transient electromagnetic field in free space is completely specified when the initial values of the electric and magnetic fields are given. Green's function for the scalar wave equation can then be used to find the field at later times. A group of computer programs that implement these equations and process the output are presented in this report.

Key words: computer programs; conservation of energy and momentum; electromagnetic field propagation; finite light pulses; graphic displays of energy density; transient electromagnetic fields.

Table of Contents

1. Introduction

In order to compute transient electromagnetic fields scattered by conductors or dielectrics, as required for the Wave Optics program at the National Bureau of Standards, we have to determine the incident fields specified by their initial values. These incident fields are free fields that obey Maxwell's equation without sources in all of space, disregarding the scatterer.

The programs described in this report allow us to compute the free fields at arbitrary points and times from given initial values of the fields.

A simple example is the plane wave: the fields are constant over planes perpendicular to a direction of propagation $\hat{n}$. The initial electric field, $\vec{E}_0(\zeta)$, is a vector field perpendicular to $\hat{n}$ that is a function of a single variable; the free electric field at other times is then given by

$$\vec{E}(\vec{x},t) = \vec{E}_0(\zeta - ct) \tag{1}$$

where c is the speed of light in vacuum and

$$\zeta = \vec{x} \cdot \hat{n}, \quad \hat{n} \cdot \vec{E}_0 = 0. \tag{2}$$

The magnetic field of this plane wave is then given by

$$c\vec{B}(\vec{x},t) = \hat{n} \times \vec{E}(\vec{x},t). \tag{3}$$

For fields other than plane waves, the incident field can be computed by integrations from the initial values of $\vec{E}$ and $\vec{B}$. The

solution of the free-field Maxwell equations is shown in section 2, computer programs developed to do the numerical integrations are described in section 3, and a few sample outputs are shown in section 4. The listings of the computer programs are given in an appendix.

2. Propagation of Localized Pulses

We have shown [1]* how the retarded Green function for the scalar wave equation,

$$G_R^{(0)}(x,x') = \frac{\delta(t - t' - |\vec{x} - \vec{x}'|/c)}{4\pi|\vec{x} - \vec{x}'|}, \qquad (4)$$

can be used to express the incident electromagnetic fields in terms of their initial values given at time t = 0 for all of space. The symbol x stands for the four-vector with components x_μ, $\mu = 0,1,2,3$, that is, $x = (ct,\vec{x})$, and $\delta(t)$ is the Dirac delta function. The resulting integrals are

$$\vec{E}(x) = -\int_V dV' \Big[(1/c^2)\vec{E}_0(\vec{x}')\partial G_R^{(0)}(x,x')/\partial t'$$

$$- \vec{B}_0(\vec{x}') \times \nabla' G_R^{(0)}(x,x')\Big]_{t'=0}, \qquad (5)$$

---

*Numbers in square brackets indicate the literature references at the end of this report.

$$\vec{B}(x) = -(1/c^2)\int_V dV' \left[\vec{E}_0(\vec{x}') \times \nabla' G_R^{(0)}(x,x')\right.$$

$$\left. + \vec{B}_0(\vec{x}')\partial G_R^{(0)}(x,x')/\partial t'\right]_{t'=0}, \qquad (6)$$

where ∇' is the gradient operator with respect to the variable $\vec{x}'$. We substitute the Green function (4) into eqs. (5) and (6) to obtain

$$\vec{E}(x) = \int_V dV' \left[\frac{1}{c^2}\vec{E}_0(\vec{x}')\frac{\delta'(t - R/c)}{4\pi R}\right.$$

$$\left. + \vec{B}_0(\vec{x}') \times \left(\frac{\delta'(t - R/c)}{4\pi Rc} + \frac{\delta(t - R/c)}{4\pi R^2}\right)\hat{R}\right], \qquad (7)$$

$$\vec{B}(x) = -\frac{1}{c^2}\int_V dV' \left[\vec{E}_0(\vec{x}') \times \left(\frac{\delta'(t - R/c)}{4\pi Rc} + \frac{\delta(t - R/c)}{4\pi R^2}\right)\hat{R}\right.$$

$$\left. - \vec{B}_0(\vec{x}')\frac{\delta'(t - R/c)}{4\pi R}\right], \qquad (8)$$

where $\vec{R} = \vec{x} - \vec{x}'$, $R = |\vec{R}|$, and $\hat{R} = \vec{R}/R$. We integrate over R and are left with integrals over a sphere of radius R = ct centered at the field point $\vec{x}$, known as the information-collecting sphere. We can use the solid angle from $\vec{x}$ and express the fields as

$$\vec{E}(x) = \frac{1}{4\pi}\int d\Omega\left[\vec{E}_0 - \vec{R}\cdot\nabla'\vec{E}_0 + (2c\vec{B}_0 - \vec{R}\cdot\nabla' c\vec{B}_0) \times \vec{R}\right]_{R = ct} , \quad (9)$$

$$c\vec{B}(x) = \frac{1}{4\pi}\int d\Omega\left[c\vec{B}_0 - \vec{R}\cdot\nabla' c\vec{B}_0 - (2\vec{E}_0 - \vec{R}\cdot\nabla'\vec{E}_0) \times \vec{R}\right]_{R = ct} . \quad (10)$$

The fields $\vec{E}_0$ and $\vec{B}_0$ are arbitrary solenoidal fields, that is, they satisfy the constraint equations

$$\nabla \cdot \vec{E}_0(\vec{x}) = 0, \quad (11)$$

$$\nabla \cdot \vec{B}_0(\vec{x}) = 0. \quad (12)$$

A simple way to find a solenoidal field is to take the curl of another vector field.

We would like the field to be spatially localized, but otherwise to look somewhat like a plane wave propagating in the z-direction. We restrict the initial fields to the region $z < 0$ and we assume that they decay essentially like a gaussian in the x- and y-directions. We choose

$$\vec{E}_0(\vec{x}) = A \exp\left[- \kappa(x^2 + y^2)\right](y\hat{i} - x\hat{j})\Phi'(z), \quad (13)$$

$$c\vec{B}_0(\vec{x}) = A \exp\left[- \kappa(x^2 + y^2)\right]\Big\{(x\hat{i} + y\hat{j})\Phi'(z)$$

$$- 2\hat{k}\left[1 - \kappa(x^2 + y^2)\right]\Phi(z)\Big\}, \qquad (14)$$

where Φ is a function that vanishes for $z \geq 0$ and has a continuous first derivative. These expressions can be used to compute the dyadics that are the gradients of $\vec{E}_0$ and $\vec{B}_0$.

The initial values of the Poynting vector and energy density are proportional to

$$\vec{E}_0 \times c\vec{B}_0 = A^2 \exp\left[- 2\kappa(x^2 + y^2)\right]\Big\{2\left[1 - \kappa(x^2 + y^2)\right]\Phi\Phi'(x\hat{i} + y\hat{j})$$

$$+ (x^2 + y^2)\Phi'^2\hat{k}\Big\}. \qquad (15)$$

$$\vec{E}_0^2 + c^2\vec{B}_0^2 = A^2 \exp\left[- 2\kappa(x^2 + y^2)\right]\Big\{2(x^2 + y^2)\Phi'^2$$

$$+ 4\left[1 - \kappa(x^2 + y^2)\right]^2\Phi^2\Big\}. \qquad (16)$$

This energy density is invariant under rotations about the z-axis, and the Poynting vector has a component in the positive z-direction and one radial component (perpendicular to the z-axis). For a light pulse, Φ has the form

$$\Phi(z) = \gamma(z)\sin(kz), \qquad (17)$$

where k is the wave number of the light produced by a laser.

3. Computer Programs

A computer program named BALL allows us to calculate the fields by performing the integrations in eqs. (9) and (10). This program calls a subroutine ICS where the unit sphere is covered by the appropriate number of patches required to carry out the integrations, and a subroutine POINT that is used to step through the field points. The output of BALL is stored in a file, that is then used as input to separate programs that present the results in graphic form. These programs are BALPLT, which produces a plot of the energy density as a function of the field points, and BALDSK, which prepares a binary interpolated file that can be shown on the monitor of our group's image processing facility or photographed by a special computer-driven camera. An additional program, BALCHK, can be used to perform a check on the accuracy of the numerical calculations by means of the conservation of energy and momentum. These programs are listed in the appendix.

3.1 Program BALL

We use this program to do the integrations required to find the fields at a time t and at a point $\vec{x}$ for a given set of initial fields. In the program shown in the appendix the pulse described in eqs. (13) and (14) has a shape given by

$$\Phi(z) = [\exp(\alpha z) - \exp(\beta z)]\sin(kz)\Theta(-z), \qquad (18)$$

if a positive k is specified, or

$$\psi(z) = z^2[\exp(\alpha z) - \exp(\beta z)]\theta(-z) \qquad (19)$$

if k vanishes. The unit step function θ makes the pulse vanish for positive z at t = 0. For the pulsed lasers used at present, the underlying fields oscillate maybe one hundred times for the duration of the pulse, but this number is being reduced in ongoing experimental research programs. Since Maxwell's equations in free space are invariant under changes of scale in space and time, the same calculations can be applied to other electromagnetic fields that may be true pulses, without an underlying sinusoidal field.

The parameters of the pulse, the region over which the field points range, and other parameters of the calculation are given in an input file BALIN.XXX, where XXX is the qualifier that identifies a particular run. The initial values of t and $\vec{x}$ as well as the increment of any one or two of these variables are specified in this file. The output file BALOUT.XXX contains the values of the components of $\vec{E}$ and $c\vec{B}$.

The integration is performed by summing the contributions to the field components of the integrands calculated at the center of each patch multiplied by the area of the patch.

Different versions of the program have been tried. Some of the variations change the way the patches on the sphere are determined (a simpler subroutine can be used at the expense of computing time or accuracy), start from other initial fields, use double precision variables to do the computations (improves the

accuracy of the results for times significantly larger than the duration of the initial pulse), or save only the energy density instead of the six field components (saves disk space). The program contains a provision that allows us to restart the calculation without much loss of time if there is an interruption.

3.2 Subroutine POINT

This subroutine is used to read the information on the region of space and time where the fields are to be calculated, and to generate successive values of $\vec{x}$ and t.

In its present form, the program requires either that one variable change over a given range, where the given number of points is to be distributed uniformly, or that two variables change, with points distributed over a rectangular grid uniformly in each direction. A simple variation allows the program to compute the fields at points located diagonally in the xy-plane by varying both x and y simultaneously.

3.3 Subroutine ICS

The values of the initial fields that contribute to the fields at a point $\vec{x}$ at time t are all on the Information-Collecting Sphere (ICS) centered at the field point $\vec{x}$ and with a radius ct.

If the initial fields have the form given by eqs. (13) and (14), with a $\Phi(z)$ specified in (18) or (19), the integrands that contribute significantly to the result are confined to a finite

cylinder whose axis is the z-axis, whose radius is a function of the decay constant κ of the gaussian, and which is limited by the planes $z = 0$ and $z = z_m$, where z_m is a function of the decay constant α of the double exponential. In particular, we have set $\kappa(x^2 + y^2)$ and αz equal to a maximum value given by the variable CUT in the input file. We thus have to find the regions of the ICS that lie inside the cylinder. Depending on the radius of the sphere and the location of its center, the whole spherical surface may be included, one or two regions may be inside the cylinder, or none at all. Once the nature and the limits of these regions are determined, each is covered by patches determined by first dividing the regions into strips by circles of constant polar angle θ and then dividing each strip or partial strip into patches by circles of constant azimuthal angle ϕ. The sizes of these patches are determined in two different ways. A length given through the input variable DPATCH is used to determine the magnitudes of the sides of the patches except when this length is comparable to the radius of the sphere. If this is the case, there may be too few patches to assure accurate values for the integral, and a maximum size of the patches is determined by dividing the sphere into N1 strips and dividing the equatorial strip into N2 patches, N1 and N2 being values supplied also through the input file.

