Electromagnetic Characterization Of The Basal Plane Region In Sea Ice

John R. Natzke and Thomas B.A. Senior

1 Introduction

In electromagnetic pulse measurements of the depth of sea ice, it is found that under certain circumstances the strength of the return from the lower surface of the ice is strongly polarization dependent [1]. This occurs when there are well-developed basal planes, i.e. planes containing brine drainage channels extending approximately 5–20 cm up from the lower surface. The planes are closely spaced and parallel to one another, being perpendicular to a common c axis. Under these circumstances, the lower surface return is a maximum when the electric vector is parallel to the c axis, and a minimum when it is perpendicular to c. In the latter case it is not unusual to lose the lower surface return completely.

In an effort to understand this phenomenon, various models have been proposed, such as equivalent media theories, parallel metallic plates, etc. These have not been successful and, in some cases, have indicated results completely opposite to those observed. Thus, the task at hand is to develop a physically-based model that explains the observed effects and shows the dependence on channel length and spacing, brine concentration, wavelength and polarization. In this report we will present such a model based on the electrical properties of the brine drainage channels and the periodic nature of the basal planes with respect to the c axis. The solution to the scattered field will be determined for plane wave incidence using the appropriate electric field integral equation whose kernel is the Green's function for a periodic infinite array. Upon solving the integral equation numerically by the method of moments, results are obtained for the reflected field which support the observations made in [1].

2 The Model

In the formation of the basal planes, the brine concentration varies from virtually zero at the top of each plane to very high at the lower surface in

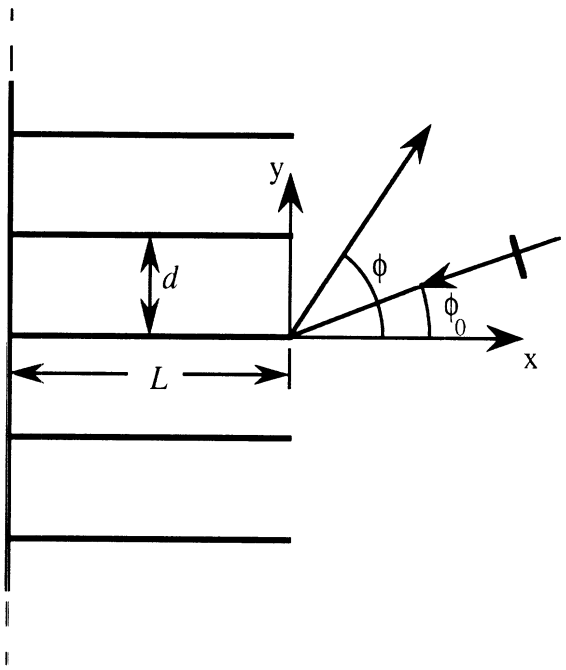

Figure 1: Infinite array of parallel resistive sheets with a perfectly conducting termination.

contact with the underlying sea [1]. It is convenient to think of the brine channels as concentrated in a layer of very small electrical thickness which can be modeled as a resistive sheet. The resistivity of each sheet therefore decreases from high at the top to zero at the bottom. An appropriate model for the basal plane region is then an infinite array of parallel resistive sheets terminated in a perfect conductor simulating the sea surface. When the resistive sheets are equally-spaced and have identical lengths and resistivity profiles, the array becomes a periodic structure, and the electric field integral equation can be readily solved numerically by employing the Green's function for a periodic infinite array.

The model of the basal plane region is shown in Fig. 1 as an infinite array of parallel resistive sheets of length L and separation d terminated in a perfect conductor with the y axis parallel to the c axis of the sea ice. The sheets have identical resistivities tapered quadratically from a maximum at the front ($x = 0$) to zero at the back ($x = -L$). The resistivity in ohms per square of the sheet at, say, $-L \leq x \leq 0$, $y = 0$ is given by

$$R(x,0) = R_0 \left(\frac{x+L}{L} \right)^2 \tag{1}$$

where R_0 is the specified maximum value of the resistivity at $x = 0$, and from the periodicity of the array in y,

$$R(x, pd) = R(x, 0)\ ,\ p = 0, \pm 1, \pm 2, \ldots\ .$$

The boundary (transition) condition of the total field across each of the sheets is

$$\hat{y} \times \hat{y} \times \bar{E}(x, pd) = -R(x, pd)\, \bar{J}(x, pd)\ ,\ -L < x \leq 0\ , \tag{2}$$

where $p = 0, \pm 1, \pm 2, \ldots$ and $\bar{J}$ is the total surface current on the sheets. For the perfect conductor in the $x = -L$ plane, the resistivity is zero, and the boundary condition for the total electric field is

$$\hat{x} \times \bar{E}(-L, y) = 0\ . \tag{3}$$

3 Electromagnetic Field Analysis

Consider the plane wave

$$E_z^i(H_z^i) = e^{-ik(x\cos\phi_0 + y\sin\phi_0)} \tag{4}$$

incident on the infinite array for E-(H-) polarization, where k is the propagation constant of the medium (assumed lossless) and a time factor $e^{-i\omega t}$ has been suppressed. The induced current on the resistive sheets and termination must be periodic in accordance with Floquet's theorem, such that

$$\bar{J}(x, pd) = \bar{J}(x, 0)\, e^{-ipkd\sin\phi_0}\ ,\ -L < x \leq 0\ , \tag{5}$$

and

$$\bar{J}(-L, y + pd) = \bar{J}(-L, y)\, e^{-ipkd\sin\phi_0} \tag{6}$$

with $p = 0, \pm 1, \pm 2, \ldots$. In light of (5) and (6), the scattered fields can be expressed in an integral form where the integration is only taken over one of the cells of the infinite array, say $-L \leq x \leq 0$, $0 \leq y < d$. For E-polarization, the scattered electric field is

$$E_z^s(x, y) = ikZ \int_{l'} J_z(x', y')\, G(x, y, x', y')\, dl'\ , \tag{7}$$

and for H-polarization, the scattered magnetic field is

$$H_z^s(x, y) = -\hat{z} \cdot \int_{l'} J_l(x', y')\, \hat{l} \times \nabla G(x, y, x', y')\, dl' \tag{8}$$

where Z is the intrinsic impedance of the medium and the path l' of the line integrals is over $-L \leq x \leq 0$, $y = 0$ and $0 \leq y < d$, $x = -L$. The unit vector $\hat{l}$ is tangent to the path of integration l', and the subscript l denotes the

component in the direction $\hat{l}$. The Green's function for the periodic infinite array is

$$G = \frac{i}{4} \sum_{p=-\infty}^{\infty} H_0^{(1)}\left(k\sqrt{(x-x')^2 + (y-y'-pd)^2}\right) e^{-ipkd\sin\phi_0} \tag{9}$$

where $H_0^{(1)}$ is the zeroth order Hankel function of the first kind. Since the series is slowly convergent, especially for small d, a more quickly converging form ($d \lesssim 2\lambda$) was derived in [2], giving

$$G = \frac{i}{2d} \sum_{p=-\infty}^{\infty} \frac{1}{k_x} e^{i(k_x|x-x'|+k_y(y-y'))} \tag{10}$$

with

$$\begin{aligned} k_x &= \sqrt{k^2 - k_y^2} \\ k_y &= k(p\lambda/d - \sin\phi_0)\,. \end{aligned}$$

To solve for the unknown surface currents in (7) and (8), an integral equation for the total electric field $\bar{E} = \bar{E}^i + \bar{E}^s$ is derived by applying the boundary conditions (2) and (3). For E-polarization, we obtain

$$\begin{aligned} E_z^i(x,y) = {} & R(x,y)\, J_z(x,y) \\ & - ikZ \int_{l'} J_z(x',y')\, G(x,y,x',y')\, dl'\,, \end{aligned} \tag{11}$$

and for H-polarization,

$$\begin{aligned} E_l^i(x,y) = {} & R(x,y)\, J_l(x,y) \\ & - ikZ \int_{l'} J_l(x',y')\, \hat{l} \cdot \left(\bar{\bar{I}} + \frac{1}{k^2}\nabla\nabla\right) G(x,y,x',y')\, dl' \end{aligned} \tag{12}$$

where (x,y) are taken on l' and, in (12), $E_l^i = (iZ/k)\,\hat{l}\cdot\nabla\times(\hat{z}H_z^i)$.