After the above preliminary computations are done for a given time and field point, successive calls to the subroutine ICS return values of the components of the unit vector to the center of a patch and the solid angle subtended by the patch,

given by

$$\Delta\Omega = 2\,\Delta\phi\,\sin\theta\,\sin(\tfrac{1}{2}\Delta\theta). \qquad (20)$$

We choose a circular cap of angular diameter equal to $\Delta\theta$ as the patch at either of the poles; the corresponding solid angle is

$$\Delta\Omega = 4\pi\,\sin^2(\tfrac{1}{4}\Delta\theta). \qquad (21)$$

These values might be approximated by $\sin\theta\Delta\theta\Delta\phi$ and $\pi(\tfrac{1}{2}\Delta\theta)^2$, respectively, if the accuracy needed allows this substitution. We have not determined precisely when this is allowable, but we found that, in a particular case, the switch made a significant difference.

Subroutine ICS calls the subroutines PATCH to determine the components of the unit vector for a patch on a partial strip, RPATCH to do likewise for a full strip, and FILL to transfer these values and the solid angle to COMMON. These subroutines are listed together with ICS.

3.4 Program BALPLT

We use this program to produce plots of the energy density of the electromagnetic field as a function of the one or two variables that change over a given interval. The input comes from the files BALIN.XXX and BALOUT.XXX, and the output is directed to a plotter or printer/plotter.

When only one variable changes, the data on the dependent and independent variables, as well as labels, are passed to the subroutine DRAW4, a general purpose plotting subprogram described in reference [2]. When two variables change, the number of

points in either variable is reduced to a maximum of 100 if necessary, and then the data are passed to subroutine PLOT3D, which produces three-dimensional plots and is described in reference [3].

The nature of the Green function for the wave equation is such that the fields propagate exactly with the speed of light. Thus, the field in the xz-plane at a time t is confined to a ring of radius ct and of a width equal to the size of the initial pulse. When the grid of points where the fields are calculated and plotted is superimposed on this ring, the appearance of the resulting three-dimensional plot is that of a large number of isolated peaks if the distance between the points is comparable to the width of the ring. This problem can be minimized by choosing a density of points that is high enough.

This program can be easily adapted to show all components of the electromagnetic field, or to show the energy density when only this quantity is saved in the file BALOUT.XXX.

3.5 Program BALDSK

This program was developed to prepare a raster file of values proportional to the energy density of the electromagnetic field that can be shown on the monitor of the image processing facility or photographed. We assume that the time t is fixed, and that the grid extends over a range of values of z and positive values of x. We first interpolate in the x-direction to 256 points and then reflect this set of values about the z-axis to take advantage of the symmetry in the field, and then we

interpolate to 512 such records in the z-direction. The intensity of a single graph is scaled to values between 0 and 255, and the maximum intensity of a particular graph in a series for varying values of t is scaled to fit a range between 150 and 255.

3.6 Program BALCHK

To use BALCHK, the fields have to be computed for a fixed value of t and for values of z and positive x that cover the region where the fields are significantly different from zero. BALCHK performs a numerical integration to compute the values of the total energy and momentum of the pulse; these values allow us to check the validity of the calculations by comparing them to the initial values obtained from the analytical expressions (15), (16), and (18) or (19). The x- and y-components of the momentum vanish; the z-component of the momentum and the energy density are proportional to

$$\int dV(\vec{E} \times c\vec{B})_z = \pi A^2 \frac{1}{4\kappa^2}\left[\frac{1}{4\alpha^3} + \frac{8(\alpha^2 - 4\alpha\beta + \beta^2)}{(\alpha + \beta)^5} + \frac{1}{4\beta^3}\right], \qquad (22)$$

$$\int dV(\vec{E}_0^2 + c^2\vec{B}_0^2) = 2\pi A^2\left[\frac{1}{4\kappa^2}\left(\frac{1}{4\alpha^3} + \frac{8(\alpha^2 - 4\alpha\beta + \beta^2)}{(\alpha + \beta)^5} + \frac{1}{4\beta^3}\right)\right.$$

$$\left. + \frac{3}{8\kappa}\left(\frac{1}{\alpha^5} - \frac{64}{(\alpha + \beta)^5} + \frac{1}{\beta^5}\right)\right]. \qquad (23)$$

To compute these integrals numerically at later times, we divide the space where we compute the fields into tori of square cross sections based on the grid in the xz-plane, and we multiply the volume of each torus by the average of the energy density or Poynting vector as computed at the four corners of the square in the grid.

Since all the contributions to the energy are positive, if the grid is too coarse (that is, if it misses much of the ring where the fields are concentrated), the agreement can be poor (that is, the discrepancy can be larger than a few percent) without the values of the field being incorrect. In these cases, the conservation of momentum generally shows better agreement than the conservation of energy.

The initial values of the z-component of the Poynting vector (15) are positive, so that the pulse as a whole moves in the positive z-direction.

If only the energy density of the field is saved, we can still check for the conservation of energy.

4. Sample Outputs

We use a printer/plotter to obtain the outputs from the program BALPLT. In fig. 1 we show three-dimensional plots of the energy density of the simple pulse given by eq. (19) as functions of x and z at the initial time and after 1.2×10^{-12} s, which corresponds to a distance of 3.6×10^{-4} m. The pulse moves mainly forward in the z-direction and to the sides, and the peak intensity decreases from 3.3×10^{-18} to 5.9×10^{-20} in arbitrary

units.

Figure 2 shows the same features for a modulated pulse specified by eq. (18). Although the wave number of the modulation is not much bigger than the size of the pulse ($k = 1\times10^{5}\ m^{-1}$ and $\alpha = 2\times10^{4}\ m^{-1}$), the shape of this pulse changes little over the chosen period of time, the motion is mostly in the z-direction and the maximum intensity decreases from 0.74 to 0.46 only. The superposition of the plotting grid on the ridges that belong to the energy density function produces additional peaks in the plot. Figure 3 shows the x-, y-, and z-components of the fields $\vec{E}$ and $c\vec{B}$ for the unmodulated pulse. They are shown at the point (1×10^{-4}, 1×10^{-4}, 0) as functions of time between 0 and 1.2×10^{-12} s, and the same fields at the time 1.2×10^{-12} s for $y = 0$ and $z = 2\times10^{-4}$ as a function of x between the values -6×10^{-4} and 6×10^{-4} m (this plot exhibits the symmetry about the z-axis: E_2 and B_1 are antisymmetric while B_3 is symmetric).

Figure 4 shows two photographs taken with a special camera of an image produced from a finer grained computation of the energy density shown in fig. 1, with the fields calculated on a 256×256 grid. This camera can produce a moving picture by photographing a sequence of frames.

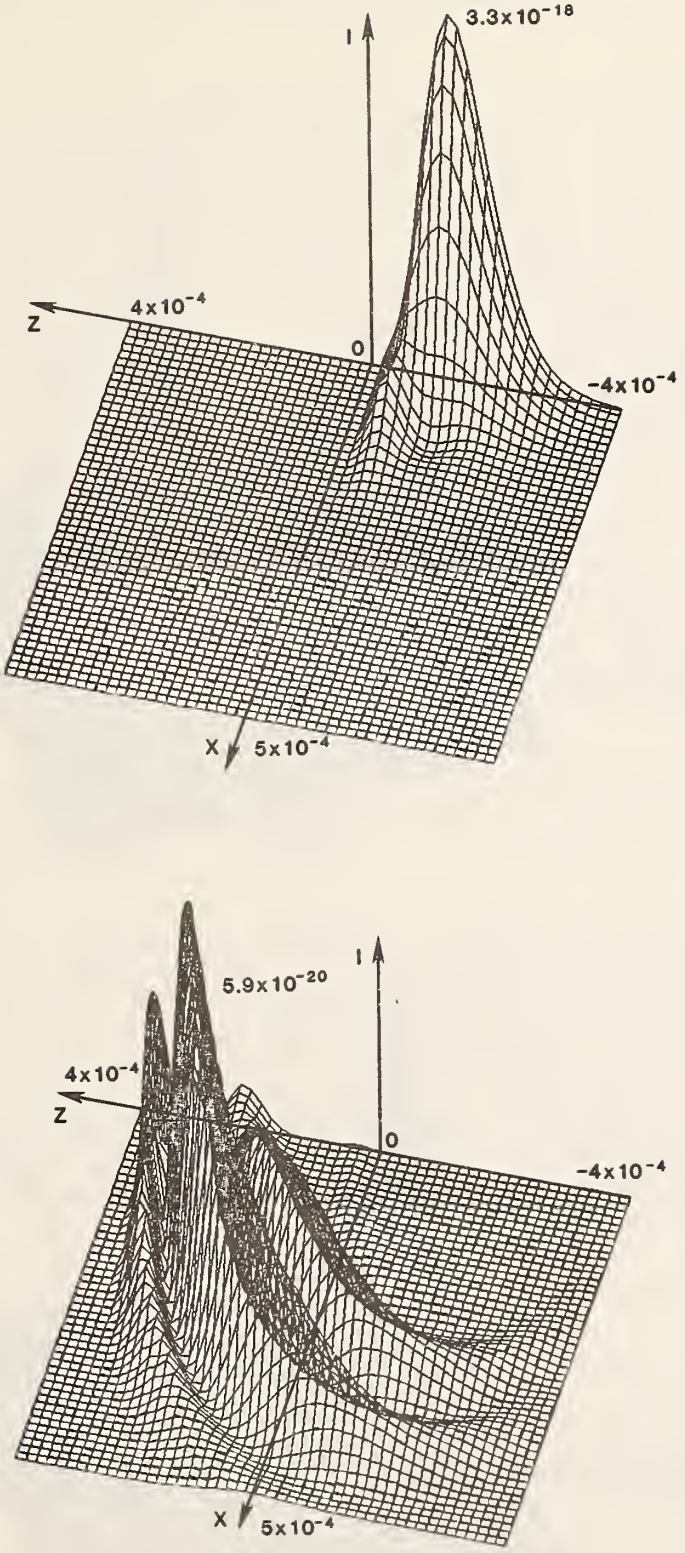

Fig. 1. Energy density of the simple pulse as a function of x and z at the initial time and after 1.2×10^{-12} s. There is cylindrical symmetry about the z-axis. The scale of distances is shown in meters and the maximum intensities in arbitrary units.

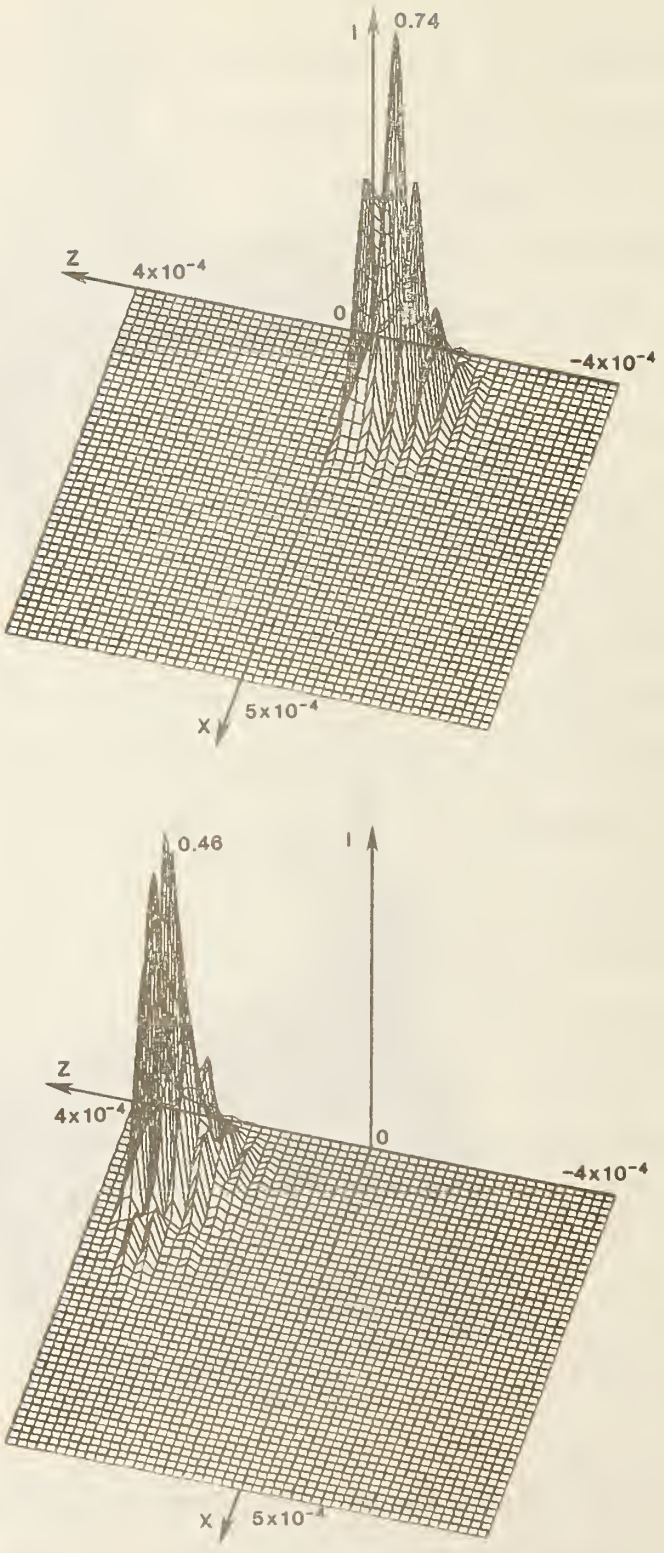

Fig. 2. Same as fig. 1 for a modulated pulse.

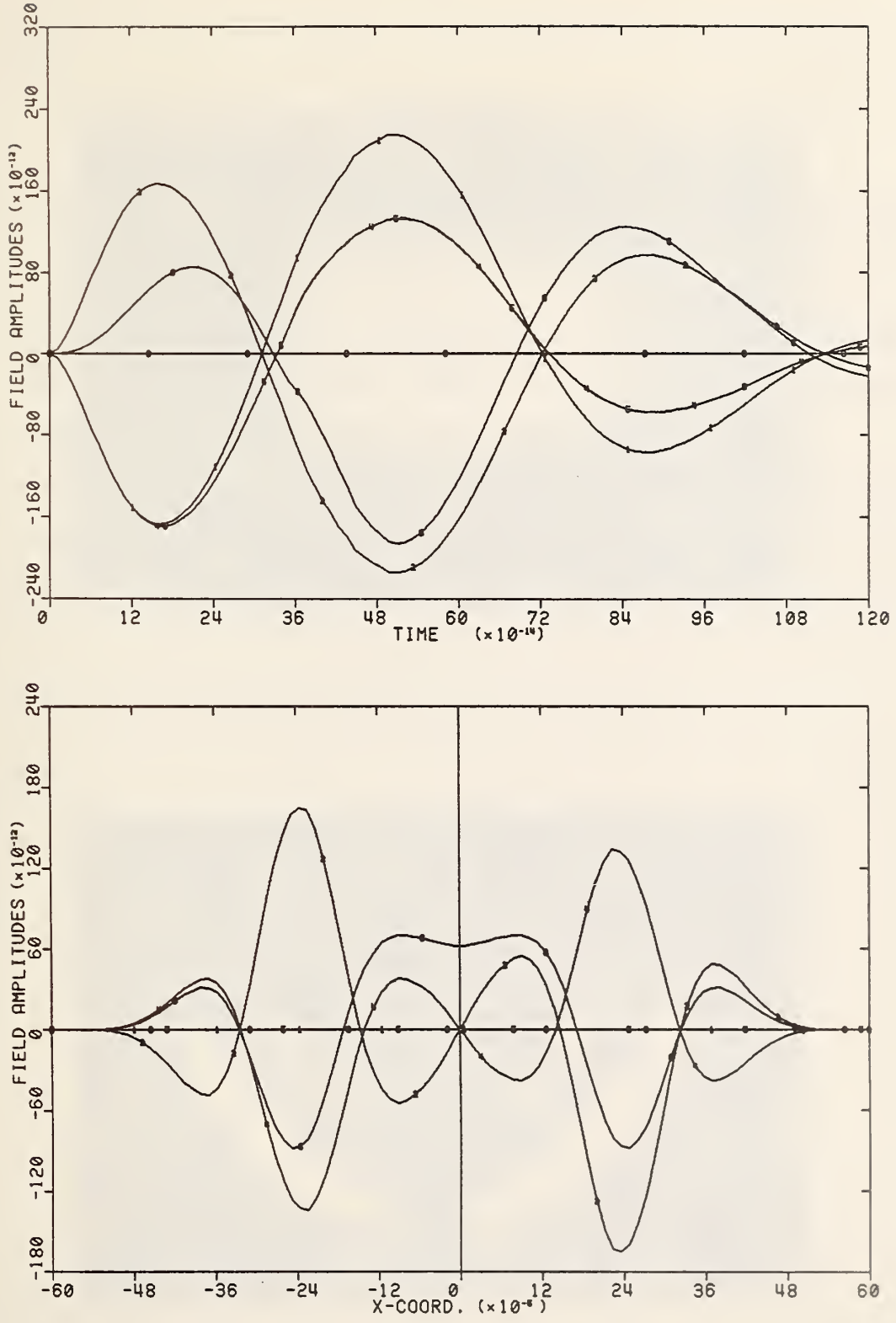

Fig. 3. Plots of the components of the electromagnetic fields; the curves are labeled from 1 to 6 to identify E_1, E_2, E_3, B_1, B_2, and B_3, respectively. Top: functions of time at the point $(1\times10^{-4}, 1\times10^{-4}, 0)$. Bottom: functions of x at a time $t = 1.2\times10^{-12}$ for fixed $y = 0$ and $z = 2\times10^{-4}$ m.

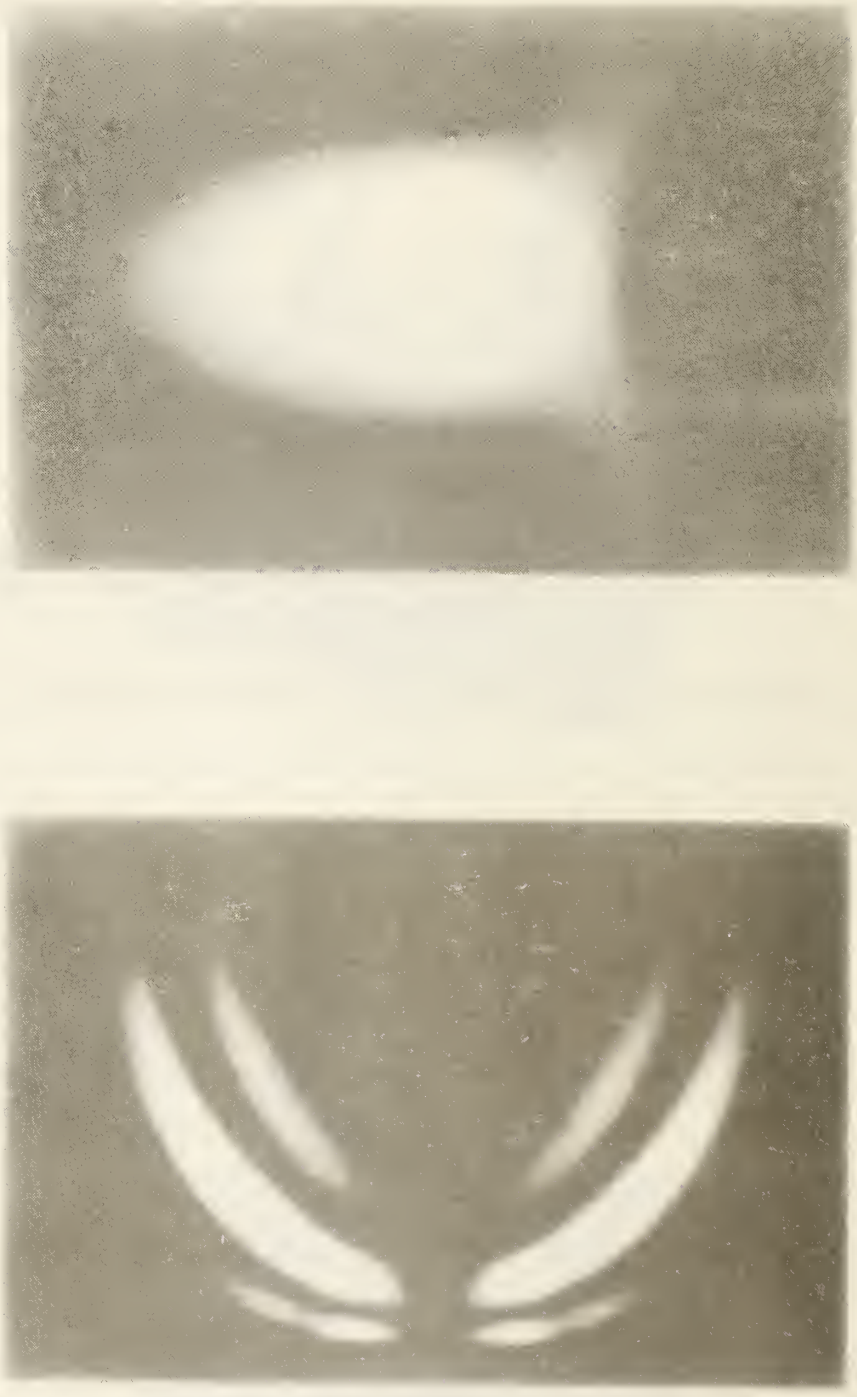

Fig 4. Photographs of energy densities shown in fig. 1. The pulse is propagating mainly to the left. Symmetry is used to extend the graph to negative values of x.

References

[1] Marx, E. Integral Equations for Transient Electromagnetic Fields. NBS Technical Note 1157, February 1982.

[2] Noon, T. V. User's Manual for Modular Analysis-Package Libraries ANAPAC and TRANL. Harry Diamond Laboratories Technical Report TR-1782, November 1976.

[3] Nelson, D. L. Perspective Plotting of Two-Dimensional Arrays --- PLOT3D, a Computer Program for a Digital Plotter, University of Maryland Technical Report 553, March 1966.

Appendix

In this appendix we provide listings of the programs and subroutines BALL, POINT, ICS, BALPLT, BALDSK, and BALCHK.