Upon solution of (11) and (12), the scattered fields are given by (7) and (8), respectively, and we are interested in the region $x > 0$. Inspection of the Green's function (10) reveals that the scattered field is comprised of reflected waves propagating away from the structure for k_x real ($k_y < \pm k$) and surface waves decaying exponentially away from the structure for k_x imaginary ($k_y > \pm k$). Thus the condition on the index p for a reflected wave to occur is

$$-\frac{d}{\lambda}(1 - \sin\phi_0) < p < \frac{d}{\lambda}(1 + \sin\phi_0)\,, \tag{13}$$

and the p^{th} wave propagates in the direction

$$\phi_p^r = \sin^{-1}(p\lambda/d - \sin\phi_0)\,, \tag{14}$$

taking the principle value of the arcsine measured from the positive x axis. In the far field only the reflected waves are observable, and for E-polarization, the p^{th} reflected wave is

$$E_{zp}^{r} = -\frac{kZ}{2k_x d}\, e^{i(k_x x + k_y y)} \int_{l'} J_z(x', y')\, e^{-i(k_x x' + k_y y')}\, dl' \,, \tag{15}$$

and likewise for H-polarization,

$$H_{zp}^{r} = \frac{1}{2d}\, e^{i(k_x x + k_y y)} \cdot \int_{l'} \hat{l} \cdot \left[\hat{x} \frac{k_y}{k_x} J_x(x', y') - \hat{y} J_y(x', y') \right] e^{-i(k_x x' + k_y y')}\, dl' \tag{16}$$

where k_x and k_y are evaluated at the p^{th} value of the infinite series.

4 Numerical Implementation

The integral equations (11) and (12) were programmed for numerical solution by the method of moments to determine the unkown surface currents on one cell of the infinite array. The surface currents were discretized by employing a pulse basis function expansion, and the remaining integral of the Green's function (10) over each segment was then evaluated analytically. Applying point matching over l' yielded a set of linear equations for solution, the unkown coefficients being the surface currents on each segment of the expansion. For E-polarization an appropriate upper limit on the infinite series in (10) was found to be $p = p_{max} \simeq 60$ for $d \lesssim 2$, with the convergence improving for decreasing d. For H-polarization the convergence was slowed due to the derivatives of the kernel (see (12)), and an auxiliary series was introduced following the development in [2] to improve the convergence rate such that $p_{max} \simeq 60$, $d \lesssim 2$.

The numerical implementation of the method of moments solution is contained in the code INFARR as listed in the Appendix. The user is prompted to choose either E- or H-polarization and to enter the length L, separation d, maximum resistivity R_0, and the order of taper of the resistive sheets, along with the upper limit p_{max} and angle of incidence ϕ_0. Other options include removing the perfectly conducting termination from the solution, specifying a constant profile for $R(x, 0)$ rather than a tapered one, and generating results over a range of L (all other parameters constant). We note that since a singularity exists in the Green's function series if $k_y = \pm k$, a restriction on d and ϕ_0 is that $p\lambda/d - \sin\phi_0 \neq \pm 1$ for any value of p. The user is notified if such a case exists and requested to change either parameter. The sampling rate used is 100 segments per wavelength, and this is increased by

$10(1-f)$, $f = d, L$ for $d, L < \lambda$ to ensure a proper sampling. The output then is the magnitude and phase of the specular reflected field given by (15) or (16) with $p = 0$, whose direction of propagation is $\phi_0^r = -\phi_0$. The user is also informed of which other reflected waves exist, if any, according to (13) and their direction of propagation (14). The magnitude and phase of these fields could be output as well by a minor change in the code.

5 Results and Discussion

For the results presented here, the case of primary interest was when $d < 1\lambda$ and $\phi_0 = 0$. Consequently, the condition (13) gives $p = 0$ as the only reflected wave. The quadratic resistivity profile in (1) will be assumed, and p_{max} is set to 50.

5.1 *E*-polarization

The reflected field E_{z0}^r was generated by (15) as a function of L, $0.1\lambda \leq L \leq 2\lambda$, for several values of d with $R_0 = 500\Omega$ as shown in Fig. 2 and for several values of R_0 with $d = 0.2\lambda$ as shown in Fig. 3. As L approaches zero, the reflected field becomes -1, which is the reflected field of the perfectly conducting termination. As L increases, the magnitude decreases to an asymptotic value. The asymptotes imply that there is no longer any contribution from the termination at $x = -L$ and are thus attributable to the scattering from the leading edges of the resistive sheets alone. This was verified by comparing the asymptotic values to the reflected fields produced when the termination is removed from the solution. Figure 4 shows this comparison for $d = 0.2\lambda$, $R_0 = 500\Omega$, and the asymptotes are indistinguishable. This can also be shown by comparing the results of Figs. 2 and 3 to the reflected field from resistive sheets of constant resistivity R_0 with the termination removed, for which the results of each case are almost identical as a function of d and R_0 for large L (see Fig. 5). In the limit L approaches infinity the reflected field in (15) is virtually equivalent to that from an infinite array of half planes of constant resistivity R_0. Therefore, as lossy parallel plate waveguides, the resistive sheets tremendously reduce the contribution from the termination.

We note that the rate of convergence to the asymptotic values decreases with increasing d and R_0, providing a larger return for $L \lesssim 0.5\lambda$ than for smaller d and R_0 (see Figs. 2 and 3). Thus the penetration of the incident field increases for larger d and R_0 as expected, which is also evidenced by the increased oscillations attributable to the termination. Even though this penetration increases the contribution from the termination, the asymptotic values decrease with increasing d and R_0, implying that the leading edge