These programs use some general purpose software provided with the Interdata computer system or developed for plotting. We do not discuss or list these subroutines here; they are SYSIO, PLOT, SYMBOL, DRAW4, and PLOT3D.

```
      PROGRAM                    BALL
C
C     MICRO AND OPTICAL METROLOGY GROUP
C     NATIONAL BUREAU OF STANDARDS
C
C     EGON MARX       11/29/83
C
C     THIS PROGRAM IS USED TO COMPUTE THE FIELDS OF A SPATIALLY LOCALIZED
C     PULSE AT A TIME T AND AT PRESCRIBED FIELD POINTS WHEN THE
C     INITIAL FIELDS ARE GIVEN AS SOLENOIDAL FUNCTIONS OF A VECTOR
C     VARIABLE (X,Y,Z).
C
C     R1,R2,R3 = COMPONENTS OF THE UNIT VECTOR TO A PATCH
C     SOLANG = SOLID ANGLES SUBTENDED BY A PATCH
C     EL1,EL2,EL3 = ARRAYS TO STORE COMPONENTS OF THE ELECTRIC FIELD
C     CBM1,CBM2,CBM3 = ARRAYS TO STORE COMPONENTS OF THE MAGNETIC FIELD
C     CAPPA = DECAY CONSTANT FOR GAUSSIAN IN X- AND Y-DIRECTIONS
C     CUT = CUTOFF PARAMETER FOR INTEGRAND
C     ALPHA = RISE CONSTANT FOR DOUBLE EXPONENTIAL IN Z-DIRECTION
C     BETA = DECAY CONSTANT FOR DOUBLE EXPONENTIAL IN Z-DIRECTION
C     K = (REAL) WAVE NUMBER FOR MODULATION IN Z-DIRECTION
C         (NO MODULATION FOR K = 0.)
C     DPATCH = APPROXIMATE SIZE OF PATCH
C     NI, NJ = NUMBER OF POINTS IN GRID FOR CALCULATION
C
      DIMENSION EL1(512),EL2(512),EL3(512),CBM1(512),CBM2(512),CBM3(512)
      LOGICAL EX
      REAL K,KZP
      CHARACTER QAL*3,FL1*9,FL2*10
      COMMON /RCOM/ R,R1,R2,R3,SOLANG
      COMMON /ICOM/ N1,N2,NX
      COMMON /PARAM/ ALPHA,BETA,CAPPA,CUT,DPATCH,PI,TOPI,PITO,FOPI,C
C
C     INITIALIZE VALUES
C
      NR=0
      C=3.E8
      PITO=2.*ATAN(1.)
      PI=2.*PITO
      TOPI=4.*PITO
      FOPI=8.*PITO
      FOPIM1=.5/TOPI
C
C     READ IN VALUES OF CONSTANTS FOR THE PULSE
C
      OPEN(1,FILE='BALL.QAL',STATUS='OLD')
      READ(1,3) QAL,IFL
      FL1='BALIN.'//QAL
      FL2='BALOUT.'//QAL
      OPEN(3,FILE=FL1,STATUS='OLD')
      READ(3,1) CAPPA,CUT,ALPHA,BETA,K,DPATCH
      READ(3,2) NI,NJ
      IF(NI.GT.512.OR.NJ.GT.512) CALL EXIT(1)
      NJ4=NJ*4
      NJ44=NJ4+4
```

```
C
C      READ FILE QUALIFIER AND CHECK FOR RESTART
C
       INQUIRE(FILE=FL2,EXIST=EX)
       IF(IFL.EQ.1.AND.EX) THEN
         OPEN(2,FILE=FL2,STATUS='OLD',ACCESS='DIRECT',
     1   FORM='UNFORMATTED',RECL=NJ4)
         INQUIRE(2,SIZE=NR)
         NX=NR/6
         IF(NX.GE.NI) THEN
           CLOSE(2)
           STOP
         ENDIF
       ELSE
         OPEN(2,FILE=FL2,STATUS='RENEW',RECL=NJ4,BLOCKSIZE=NJ44,
     1   FORM='UNFORMATTED',ACCESS='DIRECT')
       ENDIF
       CLOSE(2)
C
C      GET INITIAL PULSE DATA
C
       CALL POINT(DUM,DUM,DUM,DUM,NI,NJ,.TRUE.)
C
C      COMPUTE THE FIELDS FOR EACH TIME AT THE GIVEN POINT
C
       DO 350 I=NX+1,NI
         DO 300 J=1,NJ
C
C      FIND THE COORDINATES OF THE FIELD POINT AND THE TIME
C
           CALL POINT(X,Y,Z,T,NI,NJ,.FALSE.)
           IF(T.NE.0..OR.Z.GE.0.) GO TO 110
C
C      COMPUTATIONS FOR T=0
C
           XP=X
           YP=Y
           ZP=Z
           XP2=XP**2
           YP2=YP**2
           ZP2=ZP**2
           CXP2=CAPPA*XP2
           CYP2=CAPPA*YP2
           CX2Y2=CXP2+CYP2
           AZP=ALPHA*ZP
           IF(AZP.LT.-25..OR.CX2Y2.GT.25.) GO TO 110
           EAZ=EXP(AZP)
           EBZ=EXP(BETA*ZP)
           DEXZ=EAZ-EBZ
           DEXZP=ALPHA*EAZ-BETA*EBZ
           IF(K.EQ.0.) THEN
             FZ=ZP2*DEXZ
             FPZ=ZP2*DEXZP+2.*ZP*DEXZ
           ELSE
             KZP=K*ZP
```

```
            SKZ=SIN(KZP)
            CKZ=COS(KZP)
            FZ=DEXZ*SKZ
            FPZ=DEXZP*SKZ+DEXZ*K*CKZ
          END IF
          TCXYP=2.*CAPPA*XP*YP
          ECX2Y2=EXP(-CX2Y2)
          E1=YP*ECX2Y2*FPZ
          E2=-XP*ECX2Y2*FPZ
          E3=0.
          CB1=XP*ECX2Y2*FPZ
          CB2=YP*ECX2Y2*FPZ
          CB3=2.*(CX2Y2-1.)*ECX2Y2*FZ
          GO TO 210
110       E1=0.
          E2=0.
          E3=0.
          CB1=0.
          CB2=0.
          CB3=0.
          IF(T.EQ.0.) GO TO 210
C
C     ADD THE CONTRIBUTIONS FROM EACH PATCH ON THE UNIT SPHERE
C
120       CALL ICS(X,Y,Z,T,*200,*210)
C
C     COMPUTE THE COMPONENTS OF THE VECTOR
C     FROM THE SOURCE POINT TO THE FIELD POINT
C
          RU1=-R1
          RU2=-R2
          RU3=-R3
          SLJ=SOLANG
          RV1=R*RU1
          RV2=R*RU2
          RV3=R*RU3
C
C     COMPUTE THE COORDINATES OF THE SOURCE POINT
C
          XP=X-RV1
          YP=Y-RV2
          ZP=Z-RV3
C
C     COMPUTE THE INTEGRANDS
C
          XP2=XP**2
          YP2=YP**2
          ZP2=ZP**2
          CXP2=CAPPA*XP2
          CYP2=CAPPA*YP2
          CX2Y2=CXP2+CYP2
          AZP=ALPHA*ZP
C
C     FIND DOUBLE EXPONENTIAL AND ITS DERIVATIVES
C
```

```
            EAZ=EXP(AZP)
            EBZ=EXP(BETA*ZP)
            DEXZ=EAZ-EBZ
            DEXZP=ALPHA*EAZ-BETA*EBZ
            DEXZPP=ALPHA**2*EAZ-BETA**2*EBZ
            IF(K.EQ.0.) THEN
              FZ=ZP2*DEXZ
              FPZ=ZP2*DEXZP+2.*ZP*DEXZ
              FPPZ=ZP2*DEXZPP+4.*ZP*DEXZP+2.*DEXZ
            ELSE
              KZP=K*ZP
              SKZ=SIN(KZP)
              CKZ=COS(KZP)
              FZ=DEXZ*SKZ
              FPZ=DEXZP*SKZ+DEXZ*K*CKZ
              FPPZ=(DEXZPP-K**2*DEXZ)*SKZ+2.*DEXZP*K*CKZ
            END IF
C
C       COMPUTE FIELDS AT SOURCE POINT
C
            TCXYP=2.*CAPPA*XP*YP
            ECX2Y2=EXP(-CX2Y2)
            EO1=YP*ECX2Y2*FPZ
            EO2=-XP*ECX2Y2*FPZ
            EO3=0.
            CBO1=XP*ECX2Y2*FPZ
            CBO2=YP*ECX2Y2*FPZ
            CBO3=2.*(CX2Y2-1.)*ECX2Y2*FZ
C
C       COMPUTE GRADIENTS OF FIELDS AT SOURCE POINT
C
            DEO11=-ECX2Y2*TCXYP*FPZ
            DEO21=ECX2Y2*(1.-2.*CYP2)*FPZ
            DEO31=ECX2Y2*YP*FPPZ
            DEO12=-ECX2Y2*(1.-2.*CXP2)*FPZ
            DEO22=-DEO11
            DEO32=-ECX2Y2*XP*FPPZ
            DEO13=0.
            DEO23=0.
            DEO33=0.
            DBO11=-DEO12
            DBO21=-DEO22
            DBO31=-DEO32
            DBO12=DBO21
            DBO22=DEO21
            DBO32=DEO31
            DBO13=ECX2Y2*2.*CAPPA*XP*(2.-CX2Y2)*FZ
            DBO23=ECX2Y2*2.*CAPPA*YP*(2.-CX2Y2)*FZ
            DBO33=ECX2Y2*2.*(CX2Y2-1.)*FPZ
C
C       COMPUTE DIRECTIONAL DERIVATIVES ALONG R
C
            RDEO1=RV1*DEO11+RV2*DEO21+RV3*DEO31
            RDEO2=RV1*DEO12+RV2*DEO22+RV3*DEO32
            RDEO3=RV1*DEO13+RV2*DEO23+RV3*DEO33
```

```
          RDBO1=RV1*DBO11+RV2*DBO21+RV3*DBO31
          RDBO2=RV1*DBO12+RV2*DBO22+RV3*DBO32
          RDBO3=RV1*DBO13+RV2*DBO23+RV3*DBO33
C
C      COMPUTE VECTOR PRODUCTS
C
          EVRO1=EO2*RU3-EO3*RU2
          EVRO2=EO3*RU1-EO1*RU3
          EVRO3=EO1*RU2-EO2*RU1
          CBVRO1=CBO2*RU3-CBO3*RU2
          CBVRO2=CBO3*RU1-CBO1*RU3
          CBVRO3=CBO1*RU2-CBO2*RU1
          RDEVR1=RDEO2*RU3
          RDEVR2=-RDEO1*RU3
          RDEVR3=RDEO1*RU2-RDEO2*RU1
          RDBVR1=RDBO2*RU3-RDBO3*RU2
          RDBVR2=RDBO3*RU1-RDBO1*RU3
          RDBVR3=RDBO1*RU2-RDBO2*RU1
C
C      ADD CONTRIBUTIONS TO THE FIELDS AT THE FIELD POINT
C
          DE1=SLJ*(EO1-RDEO1+(2.*CBVRO1-RDBVR1))
          DE2=SLJ*(EO2-RDEO2+(2.*CBVRO2-RDBVR2))
          DE3=SLJ*(EO3-RDEO3+(2.*CBVRO3-RDBVR3))
          DB1=SLJ*(CBO1-RDBO1-(2.*EVRO1-RDEVR1))
          DB2=SLJ*(CBO2-RDBO2-(2.*EVRO2-RDEVR2))
          DB3=SLJ*(CBO3-RDBO3-(2.*EVRO3-RDEVR3))
          E1=E1+DE1
          E2=E2+DE2
          E3=E3+DE3
          CB1=CB1+DB1
          CB2=CB2+DB2
          CB3=CB3+DB3
          GO TO 120
200       E1=E1*FOPIM1
          E2=E2*FOPIM1
          E3=E3*FOPIM1
          CB1=CB1*FOPIM1
          CB2=CB2*FOPIM1
          CB3=CB3*FOPIM1
C
C      SAVE THE COMPONENTS OF THE FIELDS
C      IF MAGNITUDE LESS THAN 1.E-39, SET EQUAL TO ZERO TO AVOID
C      UNDERFLOW WHEN SQUARING
C
210       IF(ABS(E1).LT.1.E-39) E1=0.
          IF(ABS(E2).LT.1.E-39) E2=0.
          IF(ABS(E3).LT.1.E-39) E3=0.
          IF(ABS(CB1).LT.1.E-39) CB1=0.
          IF(ABS(CB2).LT.1.E-39) CB2=0.
          IF(ABS(CB3).LT.1.E-39) CB3=0.
          EL1(J)=E1
          EL2(J)=E2
          EL3(J)=E3
          CBM1(J)=CB1
```

```
            CBM2(J)=CB2
            CBM3(J)=CB3
300       CONTINUE
C
C       SAVE ARRAYS
C
            OPEN(2,FILE=FL2,STATUS='OLD',ACCESS='DIRECT',
        1   FORM='UNFORMATTED',RECL=NJ4)
            CALL WRITUN(EL1,NJ,NR+1)
            CALL WRITUN(EL2,NJ,NR+2)
            CALL WRITUN(EL3,NJ,NR+3)
            CALL WRITUN(CBM1,NJ,NR+4)
            CALL WRITUN(CBM2,NJ,NR+5)
            CALL WRITUN(CBM3,NJ,NR+6)
            NR=NR+6
            CLOSE(2)
350     CONTINUE
        STOP
1       FORMAT(E12.5)
2       FORMAT(4I4)
3       FORMAT(C3,/,I1)
        END
C
        SUBROUTINE WRITUN(A,N,IR)
C
C       THIS SUBROUTINE IS USED TO WRITE UNFORMATTED DATA TO
C       A DIRECT ACCESS FILE
C
        DIMENSION A(N)
        WRITE(2,REC=IR) A
        RETURN
        END
```

```
      SUBROUTINE POINT(X,Y,Z,T,NI,NJ,INP)
C
C     MICRO AND OPTICAL METROLOGY GROUP
C     NATIONAL BUREAU OF STANDARDS
C
C     EGON MARX       11/29/83
C
C     THIS SUBROUTINE RETURNS A SEQUENCE OF FIELD POINTS
C     TO THE MAIN PROGRAM BALL.
C
      LOGICAL INP
      DATA DX,DY,DZ,DT,NT1,NT2/4*0.,2*0/
      COMMON /RCOM/ R,R1,R2,R3,SOLANG
      COMMON /ICOM/ N1,N2,NX
      COMMON /PARAM/ ALPHA,BETA,CAPPA,CUT,DPATCH,PI,TOPI,PITO,FOPI,C
C
C     GO TO 200 TO READ NEW INPUT DATA IF "INP" IS TRUE.
C
      IF(INP) GO TO 200
C
C     PASS NEW COORDINATES
C
      X=XX
      Y=YY
      Z=ZZ
      T=TT
      NJ1=NJ1+1
C     CHECK FOR END OF ROW
      IF(NJ1.LE.NJ) THEN
C     INCREMENT VARIABLE
        IF(NT1.EQ.1) THEN
          XX=XX+DX
        ELSE IF(NT1.EQ.2) THEN
          YY=YY+DY
        ELSE IF(NT1.EQ.3) THEN
          ZZ=ZZ+DZ
        ELSE
          TT=TT+DT
        END IF
      ELSE
        NJ1=1
C     RESET VARIABLE TO INITIAL VALUE
        IF(NT1.EQ.1) THEN
          XX=XO
        ELSE IF(NT1.EQ.2) THEN
          YY=YO
        ELSE
          ZZ=ZO
        END IF
C     INCREMENT OTHER VARIABLE
        IF(NT2.EQ.2) THEN
          YY=YY+DY
        ELSE IF(NT2.EQ.3) THEN
          ZZ=ZZ+DZ
        ELSE
```

```
          TT=TT+DT
        END IF
      END IF
      RETURN
C
C     STATEMENTS TO READ IN FIELD-POINT COORDINATES,
C     THE INITIAL TIME, NUMBER OF THETA STRIPS,
C     AND MAXIMUM NUMBER OF PATCHES ON STRIP
C
200   READ(3,1) XX,YY,ZZ,TT
C     SAVE INITIAL VALUES
      XO=XX
      YO=YY
      ZO=ZZ
      READ(3,3) N1,N2
C     READ PULSE SIZE AND DURATION
      READ(3,1) DELX,DELY,DELZ,DELT
      NJ1=1
C     CHECK FOR PROPER GRID VALUES
      IF(NI.LE.0.OR.NJ.LE.1) CALL EXIT(7)
C     COMPUTE COORDINATE INCREMENTS
      NN=NJ
      IF(DELT.NE.0.) THEN
        DT=DELT/(NN-1)
        NN=NI
        NT1=4
        NT2=4
      END IF
      IF(DELZ.NE.0.) THEN
        DZ=DELZ/(NN-1)
        NN=NI
        NT1=3
        NT2=MAXO(NT2,3)
      END IF
      IF(DELY.NE.0.) THEN
        DY=DELY/(NN-1)
        NN=NI
        NT1=2
        NT2=MAXO(NT2,2)
      END IF
      IF(DELX.NE.0.) THEN
        DX=DELX/(NN-1)
        NT1=1
        NT2=MAXO(NT2,1)
      END IF
C     STOP IF ALL DELX ARE ZERO
      IF(NT1.EQ.0) THEN
        WRITE(6,*) 'ALL INCREMENTS ARE ZERO'
        CALL EXIT(8)
      END IF
C     GIVE WARNING IF ONLY ONE INCREMENT IS GIVEN BUT NI IS NOT 1
      IF(NT1.EQ.NT2.AND.NI.GT.1) WRITE(6,*)
     1 'WARNING: ONLY ONE INCREMENT GREATER THAN ZERO'
      IF(NX.EQ.0) RETURN
C
```

```
C       ADVANCE COORDINATES IF RESTARTING
C
        IF(NI.EQ.1) CALL EXIT(9)
        DO 260 I=1,NX
        IF(NT2.EQ.2) THEN
          YY=YY+DY
        ELSE IF(NT2.EQ.3) THEN
          ZZ=ZZ+DZ
        ELSE
          TT=TT+DT
        END IF
260     CONTINUE
        RETURN
1       FORMAT(E12.5)
3       FORMAT(2I4)
        END
```

```
      SUBROUTINE ICS(X,Y,Z,T,*,*)
C
C     MICRO AND OPTICAL METROLOGY GROUP
C     NATIONAL BUREAU OF STANDARDS
C
C     EGON MARX       11/29/83
C
C     THE FOLLOWING GROUP OF SUBROUTINES IS USED TO COMPUTE A PATCH
C     COVERING FOR THE INFORMATION-COLLECTING SPHERE, AND THEY RETURN
C     TO THE MAIN PROGRAM BALL THE VALUES OF THE COMPONENTS OF THE
C     UNIT VECTOR TO A PATCH AND THE SOLID ANGLE SUBTENDED BY THE PATCH
C
C     DPATCH = APPROXIMATE LENGTH OF A SIDE OF A PATCH
C     N1 = NUMBER OF THETA STRIPS (ALTERNATIVE)
C     N2 = MAXIMUM NUMBER OF ANGLES PHI (ALTERNATIVE)
C     IP = POINTER FOR PATCHES
C     IRP = POINTER FOR PATCHES
C
      DIMENSION PMIN(2),PMAX(2),PRP(2),RPMIN(2),RPMAX(2)
      COMMON /RCOM/ R,R1,R2,R3,SOLANG
      COMMON /ICOM/ N1,N2,NX
      COMMON /PARAM/ ALPHA,BETA,CAPPA,CUT,DPATCH,PI,TOPI,PITO,FOPI,C
      DATA LBL/0/
      GO TO (150,200,250),LBL
      IP=0
      IRP=0
      G=180./PI
C     COMPUTE RADIUS OF CYLINDER WHERE INITIAL DATA ARE SIGNIFICANT
      RHO=SQRT(CUT/CAPPA)
C     COMPUTE COORDINATE OF BOTTOM OF CYLINDER
      ZM=-CUT/ALPHA
C     COMPUTE RADIUS OF INFORMATION-COLLECTING SPHERE
      R=C*T
C     CHECK THAT Z IS SUCH THAT THERE MAY BE AN INTERSECTION
      IF(Z.LT.R.AND.Z.GT.ZM-R) GO TO 70
      RETURN 2
C     COMPUTE DISTANCE OF FIELD POINT TO Z-AXIS
70    RP=SQRT(X**2+Y**2)
C     COMPUTE AZIMUTH ANGLE OF FIELD POINT
      IF(X.EQ.0..AND.Y.EQ.0.) THEN
        PHI0=0.
      ELSE
        PHI0=ATAN2(-Y,-X)
      ENDIF
C     COMPUTE ANGULAR SIZE OF PATCHES
      DTH=DPATCH/R
      NP=PI/DTH
C     COMPARE TO MINIMUM NUMBER OF STRIPS
      IF(NP.LT.N1) DTH=PI/N1
C     COMPUTE PATCH SIZE IN PHI-DIRECTION
      DL2=DPATCH
      IF(2*NP.LT.N2) DL2=TOPI*R/N2
      DTH2=DTH*.5
C     COMPUTE THETA FOT TOP AND BOTTOM OF CYLINDER
      THETAT=0.
```

```
      THETAB=PI
      IF(ABS(Z).LE.R) THETAT=ACOS(-Z/R)+1.E-5
      IF(ABS(ZM-Z).LE.R) THETAB=ACOS((ZM-Z)/R)
C     COMPUTE DISTANCE FROM FIELD POINT TO CYLINDER
      RPR=ABS(RP-RHO)
C     CHECK IF FIELD POINT IS OUTSIDE CYLINDER
      IF(RP.GT.RHO) THEN
C     CHECK FOR INTERSECTION
        IF(R.LE.RPR) THEN
          RETURN 2
        ELSE
C     COMPUTE THETA FOR INTERSECTION
          THETA1=ASIN(RPR/R)
          THETA2=PI-THETA1
C     CHECK FOR SECOND INTERSECTION
          IF(R.LE.RP+RHO) THEN
C     CHECK POSITION OF TOP AND BOTTOM OF CYLINDER
            IF(THETA2.LE.THETAT.OR.THETA1.GE.THETAB) THEN
              RETURN 2
            ELSE
C     SAVE RANGE OF THETA FOR PARTIAL STRIP
              IP=IP+1
              PMIN(IP)=AMAX1(THETA1,THETAT)
              PMAX(IP)=AMIN1(THETA2,THETAB)
              PRP(IP)=RP
            ENDIF
          ELSE
C     COMPUTE ANGLES FOR SECOND INTERSECTION
            THETA3=ASIN((RP+RHO)/R)
            THETA4=PI-THETA3
C     CHECK POSITION OF TOP AND BOTTOM OF CYLINDER
            IF(THETA2.LE.THETAT.OR.THETA1.GE.THETAB.OR.
     1      (THETA3.LE.THETAT.AND.THETA4.GE.THETAB)) THEN
              RETURN 2
            ELSE
              IF(THETAT.LT.THETA3) THEN
C     SAVE RANGE OF THETA FOR PARTIAL STRIP
                IP=IP+1
                PMIN(IP)=AMAX1(THETAT,THETA1)
                PMAX(IP)=AMIN1(THETAB,THETA3)
                PRP(IP)=RP
                IF(THETAB.GT.THETA4) THEN
C     SAVE RANGE OF THETA FOR PARTIAL STRIP
                  IP=IP+1
                  PMIN(IP)=THETA4
                  PMAX(IP)=AMIN1(THETAB,THETA2)
                  PRP(IP)=RP
                ENDIF
              ELSE
C     SAVE RANGE OF THETA FOR PARTIAL STRIP
                IP=IP+1
                PMIN(IP)=AMAX1(THETAT,THETA4)
                PMAX(IP)=AMIN1(THETAB,THETA2)
                PRP(IP)=RP
              ENDIF
```

```
            ENDIF
          ENDIF
        ENDIF
C     IF THE FIELD POINT IS INSIDE THE CYLINDER
      ELSE
C     CHECK IF CYLINDER DOES NOT INTERSECT SPHERE IN TWO CURVES
        IF(R.LE.RP+RHO) THEN
C     CHECK IF SPHERE IS INSIDE CYLINDER
          IF(R.LE.RPR) THEN
C     SAVE RANGE OF THETA FOR FULL STRIP
            IRP=IRP+1
            RPMIN(IRP)=THETAT
            RPMAX(IRP)=THETAB
C     IF SPHERE INTERSECTS CYLINDER IN ONE CURVE
          ELSE
C     FIND EXTREME ANGLES
            THETA1=ASIN(RPR/R)
            THETA2=PI-THETA1
C     CHECK POSITION OF TOP OF CYLINDER
            IF(THETAT.LT.THETA1) THEN
C     SAVE RANGE OF THETA FOR FULL STRIP
              IRP=IRP+1
              RPMIN(IRP)=THETAT
              RPMAX(IRP)=AMIN1(THETA1,THETAB)
            ENDIF