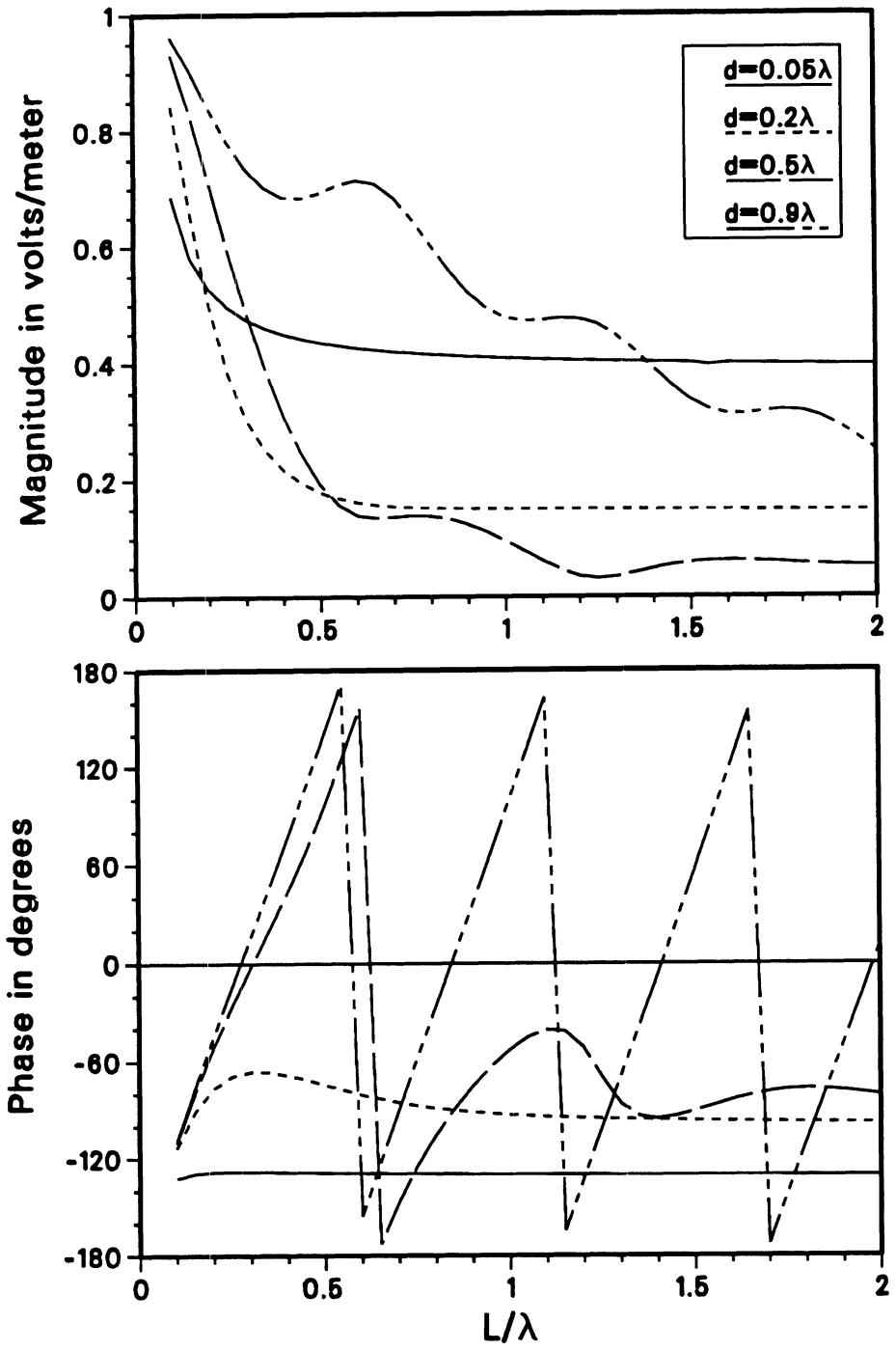

Figure 2: Reflected field E^r_{z0} of the infinite array for several d with $\phi_0 = 0$ and $R_0 = 500\Omega$ (quadratic taper).

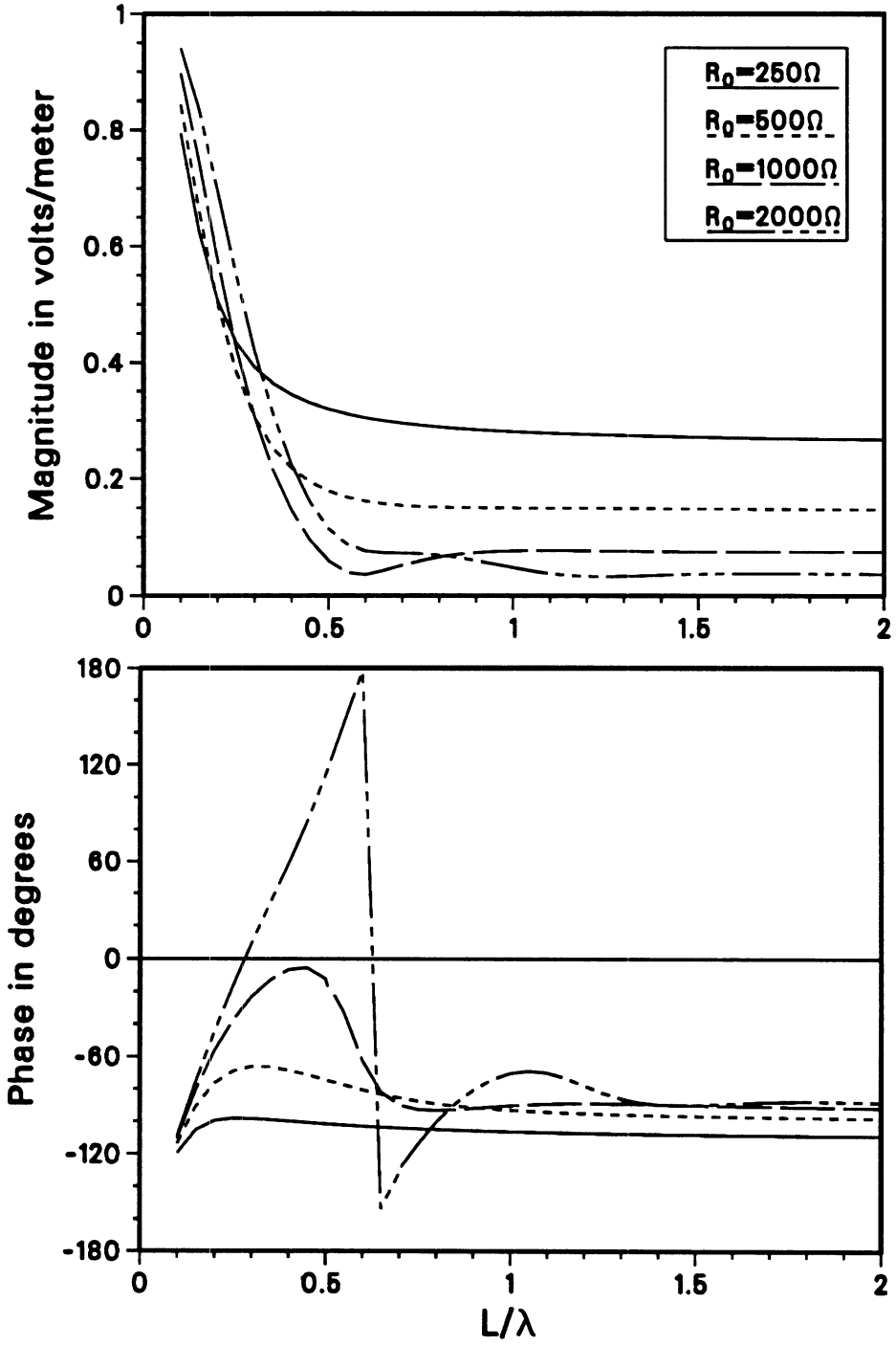

Figure 3: Reflected field E_{z0}^r of the infinite array for several R_0 with $\phi_0 = 0$ and $d = 0.2\lambda$ (quadratic taper).

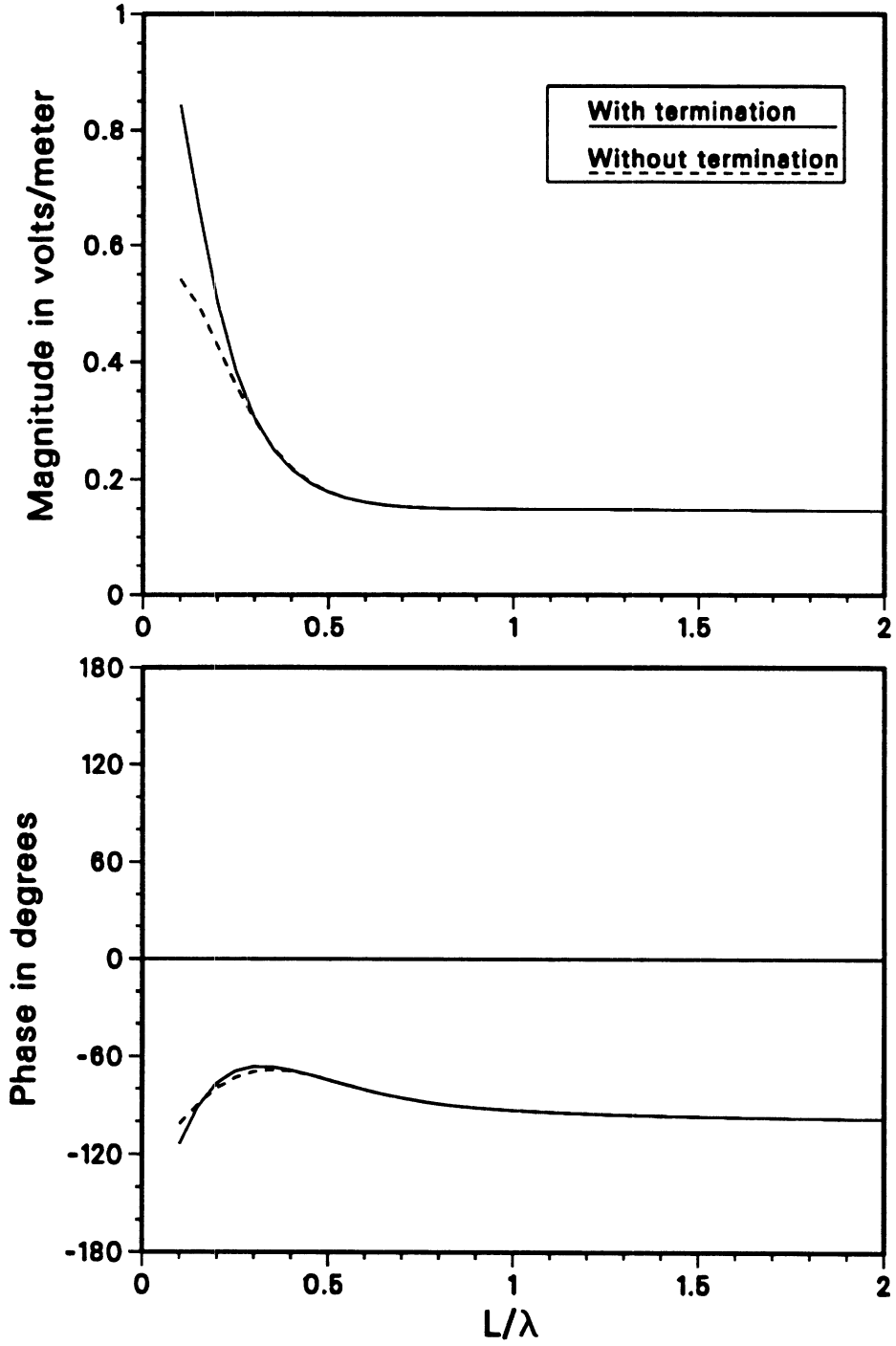

Figure 4: Comparison of reflected fields E^r_{z0} of the infinite array with and without the perfectly conducting termination at $x = -L$ for $\phi_0 = 0$, $d = 0.2\lambda$, and $R_0 = 500\Omega$ (quadratic taper).

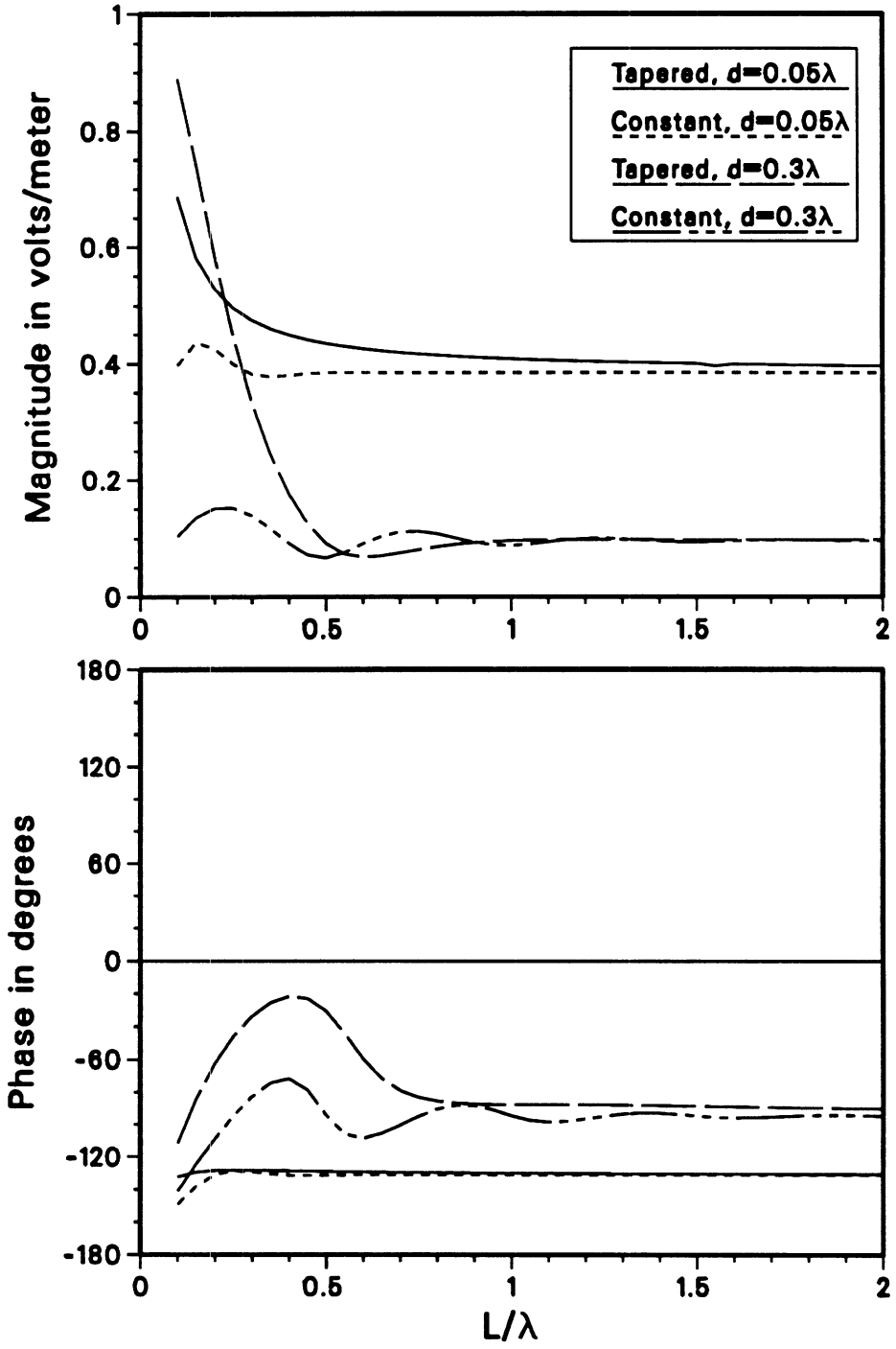

Figure 5: Comparison of reflected fields E^r_{z0} of the infinite array with quadratic taper and constant resistivity profiles for $\phi_0 = 0$ and $R_0 = 500\Omega$.

contribution is reduced. This behavior is expected for increasing R_0, but requires further analysis as a function of d.

As d approaches zero, the interaction between the adjacent sheets in the infinite array increases, and the effective resistivity decreases since the sheets appear in parallel with one another. Furthermore, in the limit the infinite array becomes a lossy material slab of thickness L, which can be characterized by some reflection coefficient, assuming the field transmitted at the leading surface is completely absorbed. Modeling the slab with a resistive sheet of resistivity $R_0/2$ in the $x = 0$ plane, we find that

$$E_z^r = -\frac{1}{1 + \frac{R_0}{Z}\cos\phi_0} e^{ik(x\cos\phi_0 - y\sin\phi_0)} ,$$

which gives comparable results to the asymptotic values of (15) for $d = 0.05\lambda$. The choice of a resistive sheet model as opposed to a lossy dielectric layer one is consistent with the nature of the transmitted field propagating in the $\phi_0 - \pi$ direction and gives the proper dependence on R_0. For $d \simeq 0.2\lambda$, it was observed from the data that for normal incidence and $R_0 \gtrsim 500\Omega$ the asymptotic value was comparable to the edge on backscatter of a resistive half plane of constant resistivity R_0 normalized by the backscatter of a perfectly conducting half plane, giving

$$E_z^r = -\frac{iZ}{4R_0} e^{-\frac{Z}{\pi R_0}} , \ \phi_0 = 0 .$$

The accuracy of this model in magnitude and phase implies that any interaction between adjacent resistive sheets is negligible. As d is increased from 0.2λ, the asymptotic values of E_z^r continue to decrease, suggesting that the absorption of the incident field is increased, though this phenomenom is not yet understood.

5.2 H-polarization

The reflected field H_{z0}^r was generated by (16), and, as expected, the magnitude is a constant as a function of L for $\phi_0 = 0$, as shown in Fig. 6 with $d = 0.2\lambda$, $R_0 = 500\Omega$. The reflection is simply the reflected field from the perfectly conducting termination, which is e^{i2kL}. With the electric vector perpendicular to the resistive sheets, the sheets have no effect on the incident field, and this is so as a function of d, L, and R_0. For $\phi_0 \simeq 0$ the result still remains virtually unchanged, and as ϕ_0 is increased the effect of the resistive sheets increases, which is evidenced by the results in Fig. 6 for $\phi_0 = \pi/4$.