            IF(THETAT.LT.THETA2.AND.THETAB.GT.THETA1) THEN
C     SAVE RANGE OF THETA FOR PARTIAL STRIP
              IP=IP+1
              PMIN(IP)=AMAX1(THETA1,THETAT)
              PMAX(IP)=AMIN1(THETA2,THETAB)
              PRP(IP)=-RP
            ENDIF
            IF(THETAB.GT.THETA2) THEN
C     SAVE RANGE OF THETA FOR FULL STRIP
              IRP=IRP+1
              RPMIN(IRP)=AMAX1(THETA2,THETAT)
              RPMAX(IRP)=THETAB
            ENDIF
          ENDIF
C     IF SPHERE INTERSECTS CYLINDER IN TWO CURVES
        ELSE
C     COMPUTE TWO EXTREME ANGLES
          THETA3=ASIN((RP+RHO)/R)
          THETA4=PI-THETA3
C     CHECK POSITION OF TOP AND BOTTOM OF CYLINDER
          IF(THETAT.GE.THETA3.AND.THETAB.LE.THETA4) THEN
            RETURN 2
          ELSE
C     COMPUTE OTHER TWO ANGLES
            THETA1=ASIN(RPR/R)
            THETA2=PI-THETA1
C     CHECK TOP OF CYLINDER
            IF(THETAT.LT.THETA1) THEN
C     SAVE RANGE OF THETA FOR FULL STRIP
              IRP=IRP+1
```

```
                RPMIN(IRP)=THETAT
                RPMAX(IRP)=AMIN1(THETAB,THETA1)
              ENDIF
              IF(THETAT.LT.THETA3.AND.THETAB.GT.THETA1) THEN
C       SAVE RANGE OF THETA FOR PARTIAL STRIP
                IP=IP+1
                PMIN(IP)=AMAX1(THETAT,THETA1)
                PMAX(IP)=AMIN1(THETAB,THETA3)
                PRP(IP)=-RP
              ENDIF
              IF(THETAT.LT.THETA2.AND.THETAB.GT.THETA4) THEN
C       SAVE RANGE OF THETA FOR PARTIAL STRIP
                IP=IP+1
                PMIN(IP)=AMAX1(THETAT,THETA4)
                PMAX(IP)=AMIN1(THETAB,THETA2)
                PRP(IP)=-RP
              ENDIF
              IF(THETAB.GT.THETA2) THEN
C       SAVE RANGE OF THETA FOR FULL STRIP
                IRP=IRP+1
                RPMIN(IRP)=AMAX1(THETAT,THETA2)
                RPMAX(IRP)=THETAB
              ENDIF
            ENDIF
          ENDIF
        ENDIF
C       END OF GEOMETRICAL CONSIDERATIONS BETWEEN SPHERE AND CYLINDER
C       CHECK FOR PARTIAL STRIPS
140     IF(IP.EQ.0) GO TO 190
C       INDICATE PROCESSING OF PARTIAL STRIP
        LBL=1
C       GET DATA FOR STRIP
        THMIN=PMIN(IP)
        THMAX=PMAX(IP)
        PR=PRP(IP)
C       DECREMENT POINTER
        IP=IP-1
C       COMPUTE COMPONENTS OF UNIT VECTOR TO PATCH
150     CALL PATCH(THMIN,THMAX,DTH,DL2,RHO,PR,PHIO,*140)
        RETURN
C       CHECK FOR FULL STRIPS
190     IF(IRP.EQ.0) GO TO 210
C       INDICATE PROCESSING OF FULL STRIP
        LBL=2
C       GET DATA FOR STRIP
        THMIN=RPMIN(IRP)
        THMAX=RPMAX(IRP)
C       DECREMENT POINTER
        IRP=IRP-1
C       COMPUTE COMPONENTS OF UNIT VECTOR TO PATCH
200     CALL RPATCH(THMIN,THMAX,DTH,DL2,*190)
        RETURN
C       INDICATE NO MORE STRIPS
210     LBL=3
        RETURN
```

```
C       ALL DONE FOR THIS FIELD POINT
250     LBL=0
        RETURN 1
        END
C
        SUBROUTINE PATCH(THMIN,THMAX,DTH,DL2,RHO,RP,PHIO,*)
C
C       THIS SUBROUTINE COMPUTES THE COMPONENTS OF THE UNIT VECTOR
C       TO A PATCH IN A PARTIAL STRIP
C
        COMMON /RCOM/ R,R1,R2,R3,SOLANG
        COMMON /PARAM/ ALPHA,BETA,CAPPA,CUT,DPATCH,PI,TOPI,PITO,FOPI,C
        DATA LBL/0/
        GO TO (100,110,120),LBL
C       CHECK FOR FIELD POINT ON AXIS OF CYLINDER (NO PARTIAL STRIPS)
        IF(RP.EQ.0) RETURN 1
C       COMPUTE RANGE IN THETA
        DELTH=THMAX-THMIN
C       COMPUTE NUMBER OF STRIPS
        NN1=DELTH/DTH
        DT1=DELTH/(NN1+1)
        DT2=DT1*.5
        B=.5/RP
        A=(RHO**2-RP**2)*B
        TH=THMIN+DT2
C       COMPUTE SOLID ANGLE SUBTENDED BY PATCH
        SLG=2.*DL2*SIN(DT2)/R
100     ST=SIN(TH)
        RST=R*ST
        CT=COS(TH)
C       COMPUTE SIZE OF PATCH IN AZIMUTHAL DIRECTION
        DPHI=DL2/RST
C       COMPUTE RANGE IN AZIMUTH FOR PARTIAL STRIP
        DELPHI=ACOS(B*RST-A/RST)
        NN2=DELPHI/DPHI
        DPHI=2.*DELPHI/(2*NN2+1)
        PHI1=PHIO
        PHI2=PHIO
C       SAVE COMPONENTS OF UNIT VECTOR AND SIZE OF CENTER PATCH
        CALL FILL(ST,CT,PHIO,SLG)
        I=1
        IF(NN2.EQ.0) GO TO 130
C       INDICATE ADDITIONAL PATCHES
        LBL=2
        RETURN
C       INCREMENT ANGLES
110     PHI1=PHI1+DPHI
        PHI2=PHI2-DPHI
C       SAVE COMPONENTS OF UNIT VECTOR AND SIZE OF PATCH
        CALL FILL(ST,CT,PHI1,SLG)
C       INDICATE SECOND ANGLE
        LBL=3
        RETURN
C       SAVE COMPONENTS OF UNIT VECTOR AND SIZE OF PATCH
120     CALL FILL(ST,CT,PHI2,SLG)
```

```
      I=I+1
C     CHECK IF DONE WITH STRIP
      IF(I.LE.NN2) THEN
C     INDICATE BACK TO FIRST ANGLE
        LBL=2
        RETURN
      ENDIF
C     INCREMENT THETA
130   TH=TH+DT1
C     CHECK IF DONE
      IF(TH.LE.THMAX) THEN
C     INDICATE START OF NEXT STRIP
        LBL=1
        RETURN
      ENDIF
C     INDICATE ALL DONE
      LBL=0
      RETURN 1
      END
C
      SUBROUTINE RPATCH(THMIN,THMAX,DTH,DL2,*)
C
C     THIS SUBROUTINE COMPUTES THE COMPONENTS OF THE UNIT VECTOR
C     TO A PATCH IN A FULL STRIP
C
      COMMON /RCOM/ R,R1,R2,R3,SOLANG
      COMMON /PARAM/ ALPHA,BETA,CAPPA,CUT,DPATCH,PI,TOPI,PITO,FOPI,C
      DATA LBL/0/
      GO TO (80,90,100,110),LBL
      IF(THMIN.EQ.0.) THEN
C     PROCESS CAP AT TOP OF SPHERE
        DT2=AMIN1(DTH*.5,THMAX)
        S1=FOPI*SIN(DT2*.5)**2
        CALL FILL(0.,1.,0.,S1)
        IF(DT2.EQ.THMAX) RETURN 1
        THMIN=DT2
        LBL=1
        RETURN
      ENDIF
80    IF(THMAX.EQ.PI) THEN
C     PROCESS CAP AT BOTTOM OF SPHERE
        DT2=AMIN1(DTH*.5,PI-THMIN)
        S1=FOPI*SIN(DT2*.5)**2
        CALL FILL(0.,-1.,0.,S1)
        THMAX=PI-DT2
        IF(THMIN.GE.THMAX) RETURN 1
        LBL=2
        RETURN
      ENDIF
C     COMPUTE RANGE IN THETA
90    DELTH=THMAX-THMIN
      NN1=DELTH/DTH
      DT1=DELTH/(NN1+1)
      DT2=DT1*.5
      TH=THMIN+DT2
```

```
100   ST=SIN(TH)
      RST=R*ST
      CT=COS(TH)
C     COMPUTE NUMBER OF PATCHES IN STRIP
      NN2=MAXO(INT(TOPI*RST/DL2+1.),3)
C     COMPUTE SIZE OF PATCH IN AZIMUTHAL DIRECTION
      DPHI=TOPI/NN2
      PHI=0.
C     COMPUTE SOLID ANGLE SUBTENDED BY PATCH
      SLG=2.*DPHI*ST*SIN(DT2)
      I=1
C     INDICATE NEXT PATCH
      LBL=4
C     SAVE COMPONENTS OF UNIT VECTOR AND SIZE OF PATCH
110   CALL FILL(ST,CT,PHI,SLG)
C     INCREMENT ANGLE
      PHI=PHI+DPHI
      I=I+1
      IF(I.LE.NN2) RETURN
      TH=TH+DT1
      IF(TH.LT.THMAX) THEN
C     INDICATE END OF STRIP PROCESSING
        LBL=3
        RETURN
      ENDIF
C     INDICATE ALL DONE
      LBL=0
      RETURN 1
      END
C
      SUBROUTINE FILL(ST,CT,PHI,SLG)
C
C     THIS SUBROUTINE COMPUTES AND STORES THE COMPONENTS OF THE
C     UNIT VECTOR TO THE PATCH, AND THE SIZE OF THE PATCH
C
      COMMON /RCOM/ R,R1,R2,R3,SOLANG
      R1=ST*COS(PHI)
      R2=ST*SIN(PHI)
      R3=CT
      SOLANG=SLG
      RETURN
      END
```

```
      PROGRAM                          BALPLT
C
C     MICRO AND OPTICAL METROLOGY GROUP
C     NATIONAL BUREAU OF STANDARDS
C
C     EGON MARX       11/29/83
C
C     THIS PROGRAM IS USED TO MAKE A THREE-DIMENSIONAL PLOT
C     OF THE ENERGY DENSITY FROM THE OUTPUT OF PROGRAM BALL IF TWO
C     COORDINATES VARY, OR A PLOT OF EACH COMPONENT OF THE ELECTROMAGNETIC
C     FIELD IF ONLY ONE COORDINATE VARIES.
C
      DIMENSION EB(100,100),XLAB(2,4),VAR(512)
      DIMENSION EL1(512),EL2(512),EL3(512),CBM1(512),CBM2(512),CBM3(512)
      INTEGER PRLINE(20)
      CHARACTER QAL*3,FL1*10,FL2*9
      EQUIVALENCE (EB,VAR)
      DATA XLAB/'X-COORD.','Y-COORD.','Z-COORD.','  TIME  '/
C
C     INITIALIZE THE PLOTTING ROUTINES
C
      READ(5,*) IANG1,IANG2
      THETA=IANG1
      PHI=IANG2
      READ(5,3) QAL
      FL1='BALOUT.'//QAL
      FL2='BALIN.'//QAL
      OPEN(4,FILE=FL2,STATUS='OLD')
      READ(4,2) NI,NJ
C
C     CHECK FOR VALID VALUES FOR PLOTTING ANGLES
C
      IF(NI.GT.1.AND.(IANG1.EQ.0.OR.IANG2.EQ.0)) THEN
        WRITE(6,*) 'SUPPLY NONZERO VALUES FOR THETA AND PHI'
        CALL EXIT(1)
      END IF
      NJ4=NJ*4
      NDI=(NI-1)/100+1
      NDJ=(NJ-1)/100+1
      READ(4,1) XO,YO,ZO,TO
      READ(4,4) IDUM,IDUM
      READ(4,1) DELX,DELY,DELZ,DELT
      OPEN(2,FILE=FL1,STATUS='OLD')
C
C     SELECT VALUES FOR PLOTTING
C     READ IN DATA FILES