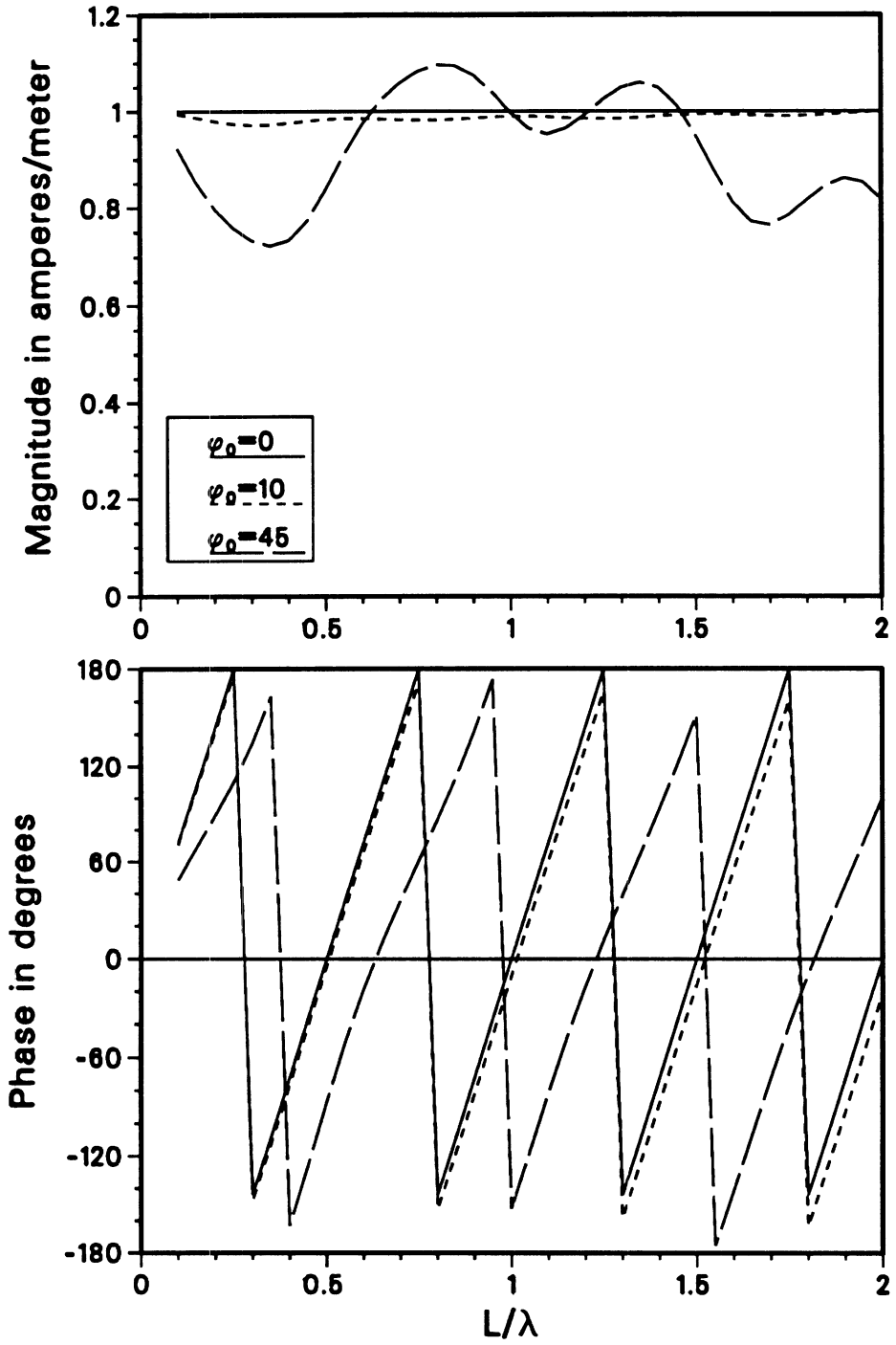

Figure 6: Reflected field H_{z0}^{r} of the infinite array for several ϕ_0 with $d = 0.2\lambda$ and $R_0 = 500\Omega$ (quadratic taper).

References

[1] Kovacs, A. and R. M. Morey, "Radar Anisotropy of Sea Ice Due to Preferred Azimuthal Orientation of the Horizontal *c* Axes of Ice Crystals," *J. Geophys. Res.*, Vol. 83, No. C12, pp. 6037-6046, 1978.

[2] Sarabandi, K., "Scattering from Variable Resistive and Impedance Sheets," *J. Electromagn. Waves Appl.*, Vol. 4, No. 9, pp. 865-891, 1990.

Appendix

The following is the fortran code INFARR listed with its subroutines:

```
C
C     INFARR.FTN                          John R. Natzke
C                                         9/17/92
C
C       This program computes the scattering from an infinite array
C       of parallel resistive sheets of length L and separation d
C       under plane wave illumination with the exp(-iwt) time
C       convention.  The sheets are terminated with a perfectly
C       conducting plane.
C
C       The user is prompted for all of the input parameters, and
C       the output is the specular reflected field written to the
C       files smagdt (magnitude) and sphadt (phase).
C
C
        parameter(ip=2500,ic=4)
        integer EorH,indx(ic),xory
        real k,L,ky
        real RC(ic),R(ip),xp(ip),yp(ip),dxC(ic),dyC(ic),drC(ic)
        complex ci,ctemp,carg,Erz,Hrz,Lzn,Lxn,Lyn,dGxn,dGyn
        complex Z(ip,ip),In(ip),Vm,kx,kxp,Km
        integer ipvt(ip),iretrn
        complex wk(ip)
        data ci,pi/(0.,1.),3.141593/
        open(3,file='smagdt')
        open(4,file='sphadt')
c...Declaring constant values
 10     k=2*pi
        Zo=sqrt(4.e-07*pi/8.854e-12)
        noIter=1
        drg=pi/180
c...Prompting user for input parameters
        print *,'1) E- or 2) H-pol?'
        read *,EorH
        print*,'1) With or 2) without pec termination?'
        read*,ibac
        numC=1
        if(ibac .eq. 1) numC=2
        indx(1)=1
 15     do 20 i=1,numC
          if(i .eq. 1)then
            print*,'For the resisitive strips --'
            xory=1
            print *,' Enter L:'
            read *,L
            W=L
            xm=-W
            ym=0
            print*,' 1) Constant or 2) Tapered Resistivity?'
            read*,ich
            ich1=ich
          else
            print*,'For the back short --'
            ich=1
            xory=2
            xm=-L
            ym=0
            print *,' Enter d:'
            read *,d
            W=d
          endif
c          print *,' Number of elements',int(10*(1-W)+100*W)
c          read *,
          if(W .lt. 1)then
```

```
            N=int(10*(1-W)+100*W)
          else
            N=int(100*W)
          endif
          drC(i)=W/N
          if(xory .eq. 1)then
            dxC(i)=W/N
            dyC(i)=0
          else
            dxC(i)=0
            dyC(i)=W/N
          endif
          if(ich .eq. 1)then
            if(i .eq. 1)then
              print*,' Surface resistivity (ohms):'
              read*,RC(i)
            else
              RC(i)=0
            endif
          else
            RC(i)=0
            rLod=W
            print*,' Maximum resistivity Ro (ohms):'
            read*,Ro
            print *,' Order of taper (1 = linear):'
            read *,itap
          endif
          dxstar=dxC(i)/2
          dystar=dyC(i)/2
          do 18 j=indx(i),indx(i)+N-1
c       Element position arrays
            xp(j)=xm+dxstar+(j-indx(i))*dxC(i)
            yp(j)=ym+dystar+(j-indx(i))*dyC(i)
c       Resistive Loading
            if(ich .eq. 1)then
              R(j)=RC(i)
            else
              if(xory .eq. 1)then
                diff=xp(j)-xm
              else
                diff=yp(j)-ym
              endif
              R(j)=Ro*(diff/rLod)**itap+RC(i)
            endif
 18       continue
          indx(i+1)=indx(i)+N
 20      continue
         if(numC .eq. 1)then
           print *,' Enter d:'
           read *,d
         endif
         M=indx(numC)+N-1
         print*,'   Total number of current elements M =',M
         print*,'Pmax:'
         read*,Pmax
c         print *,'Number of angular iterations:'
c         read *,noIter
c         if(noIter .gt. 1)then
c           star=1*drg
c           fini=pi-star
c           step=(fini-star)/(noIter-1)
c           phi=star
c         endif
         print*,'Incident angle:'
         read*,phio
         phio=phio*drg
         do 30 jp=-8*Pmax,8*Pmax,1
           chk=jp/d-sin(phio)
           if(chk .eq. 1 .or. chk .eq. -1)then
             print*,' SINGULARITY for p =',jp,': Change d or phio.'
             GOTO 15
           endif
```

```
 30     continue
        print *,'Number of length iterations:'
        read *,noIter
        if(noIter .gt. 1)then
          print*,'Final length'
          read*,fini
          wstar=L
          wstep=(fini-wstar)/(noIter-1)
        endif
      DO 700 iter=1,noIter
        print *,' '
        print *, ' M = ',M,' L =',L,' d =',d,' phio =',phio/drg
        print *,' Generating impedance and source matrices . . .'
c*********************************************************************
c*******************   Impedance, Source,   ************************
c******************   and Current Matrices   ***********************
c*********************************************************************
        A=k*Zo/d
        inr=1
        do 250 j=1,M
          xm=xp(j)
          ym=yp(j)
c   Impedance matrix elements
           inc=1
           do 244 i=1,M
           if((inr.eq.inc.and.j.eq.indx(inr))
     &     .or.(inr.eq.inc.and.inc.eq.2.and.i.eq.indx(inc))
     &     .or.(inr.ne.inc))then
             xn=xp(i)
             yn=yp(i)
             dx=dxC(inc)
             dy=dyC(inc)
             dr=drC(inc)
             Z(j,i)=0
             lim=1
             if(i .eq. j) lim=8
             if(EorH .eq. 1)then
c          E-pol
               do 230 jp=-lim*Pmax,lim*Pmax
                 ky=2*pi*jp/d-k*sin(phio)
                 carg=k**2-ky**2
                 kx=csqrt(carg)
                 if(inc .eq. 1)then
                   Lzn=cexp(ci*ky*(ym-yn))/kx**2
                   if(i .eq. j)then
                     Lzn=Lzn*ci*(1-cexp(ci*kx*dx/2))
                   else
                     Lzn=Lzn*cexp(ci*kx*abs(xm-xn))
     &                      *csin(kx*dx/2)
                   endif
                 else
                   Lzn=cexp(ci*(kx*abs(xm-xn)+ky*(ym-yn)))
     &                /kx*dy/2*sinc(ky*dy/2)
                 endif
                 Z(j,i)=Z(j,i)+A*Lzn

230            continue
             else
c          H-pol
               do 240 jp=-lim*Pmax,lim*Pmax
                 ky=2*pi*jp/d-k*sin(phio)
                 carg=k**2-ky**2
                 kx=csqrt(carg)
                 kxp=ci*k*(abs(1.*jp)/d-sgn(1.*jp)*sin(phio))
                 if(inr .eq. 1)then
c                 Exs--
                  if(inc .eq. 1)then
                    Lxn=cexp(ci*ky*ym)/kx**2
                    if(i .eq. j)then
                      Lxn=Lxn*ci*(1-cexp(ci*kx*dx/2))
                    else
                      Lxn=Lxn*cexp(ci*kx*abs(xm-xn))
```

```
     &                    *csin(kx*dx/2)
                        endif
                        dGxn=0
                        xi=xn-dx/2
                        do 232 iend=1,2
                          dGxn=ci/2*sgn(xm-xi)
     &                     *cexp(ci*ky*ym)*(cexp(ci*kx*abs(xm-xi))
     &                     -cexp(ci*kxp*abs(xm-xi)))-dGxn
                          if(jp .eq. 0)then
                            dGxn=ci/2*sgn(xm-xi)
     &                       *cexp(-ci*k*(ym-yn)*sin(phio))
     &                       *(exp(k*abs(xm-xi)*sin(phio))
     &                       /(1-cexp(-k/d*(abs(xm-xi)-ci*(ym-yn))))
     &                       +cexp(-k*(abs(xm-xi)*(sin(phio)+1./d)
     &                       +ci*(ym-yn)/d))/(1-cexp(-k/d*(abs(xm-xi)
     &                       +ci*(ym-yn)))))+dGxn
                          endif
                          xi=xn+dx/2
232                     continue
                        Z(j,i)=Z(j,i)+A*(Lxn-dGxn/k**2)
                      else
                        dGxn=0
                        yi=yn-dy/2
                        do 234 iend=1,2
                          dGxn=ci/2*sgn(xm-xn)
     &                     *cexp(ci*ky*(ym-yi))*(cexp(ci*kx*abs(xm-xn))
     &                     -cexp(ci*kxp*abs(xm-xn)))-dGxn
                          if(jp .eq. 0)then
                            dGxn=ci/2*sgn(xm-xn)
     &                       *cexp(-ci*k*(ym-yi)*sin(phio))
     &                       *(exp(k*abs(xm-xn)*sin(phio))
     &                       /(1-cexp(-k/d*(abs(xm-xn)-ci*(ym-yi))))