C
      II=0
      DO 110 I=1,NI
        CALL READUN(EL1,NJ)
        CALL READUN(EL2,NJ)
        CALL READUN(EL3,NJ)
        CALL READUN(CBM1,NJ)
        CALL READUN(CBM2,NJ)
        CALL READUN(CBM3,NJ)
```

```
        IF(NI.EQ.1) GO TO 200
        IF(MOD(I-1,NDI).NE.0) GO TO 110
        II=II+1
        JJ=0
        DO 100 J=1,NJ,NDJ
          JJ=JJ+1
          EB(II,JJ)=EL1(J)**2+EL2(J)**2+EL3(J)**2+
     1    CBM1(J)**2+CBM2(J)**2+CBM3(J)**2
100     CONTINUE
110   CONTINUE
      UMAX=0.
      DO 120 I=1,II
      DO 120 J=1,JJ
        UMAX=AMAX1(UMAX,EB(I,J))
120   CONTINUE
C
C     PREPARE A THREE-DIMENSIONAL PLOT
C
      CALL PLOTS(0,0,0)
      ENCODE(PRLINE,13) UMAX
      CALL SYMBOL(.5,9.2,.1,PRLINE,0.,36)
      IF(DELY.NE.0..AND.DELT.NE.0.) GO TO 130
      IF(DELZ.NE.0..AND.DELT.NE.0.) GO TO 140
      IF(DELY.NE.0..AND.DELZ.NE.0.) GO TO 150
      IF(DELX.NE.0..AND.DELZ.NE.0.) GO TO 160
      IF(DELX.NE.0..AND.DELY.NE.0.) GO TO 170
      CALL SYMBOL(.5,10.,.15,
     1 'ENERGY DENSITY AS A FUNCTION OF X AND T',0.,39)
      ENCODE(PRLINE,7) XO,YO,ZO,TO,DELX,DELT
      GO TO 180
130   CALL SYMBOL(.5,10.,.15,
     1 'ENERGY DENSITY AS A FUNCTION OF Y AND T',0.,39)
      ENCODE(PRLINE,8) XO,YO,ZO,TO,DELY,DELT
      GO TO 180
140   CALL SYMBOL(.5,10.,.15,
     1 'ENERGY DENSITY AS A FUNCTION OF Z AND T',0.,39)
      ENCODE(PRLINE,9) XO,YO,ZO,TO,DELZ,DELT
150   CALL SYMBOL(.5,10.,.15,
     1 'ENERGY DENSITY AS A FUNCTION OF Y AND Z',0.,39)
      ENCODE(PRLINE,10) XO,YO,ZO,TO,DELY,DELZ
      GO TO 180
160   CALL SYMBOL(.5,10.,.15,
     1 'ENERGY DENSITY AS A FUNCTION OF X AND Z',0.,39)
      ENCODE(PRLINE,11) XO,YO,ZO,TO,DELX,DELZ
      GO TO 180
170   CALL SYMBOL(.5,10.,.15,
     1 'ENERGY DENSITY AS A FUNCTION OF X AND Y',0.,39)
      ENCODE(PRLINE,12) XO,YO,ZO,TO,DELX,DELY
180   CALL SYMBOL(.5,9.5,.1,PRLINE,0.,75)
      CALL PLOT3D(EB,0.,0.,100,100,1,1,II,1,1,JJ,1,THETA,PHI)
      CALL PLOT(0.,0.,999)
      STOP
C
C     IF ONLY ONE COORDINATE CHANGES, PREPARE MATERIAL FOR PLOT
C
```

```
200    IF(DELX.NE.0.) THEN
         V=XO
         DV=DELX/(NJ-1)
         NTY=1
       END IF
       IF(DELY.NE.0.) THEN
         V=YO
         DV=DELY/(NJ-1)
         NTY=2
       END IF
       IF(DELZ.NE.0.) THEN
         V=ZO
         DV=DELZ/(NJ-1)
         NTY=3
       END IF
       IF(DELT.NE.0.) THEN
         V=TO
         DV=DELT/(NJ-1)
         NTY=4
       END IF
       DO 210 I=1,NJ
         VAR(I)=V
         V=V+DV
210    CONTINUE
       ENCODE(PRLINE,6) XO,YO,ZO,TO
       CALL DRAW4(1,7,2,4,16,17,XLAB(1,NTY),'FIELD AMPLITUDES',
      1 'ELECTROMAGNETIC FIELDS, 1:E1, 2:E2, 3:E3, 4:CB1, 5:CB2, 6:CB3
      2 ',PRLINE)
C
C      PREPARE ONE PLOT WITH ALL OF THE 6 FIELD COMPONENTS AS A
C      FUNCTION OF THE INDEPENDENT VARIABLE
C
       CALL DRAW4(2,7,1,NJ,'1',100,VAR,EL1,0.,0.)
       CALL DRAW4(2,7,1,NJ,'2',110,VAR,EL2,0.,0.)
       CALL DRAW4(2,7,1,NJ,'3',120,VAR,EL3,0.,0.)
       CALL DRAW4(2,7,1,NJ,'4',130,VAR,CBM1,0.,0.)
       CALL DRAW4(2,7,1,NJ,'5',140,VAR,CBM2,0.,0.)
       CALL DRAW4(2,7,1,NJ,'6',150,VAR,CBM3,0.,0.)
       CALL DRAW4(3,7,0,0,2,-NJ,EL1,EL2,2.,2.)
       CALL PLOT(0.,0.,999)
       STOP
1      FORMAT(E12.5)
2      FORMAT(//////,2I4)
3      FORMAT(C3)
4      FORMAT(2I4)
6      FORMAT('  INITIAL VALUES: XO=',1PE8.1,',  YO=',E8.1,',  ZO=',E8.1,
      1 ',  TO=',E8.1)
7      FORMAT('XO=',1PE8.1,' YO=',E8.1,' ZO=',E8.1,' TO=',E8.1,
      1 ' DELX=',E8.1,' DELT=',E8.1)
8      FORMAT('XO=',1PE8.1,' YO=',E8.1,' ZO=',E8.1,' TO=',E8.1,
      1 ' DELY=',E8.1,' DELT=',E8.1)
9      FORMAT('XO=',1PE8.1,' YO=',E8.1,' ZO=',E8.1,' TO=',E8.1,
      1 ' DELZ=',E8.1,' DELT=',E8.1)
10     FORMAT('XO=',1PE8.1,' YO=',E8.1,' ZO=',E8.1,' TO=',E8.1,
      1 ' DELY=',E8.1,' DELZ=',E8.1)
```

```
11     FORMAT('XO=',1PE8.1,' YO=',E8.1,' ZO=',E8.1,' TO=',E8.1,
      1  ' DELX=',E8.1,' DELZ=',E8.1)
12     FORMAT('XO=',1PE8.1,' YO=',E8.1,' ZO=',E8.1,' TO=',E8.1,
      1  ' DELX=',E8.1,' DELY=',E8.1)
13     FORMAT('MAXIMUM ENERGY DENSITY IS ',1PE10.3)
       END
C
       SUBROUTINE READUN(A,N)
C
C      THIS SUBROUTINE IS USED TO READ UNFORMATTED DATA
C
       DIMENSION A(N)
       READ(2) A
       RETURN
       END
```

```
      PROGRAM                          BALDSK
C
C     MICRO AND OPTICAL METROLOGY GROUP
C     NATIONAL BUREAU OF STANDARDS
C
C     EGON MARX       11/29/83
C
C     THIS PROGRAM PREPARES FILES TO BE PROCESSED BY THE I2S
C     FACILITY TO BE PUT ON A MONITOR OR PHOTOGRAPHED.
C     THE INPUT IS THE FILE PRODUCED BY THE PROGRAM BALL.
C     THE OUTPUT IS A 512X512 SET OF INTEGERS THAT REPRESENT THE
C     SCALED INTENSITY OF THE PICTURES.
C     WHEN A SERIES OF OUTPUT FILES IS PROCESSED SIMULTANEOUSLY,
C     THE MAXIMUM INTENSITY OF EACH OUTPUT IS SCALED
C
      DIMENSION EB(512),IEB(128),EB1(512),EB2(512)
      DIMENSION EL1(512),EL2(512),EL3(512),CBM1(512),CBM2(512),CBM3(512)
      DIMENSION AMX(1000)
      LOGICAL EX
      CHARACTER*12 FL
      DATA IFL/0/
      OPEN(5,FILE='BALDSK.INP',STATUS='OLD')
      READ(5,*) N1,N2
      IF(N1.GT.N2.OR.N2.GE.1000) CALL EXIT(254)
      IF(N1.EQ.N2) GO TO 265
C     GET OLD FILE OF MAXIMA, IF IT EXISTS
      INQUIRE(FILE='BALDSK.SAV',EXIST=EX)
      IF(EX) THEN
        OPEN(3,FILE='BALDSK.SAV',STATUS='OLD')
        GO TO 240
      END IF
C     OR CREATE A NEW ONE IF NOT
      OPEN(3,FILE='BALDSK.SAV',STATUS='NEW',
     1 FORM='UNFORMATTED',RECL=4,BLOCKSIZE=8)
C     OPEN A FILE FOR OUTPUT OF MAXIMA
      OPEN(1,FILE='BALDSK.MAX',STATUS='RENEW')
      AMMAX=0.
      ALMAX=1.E20
C     GET THE MAXIMA FROM THE DIFFERENT FILES
      DO 200 I=1,1000
        FL='BALOUT.'//ITOC(I-1,IDUM)
        OPEN(2,FILE=FL,STATUS='OLD',ERR=220,SHARE='SRO')
C     FIND THE SIZE OF THE FILES
        IF(IFL.EQ.0) THEN
          IFL=1
          INQUIRE(2,SIZE=NI,RECL=NJ4)
          NI=NI/6
          NJ=NJ4/4-1
        ENDIF
        AMAX=0.
        DO 110 II=1,NI
          CALL READUN(EL1,NJ)
          CALL READUN(EL2,NJ)
          CALL READUN(EL3,NJ)
          CALL READUN(CBM1,NJ)
```

```
          CALL READUN(CBM2,NJ)
          CALL READUN(CBM3,NJ)
          DO 100 J=1,NJ
            AMAX=AMAX1(AMAX,EL1(J)**2+EL2(J)**2+EL3(J)**2+
     1      CBM1(J)**2+CBM2(J)**2+CBM3(J)**2)
100       CONTINUE
110     CONTINUE
        CLOSE(2)
        AMX(I)=AMAX
        IF(AMAX.LE.AMMAX) GO TO 120
        AMMAX=AMAX
        NN=I
120     IF(AMAX.GE.ALMAX) GO TO 200
        ALMAX=AMAX
        MM=I
200   CONTINUE
      I=1000
220   WRITE(3) I-1
      DO 230 J=1,I-1
        WRITE(3) AMX(J)
        WRITE(1,4) J-1,AMX(J)
230   CONTINUE
      WRITE(1,5) NN-1,AMMAX
      WRITE(1,6) MM-1,ALMAX
      WRITE(3) AMMAX
      WRITE(3) ALMAX
      GO TO 260
C     RETRIEVE INFORMATION FROM SAVED FILES
240   READ(3) I
      DO 250 J=1,I
        READ(3) AMX(J)
250   CONTINUE
      READ(3) AMMAX
      READ(3) ALMAX
260   COEF=105./(AMMAX-ALMAX)
C     CREATE RASTER FILES FOR PICTURES
265   DO 700 NN=N1+1,N2+1
        FL='BALOUT.'//ITOC(NN-1,IDUM)
        OPEN(2,FILE=FL,STATUS='OLD',SHARE='SRO')
C     FIND INFORMATION ON FILE SIZE
        IF(IFL.EQ.0) THEN
          IFL=1
          INQUIRE(2,SIZE=NI,RECL=NJ4)
          NI=NI/6
          NJ=NJ4/4-1
        ENDIF
        FL='BALDSK.'//ITOC(NN-1,IDUM)
        OPEN(4,FILE=FL,FORM='BINARY',STATUS='RENEW',
     1  BLOCKSIZE=512)
        WRITE(6,11) NN-1
C     FIND MAXIMUM FOR SINGLE PICTURE
        IF(N1.EQ.N2) THEN
          AMAX=0.
          DO 280 II=1,NI
            CALL READUN(EL1,NJ)
```

```
            CALL READUN(EL2,NJ)
            CALL READUN(EL3,NJ)
            CALL READUN(CBM1,NJ)
            CALL READUN(CBM2,NJ)
            CALL READUN(CBM3,NJ)
            DO 270 J=1,NJ
              AMAX=AMAX1(AMAX,EL1(J)**2+EL2(J)**2+EL3(J)**2+
     1        CBM1(J)**2+CBM2(J)**2+CBM3(J)**2)
270         CONTINUE
280       CONTINUE
          FAC=255./AMAX
          REWIND 2
        ELSE
          FAC=(COEF*(AMX(NN)-ALMAX)+150.)/AMX(NN)
        ENDIF
C      READ IN FIRST RECORD
        CALL READUN(EL1,NJ)
        CALL READUN(EL2,NJ)
        CALL READUN(EL3,NJ)
        CALL READUN(CBM1,NJ)