     &                       +cexp(-k*(abs(xm-xn)*(sin(phio)+1./d)
     &                       +ci*(ym-yi)/d))/(1-cexp(-k/d*(abs(xm-xn)
     &                       +ci*(ym-yi)))))+dGxn
                          endif
                          yi=yn+dy/2
234                     continue
                        Z(j,i)=Z(j,i)-A*dGxn/k**2
                      endif
                    else
c                     Eys--
                      if(inc .eq. 1)then
                        dGyn=0
                        xi=xn-dx/2
                        do 236 iend=1,2
                         if(jp .ne. 0)then
                          dGyn=ci/2*cexp(ci*ky*(ym-yn))
     &                     *(ky/kx*cexp(ci*kx*abs(xm-xi))+ci
     &                     *sgn(1.*jp)*cexp(ci*kxp*abs(xm-xi)))-dGyn
                         else
                          dGyn=ci/2*(-tan(phio)*cexp(ci*k
     &                     *(abs(xm-xi)*cos(phio)-(ym-yn)*sin(phio)))
     &                     -ci*cexp(-k*(abs(xm-xi)/d+ci*(ym-yn)
     &                     *sin(phio)))*(cexp(k*(abs(xm-xi)*sin(phio)
     &                     +ci*(ym-yn)/d))/(1-cexp(-k/d*(abs(xm-xi)
     &                     -ci*(ym-yn))))-cexp(-k*(abs(xm-xi)
     &                     *sin(phio)+ci*(ym-yn)/d))/(1-cexp(-k/d
     &                     *(abs(xm-xi)+ci*(ym-yn))))))-dGyn
                         endif
                         xi=xn+dx/2
236                     continue
                        Z(j,i)=Z(j,i)-A*dGyn/k**2
                      else
                        Lyn=cexp(ci*(kx*abs(xm-xn)+ky*(ym-yn)))
     &                     /kx*dy/2*sinc(ky*dy/2)
                        dGyn=0
                        yi=yn-dy/2
                        do 238 iend=1,2
                         if(jp .ne. 0)then
                          dGyn=ci/2*cexp(ci*ky*(ym-yi))
     &                     *(ky/kx*cexp(ci*kx*abs(xm-xn))+ci
```

```
     &                  *sgn(1.*jp)*cexp(ci*kxp*abs(xm-xn)))-dGyn
                       else
                        dGyn=ci/2*(-tan(phio)*cexp(ci*k
     &                  *(abs(xm-xn)*cos(phio)-(ym-yi)*sin(phio)))
     &                  -ci*cexp(-k*(abs(xm-xn)/d+ci*(ym-yi)
     &                  *sin(phio)))*(cexp(k*(abs(xm-xn)*sin(phio)
     &                  +ci*(ym-yi)/d))/(1-cexp(-k/d*(abs(xm-xn)
     &                  -ci*(ym-yi))))-cexp(-k*(abs(xm-xn)
     &                  *sin(phio)+ci*(ym-yi)/d))/(1-cexp(-k/d
     &                  *(abs(xm-xn)+ci*(ym-yi))))))-dGyn
                       endif
                       yi=yn+dy/2
238                   continue
                      Z(j,i)=Z(j,i)+A*(Lyn-dGyn/k**2)
                     endif
                    endif
240               continue
                endif
              else
                if(inr .eq. 1)then
                  if(i .lt. j)then
                    Z(j,i)=Z(i,j)
                  else
                    Z(j,i)=Z(j-1,i-1)
                  endif
                else
                  Z(j,i)=Z(j-1,i-1)
                endif
              endif
              if(i .eq. indx(inc+1)-1) inc=inc+1
244           continue
c...Incident Field (Source) matrix elements
            if(EorH .eq. 1)then
              Vm=cexp(-ci*k*(xm*cos(phio)+ym*sin(phio)))
            else
              if(inr .eq. 1)then
                Vm=Zo*sin(phio)
     &               *cexp(-ci*k*(xm*cos(phio)+ym*sin(phio)))
              else
                Vm=Zo*cos(phio)
     &               *cexp(-ci*k*(xm*cos(phio)+ym*sin(phio)))
              endif
            endif
            In(j)=Vm
            if(j .eq. indx(inr+1)-1) inr=inr+1
250       continue
c...Resistivity profile
          do 260 j=1,M
            Z(j,j)=Z(j,j)+R(j)
260       continue
c...Calling subroutines to calculate the current matrix
          print *,' Solving [Znm][In] = [Vm] . . .'
          call CGECO(Z,ip,M,ipvt,cond,wk)
          print *,'     The condition number is ',cond
          call CGESL(Z,ip,M,ipvt,In,0)
c*********************************************************************
c********************   Diffracted Field   ************************
c*********************************************************************
c...Direction of diffracted field(s)
           jp=0
           dowhile(jp .lt. d*(1+sin(phio)))
             print*,' Reflected field exists for p =',jp,
     &         ', phip =',asin(jp/d-sin(phio))/drg,'.'
             jp=jp+1
           enddo
           jp=-1
           dowhile(jp .gt. -d*(1-sin(phio)))
             print*,' Reflected field exists for p =',jp,
     &         ', phip =',asin(jp/d-sin(phio))/drg,'.'
             jp=jp-1
           enddo
c...Specular reflected field
```

```
      print*,' For p = 0, phip =,',-phio/drg,',
      jp=0
      Erz=0
      Hrz=0
      ky=k*(jp/d-sin(phio))
      carg=k**2-ky**2
      kx=csqrt(carg)
      do 610 inc=1,numC
        Lzn=0
        do 600 i=indx(inc),indx(inc+1)-1
          xn=xp(i)
          yn=yp(i)
c          print*,i,xn,yn,cabs(In(i))
          Lzn=In(i)*cexp(-ci*(kx*xn+ky*yn))+Lz
600     continue
        dr=drC(inc)
        if(EorH .eq. 1)then
          if(inc .eq. 1)then
            Erz=-k*Zo/kx/d*csin(kx*dr/2)/kx*Lzn
          else
            Erz=-k*Zo/kx/d*dr/2*sinc(ky*dr/2)*Lzn+Erz
          endif
        else
          if(inc .eq. 1)then
            Hrz=1./d*ky/kx**2*csin(kx*dr/2)*Lzn
          else
            Hrz=-1./d*dr/2*sinc(ky*dr/2)*Lzn+Hrz
          endif
        endif
610     continue
      if(EorH .eq. 1)then
        print *,'   Erz: ',cabs(Erz),pha(Erz)
        write(3,*) L,cabs(Erz)
        write(4,*) L,pha(Erz)
      else
        write(3,*) L,cabs(Hrz)
        write(4,*) L,pha(Hrz)
        print *,'   Hrz: ',cabs(Hrz),pha(Hrz)
      endif

      if(noIter .gt. 1)then
        do 620 i=1,numC
          if(i .eq. 1)then
            ich=ich1
            xory=1
            L=wstar+iter*wstep
            W=L
            xm=-W
            ym=0
          else
            ich=1
            xory=2
            xm=-L
            ym=0
            W=d
          endif
          if(W .lt. 1.)then
            exN=10*(1-W)
          else
            exN=0
          endif
          N=int(exN+100*W)
          drC(i)=W/N
          if(xory .eq. 1)then
            dxC(i)=W/N
            dyC(i)=0
          else
            dxC(i)=0
            dyC(i)=W/N
          endif
          dxstar=dxC(i)/2
          dystar=dyC(i)/2
```

```
              do 618 j=indx(i),indx(i)+N-1
                xp(j)=xm+dxstar+(j-indx(i))*dxC(i)
                yp(j)=ym+dystar+(j-indx(i))*dyC(i)
                if(ich .eq. 1)then
                  R(j)=RC(i)
                else
                  if(xory .eq. 1)then
                    diff=xp(j)-xm
                  else
                    diff=yp(j)-ym
                  endif
                  R(j)=Ro*(diff/W)**itap+RC(i)
                endif
 618            continue
                indx(i+1)=indx(i)+N
 620          continue
              M=indx(numC)+N-1
            endif

c            phio=star+(iter)*step
700       continue

          print *,' Again (1=yes) ? '
          read *,ians
          if(ians .eq. 1) GOTO 10
          print*,' '
          print*,'The output files are SMAGDT and SPHADT.'
800     stop
        END
c*****************************************************************
          FUNCTION SGN(R)
c*****************************************************************
          if(R .ne. 0)then
            sgn=R/abs(R)
          else
            sgn=1.
          endif
          return
          end
c*****************************************************************
          FUNCTION PHA(C)
c*****************************************************************
          complex c
          if(Real(c) .ne. 0)then
            pha=180./3.141593*atan2(aImag(c),Real(c))
          elseif(aImag(c) .ne. 0)then
            pha=aImag(c)/abs(aImag(c))*90
          else
            pha=90
          endif
          return
          end
C*****************************************************************C
C                                                                 C
C     The following subroutines are standard LINPACK routines     C