        CALL READUN(CBM2,NJ)
        CALL READUN(CBM3,NJ)
C      LIMIT THE NUMBER OF POINTS TO 256 FOR SYMMETRY
        NJ1=MINO(NJ,256)
C      SCALE INTENSITIES
        DO 300 J=1,NJ1
          EB(J)=FAC*(EL1(J)**2+EL2(J)**2+EL3(J)**2+
     1    CBM1(J)**2+CBM2(J)**2+CBM3(J)**2)
300     CONTINUE
C      INTERPOLATE FIRST RECORD
        IJ=257
        EB02=EB(1)
        DO 320 II=2,NJ1
          EB01=EB02
          EB02=EB(II)
          K1=(512-IJ)/(NJ1+1-II)
          FC=1./K1
          DO 310 K=0,K1-1
            EB2(IJ)=FC*(EB01*(K1-K)+EB02*K)
            IJ=IJ+1
310       CONTINUE
320     CONTINUE
        EB2(IJ)=EB02
C      FILL IN THE REFLECTED VALUES ON RECORD
        DO 330 II=1,256
          EB2(II)=EB2(513-II)
330     CONTINUE
C      PROCEED WITH OTHER RECORDS, INTERPOLATING IN OTHER DIRECTION
        KJ=1
        DO 430 I=2,NI
C      SHIFT RECORD BETWEEN BUFFERS
          DO 340 K=1,512
            EB1(K)=EB2(K)
340       CONTINUE
         IJ=257
```

```
C      READ ANOTHER RECORD
           CALL READUN(EL1,NJ)
           CALL READUN(EL2,NJ)
           CALL READUN(EL3,NJ)
           CALL READUN(CBM1,NJ)
           CALL READUN(CBM2,NJ)
           CALL READUN(CBM3,NJ)
C      SCALE NEW RECORD
           DO 350 J=1,NJ1
             EB(J)=FAC*(EL1(J)**2+EL2(J)**2+EL3(J)**2+
     1       CBM1(J)**2+CBM2(J)**2+CBM3(J)**2)
350        CONTINUE
C      INTERPOLATE NEW RECORD
           EB02=EB(1)
           DO 370 II=2,NJ1
             EB01=EB02
             EB02=EB(II)
             K1=(512-IJ)/(NJ1+1-II)
             FC=1./K1
             DO 360 K=0,K1-1
               EB2(IJ)=FC*(EB01*(K1-K)+EB02*K)
               IJ=IJ+1
360          CONTINUE
370        CONTINUE
           EB2(IJ)=EB02
C      FILL IN REFLECTION OF NEW RECORD
           DO 380 II=1,256
             EB2(II)=EB2(513-II)
380        CONTINUE
C      INTERPOLATE BETWEEN RECORDS
           L1=(512-KJ)/(NI+1-I)
           FC=1./L1
           DO 420 L=0,L1-1
             IK=1
             DO 400 II=1,128
               IVAL=0
C      PACK VALUES IN BYTES
               DO 390 K=1,4
                 IVAL=IVAL*256+INT(FC*(EB1(IK)*(L1-L)+EB2(IK)*L))
                 IK=IK+1
390            CONTINUE
               IEB(II)=IVAL
400          CONTINUE
C
C      DO SPECIAL UNFORMATTED I/O TO WRITE TO RASTER FILE
C
             CALL SYSIO(IPB,57,4,IEB,512,0)
             KJ=KJ+1
420        CONTINUE
430      CONTINUE
C      DO LAST RECORD
         IK=1
         DO 500 II=1,128
           IVAL=0
           DO 450 K=1,4
```

```
            IVAL=IVAL*256+INT(EB2(IK))
            IK=IK+1
450       CONTINUE
          IEB(II)=IVAL
500     CONTINUE
C
C     DO SPECIAL UNFORMATTED I/O TO WRITE TO RASTER FILE
C
        CALL SYSIO(IPB,57,4,IEB,512,0)
        CLOSE(2)
700   CONTINUE
      STOP
4     FORMAT(' FILE ',I3,',  MAX=',1PE12.5)
5     FORMAT(//,' FILE ',I3,',  LARGEST MAX=',1PE12.5)
6     FORMAT(//,' FILE ',I3,',  SMALLEST MAX=',1PE12.5)
11    FORMAT(1X,I3)
      END
C
      SUBROUTINE READUN(A,N)
C
C     THIS SUBROUTINE IS USED TO READ UNFORMATTED DATA
C
      DIMENSION A(N)
      READ(2) A
      RETURN
      END
```

```
      PROGRAM                          BALCHK
C
C     MICRO AND OPTICAL METROLOGY GROUP
C     NATIONAL BUREAU OF STANDARDS
C
C     EGON MARX       11/29/83
C
C     THIS PROGRAM CHECKS THE CONSERVATION OF ENERGY AND MOMENTUM
C     OF THE ELECTROMAGNETIC FIELD BY COMPARING THESE QUANTITIES
C     AT TIME T WITH THEIR INITIAL VALUES.
C
      DIMENSION EB1(512),EB2(512),EXCB1(512),EXCB2(512)
      DIMENSION EL1(512),EL2(512),EL3(512),CBM1(512),CBM2(512),CBM3(512)
      REAL K,K2
      CHARACTER FL1*12,FL2*12,QAL*3
      DATA IFL/0/
C
      PI=4.*ATAN(1.)
      TOPI=2.*PI
C     OPEN FILE WHERE THE QUALIFIERS OF FILES TO BE PROCESSED ARE STORED
      OPEN(1,FILE='BALCHK.QAL',STATUS='OLD')
C     OPEN FILE TO SAVE THE OUTPUT
      OPEN(3,FILE='BALCHK.OUT',STATUS='RENEW')
100   READ(1,5,END=400) QAL
C     READ PULSE PARAMETERS
      FL1='BALIN.'//QAL
      OPEN(4,FILE=FL1,STATUS='OLD')
      READ(4,1) CAPPA,DUM,ALPHA,BETA,K,DUM
      READ(4,2) NI,NJ
      READ(4,1) XO,YO,ZO,TO
      READ(4,2) IDUM,IDUM
      READ(4,1) DELX,DELY,DELZ,DELT
      DX=DELX/(NJ-1)
      DZ=DELZ/(NI-1)
      IF(IFL.EQ.0) THEN
        IFL=1
C
C     COMPUTE INITIAL VALUES OF THE ENERGY AND MOMENTUM OF THE
C     FIELDS FROM THE ANALYTIC EXPRESSIONS.
C
        C2=CAPPA**2
        A2=ALPHA**2
        B2=BETA**2
        AB=ALPHA*BETA
        APB=ALPHA+BETA
        IF(K.EQ.0.) THEN
          ERG=TOPI*(.25/C2*(.25/ALPHA**3+8.*(A2-4.*AB
     1     +B2)/APB**5+.25/BETA**3)
     2     +.375/CAPPA*(1./ALPHA**5-64./APB**5+1./BETA**5))
          EXB=PI*.25/C2*(.25/ALPHA**3+8.*(A2-4.*AB
     1     +B2)/APB**5+.25/BETA**3)
        ELSE
          K2=K**2
          APB2=APB**2
          ERG=TOPI*K2*(.25/C2*(.25/ALPHA-4.*(AB+K2)/(APB*(APB2+4.*K2))
```

```
     1     +.25/BETA)+1./CAPPA*(.25/(ALPHA*(A2+K2))
     2     -4./(APB*(APB2+4.*K2))+.25/(BETA*(B2+K2))))
           EXB=.25*PI*K2/C2*(.25/ALPHA-4.*(AB+K2)/(APB*(APB2+4.*K2))
     1     +.25/BETA)
         END IF
         WRITE(3,4) ERG,EXB
       END IF
       FL2='BALOUT.'//QAL
C
C      READ IN FIELD VALUES AND COMPUTE ENERGY DENSITY AND POYNTING VECTOR
C
       OPEN(2,FILE=FL2,STATUS='OLD')
C
C      DO THE NUMERICAL INTEGRATION TO FIND THE ENERGY AND MOMENTUM
C
       ERG=0.
       EXB=0.
       CALL READUN(EL1,NJ)
       CALL READUN(EL2,NJ)
       CALL READUN(EL3,NJ)
       CALL READUN(CBM1,NJ)
       CALL READUN(CBM2,NJ)
       CALL READUN(CBM3,NJ)
       DO 250 J=1,NJ
         EB2(J)=EL1(J)**2+EL2(J)**2+EL3(J)**2+
     1   CBM1(J)**2+CBM2(J)**2+CBM3(J)**2
         EXCB2(J)=EL1(J)*CBM2(J)-EL2(J)*CBM1(J)
250    CONTINUE
       DO 310 I=1,NI-1
         DO 260 J=1,NJ
           EB1(J)=EB2(J)
           EXCB1(J)=EXCB2(J)
260      CONTINUE
         CALL READUN(EL1,NJ)
         CALL READUN(EL2,NJ)
         CALL READUN(EL3,NJ)
         CALL READUN(CBM1,NJ)
         CALL READUN(CBM2,NJ)
         CALL READUN(CBM3,NJ)
         DO 270 J=1,NJ
           EB2(J)=EL1(J)**2+EL2(J)**2+EL3(J)**2+
     1     CBM1(J)**2+CBM2(J)**2+CBM3(J)**2
           EXCB2(J)=EL1(J)*CBM2(J)-EL2(J)*CBM1(J)
270      CONTINUE
         X1=X0
         DO 300 J=1,NJ-1
           X2=X1+DX
           SURF=X2**2-X1**2
           ERG=ERG+.25*(EB1(J)+EB2(J)+EB1(J+1)+EB2(J+1))*SURF
           EXB=EXB+.25*(EXCB1(J)+EXCB2(J)+EXCB1(J+1)+EXCB2(J+1))*SURF
           X1=X2
300      CONTINUE
310    CONTINUE
       ERG=ERG*PI*DZ
       EXB=EXB*PI*DZ
```

```
      WRITE(3,3) FL1,ERG,EXB
      GO TO 100
400   STOP
1     FORMAT(E12.5)
2     FORMAT(2I4)
3     FORMAT(1X,C12,2X,3E15.8)
4     FORMAT(//,19X,'ENERGY',8X,'MOMENTUM',/,' INITIAL VALUES',2E15.8)
5     FORMAT(A3)
      END
C
      SUBROUTINE READUN(A,N)
C
C     THIS SUBROUTINE IS USED TO READ UNFORMATTED DATA
C
      DIMENSION A(N)
      READ(2) A
      RETURN
      END
```

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 84-2835	2. Performing Organ. Report No.	3. Publication Date May 1984

4. TITLE AND SUBTITLE

Free-Space Propagation of Light Pulses

5. AUTHOR(S)

Egon Marx

6. PERFORMING ORGANIZATION (*If joint or other than NBS, see instructions*)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (*Street, City, State, ZIP*)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (*A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here*)

A transient electromagnetic field in free space is completely specified when the initial values of the electric and magnetic fields are given. Green's function for the scalar wave equation can then be used to find the field at later times. A group of computer programs that implement these equations and process the output are presented in this report.

12. KEY WORDS (*Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons*)

computer programs; conservation of energy and momentum; electromagnetic field propagation; finite light pulses; graphic displays of energy density; transient electromagnetic fields

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

53

15. Price

$10.00

USCOMM-DC 6043-P80