C     to perform L-U decomposition and back substitution on a     C
C     single precision complex matrix.  See CC-Memo 407 sec 2.1   C
C     for documentation on these routines.                        C
C                                                                 C
C*****************************************************************C
C                                                                 C
      SUBROUTINE CGECO(A,LDA,N,IPVT,RCOND,Z)
C                                                                 C
C*****************************************************************C
C                                                                 C
C NAASA  2.1.042 CGECO    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
      INTEGER LDA,N,IPVT(1)
      COMPLEX A(LDA,1),Z(1)
      REAL RCOND
C
C     CGECO FACTORS A COMPLEX MATRIX BY GAUSSIAN ELIMINATION
```

```
C     AND ESTIMATES THE CONDITION OF THE MATRIX.
C
C     IF  RCOND  IS NOT NEEDED, CGEFA IS SLIGHTLY FASTER.
C     TO SOLVE  A*X = B , FOLLOW CGECO BY CGESL.
C     TO COMPUTE  INVERSE(A)*C , FOLLOW CGECO BY CGESL.
C     TO COMPUTE  DETERMINANT(A) , FOLLOW CGECO BY CGEDI.
C     TO COMPUTE  INVERSE(A) , FOLLOW CGECO BY CGEDI.
C
C     ON ENTRY
C
C        A       COMPLEX(LDA, N)
C                THE MATRIX TO BE FACTORED.
C
C        LDA     INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C     ON RETURN
C
C        A       AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
C                WHICH WERE USED TO OBTAIN IT.
C                THE FACTORIZATION CAN BE WRITTEN  A = L*U  WHERE
C                L  IS A PRODUCT OF PERMUTATION AND UNIT LOWER
C                TRIANGULAR MATRICES AND  U  IS UPPER TRIANGULAR.
C
C        IPVT    INTEGER(N)
C                AN INTEGER VECTOR OF PIVOT INDICES.
C
C        RCOND   REAL
C                AN ESTIMATE OF THE RECIPROCAL CONDITION OF  A .
C                FOR THE SYSTEM  A*X = B , RELATIVE PERTURBATIONS
C                IN  A  AND  B  OF SIZE  EPSILON  MAY CAUSE
C                RELATIVE PERTURBATIONS IN  X  OF SIZE  EPSILON/RCOND .
C                IF  RCOND  IS SO SMALL THAT THE LOGICAL EXPRESSION
C                          1.0 + RCOND .EQ. 1.0
C                IS TRUE, THEN  A  MAY BE SINGULAR TO WORKING
C                PRECISION.  IN PARTICULAR,  RCOND  IS ZERO  IF
C                EXACT SINGULARITY IS DETECTED OR THE ESTIMATE
C                UNDERFLOWS.
C
C        Z       COMPLEX(N)
C                A WORK VECTOR WHOSE CONTENTS ARE USUALLY UNIMPORTANT.
C                IF  A  IS CLOSE TO A SINGULAR MATRIX, THEN  Z  IS
C                AN APPROXIMATE NULL VECTOR IN THE SENSE THAT
C                NORM(A*Z) = RCOND*NORM(A)*NORM(Z) .
C
C     LINPACK. THIS VERSION DATED 07/14/77 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C     SUBROUTINES AND FUNCTIONS
C
C     LINPACK CGEFA
C     BLAS CAXPY,CDOTC,CSSCAL,SCASUM
C     FORTRAN ABS,AIMAG,AMAX1,CMPLX,CONJG,REAL
C
C     INTERNAL VARIABLES
C
      COMPLEX CDOTC,EK,T,WK,WKM
      REAL ANORM,S,SCASUM,SM,YNORM
      INTEGER INFO,J,K,KB,KP1,L
C
      COMPLEX ZDUM,ZDUM1,ZDUM2,CSIGN1
      REAL CABS1
      CABS1(ZDUM) = ABS(REAL(ZDUM)) + ABS(AIMAG(ZDUM))
      CSIGN1(ZDUM1,ZDUM2) = CABS1(ZDUM1)*(ZDUM2/CABS1(ZDUM2))
C
CCC   Compute 1-NORM of A
C
      ANORM = 0.0E0
      DO 10 J = 1, N
```

```
      ANORM = AMAX1(ANORM,SCASUM(N,A(1,J),1))
   10 CONTINUE
C
CCC   Factor
C
      CALL CGEFA(A,LDA,N,IPVT,INFO)
C
C     RCOND = 1/(NORM(A)*(ESTIMATE OF NORM(INVERSE(A)))) .
C     ESTIMATE = NORM(Z)/NORM(Y) WHERE  A*Z = Y  AND  CTRANS(A)*Y = E .
C     CTRANS(A)  IS THE CONJUGATE TRANSPOSE OF A .
C     THE COMPONENTS OF  E  ARE CHOSEN TO CAUSE MAXIMUM LOCAL
C     GROWTH IN THE ELEMENTS OF W  WHERE  CTRANS(U)*W = E .
C     THE VECTORS ARE FREQUENTLY RESCALED TO AVOID OVERFLOW.
C
C     SOLVE CTRANS(U)*W = E
C
      EK = CMPLX(1.0E0,0.0E0)
      DO 20 J = 1, N
         Z(J) = CMPLX(0.0E0,0.0E0)
   20 CONTINUE
      DO 100 K = 1, N
         IF (CABS1(Z(K)) .NE. 0.0E0) EK = CSIGN1(EK,-Z(K))
         IF (CABS1(EK-Z(K)) .LE. CABS1(A(K,K))) GO TO 30
            S = CABS1(A(K,K))/CABS1(EK-Z(K))
            CALL CSSCAL(N,S,Z,1)
            EK = CMPLX(S,0.0E0)*EK
   30    CONTINUE
         WK = EK - Z(K)
         WKM = -EK - Z(K)
         S = CABS1(WK)
         SM = CABS1(WKM)
         IF (CABS1(A(K,K)) .EQ. 0.0E0) GO TO 40
            WK = WK/CONJG(A(K,K))
            WKM = WKM/CONJG(A(K,K))
         GO TO 50
   40    CONTINUE
            WK = CMPLX(1.0E0,0.0E0)
            WKM = CMPLX(1.0E0,0.0E0)
   50    CONTINUE
         KP1 = K + 1
         IF (KP1 .GT. N) GO TO 90
            DO 60 J = KP1, N
               SM = SM + CABS1(Z(J)+WKM*CONJG(A(K,J)))
               Z(J) = Z(J) + WK*CONJG(A(K,J))
               S = S + CABS1(Z(J))
   60       CONTINUE
            IF (S .GE. SM) GO TO 80
               T = WKM - WK
               WK = WKM
               DO 70 J = KP1, N
                  Z(J) = Z(J) + T*CONJG(A(K,J))
   70          CONTINUE
   80       CONTINUE
   90    CONTINUE
         Z(K) = WK
  100 CONTINUE
      S = 1.0E0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
C
CCC   Solve CTRANS(L)*Y = V
C
      DO 120 KB = 1, N
         K = N + 1 - KB
         IF (K .LT. N) Z(K) = Z(K) + CDOTC(N-K,A(K+1,K),1,Z(K+1),1)
         IF (CABS1(Z(K)) .LE. 1.0E0) GO TO 110
            S = 1.0E0/CABS1(Z(K))
            CALL CSSCAL(N,S,Z,1)
  110    CONTINUE
         L = IPVT(K)
         T = Z(L)
         Z(L) = Z(K)
         Z(K) = T
```

```
  120 CONTINUE
      S = 1.0E0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
C
      YNORM = 1.0E0
C
CCC   Solve L*V = Y
C
      DO 140 K = 1, N
         L = IPVT(K)
         T = Z(L)
         Z(L) = Z(K)
         Z(K) = T
         IF (K .LT. N) CALL CAXPY(N-K,T,A(K+1,K),1,Z(K+1),1)
         IF (CABS1(Z(K)) .LE. 1.0E0) GO TO 130
            S = 1.0E0/CABS1(Z(K))
            CALL CSSCAL(N,S,Z,1)
            YNORM = S*YNORM
  130    CONTINUE
  140 CONTINUE
      S = 1.0E0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
      YNORM = S*YNORM
C
CCC   Solve U*Z = V
C
      DO 160 KB = 1, N
         K = N + 1 - KB
         IF (CABS1(Z(K)) .LE. CABS1(A(K,K))) GO TO 150
            S = CABS1(A(K,K))/CABS1(Z(K))
            CALL CSSCAL(N,S,Z,1)
            YNORM = S*YNORM
  150    CONTINUE
         IF (CABS1(A(K,K)) .NE. 0.0E0) Z(K) = Z(K)/A(K,K)
         IF (CABS1(A(K,K)) .EQ. 0.0E0) Z(K) = CMPLX(1.0E0,0.0E0)
         T = -Z(K)
         CALL CAXPY(K-1,T,A(1,K),1,Z(1),1)
  160 CONTINUE
C     MAKE ZNORM = 1.0
      S = 1.0E0/SCASUM(N,Z,1)
      CALL CSSCAL(N,S,Z,1)
      YNORM = S*YNORM
C
      IF (ANORM .NE. 0.0E0) RCOND = YNORM/ANORM
      IF (ANORM .EQ. 0.0E0) RCOND = 0.0E0
      RETURN
      END
C*****************************************************************C
C                                                                 C
      SUBROUTINE CGEFA(A,LDA,N,IPVT,INFO)
C                                                                 C
C*****************************************************************C
C                                                                 C
C NAASA  2.1.043 CGEFA    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
      INTEGER LDA,N,IPVT(1),INFO
      COMPLEX A(LDA,1)
C
C     CGEFA FACTORS A COMPLEX MATRIX BY GAUSSIAN ELIMINATION.
C
C     CGEFA IS USUALLY CALLED BY CGECO, BUT IT CAN BE CALLED
C     DIRECTLY WITH A SAVING IN TIME IF  RCOND  IS NOT NEEDED.
C     (TIME FOR CGECO) = (1 + 9/N)*(TIME FOR CGEFA) .
C
C     ON ENTRY
C
C        A       COMPLEX(LDA, N)
C                THE MATRIX TO BE FACTORED.
C
C        LDA     INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
```

```
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C     ON RETURN
C
C        A       AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
C                WHICH WERE USED TO OBTAIN IT.
C                THE FACTORIZATION CAN BE WRITTEN  A = L*U  WHERE
C                L  IS A PRODUCT OF PERMUTATION AND UNIT LOWER
C                TRIANGULAR MATRICES AND  U  IS UPPER TRIANGULAR.
C
C        IPVT    INTEGER(N)
C                AN INTEGER VECTOR OF PIVOT INDICES.
C
C        INFO    INTEGER
C                = 0  NORMAL VALUE.
C                = K  IF  U(K,K) .EQ. 0.0 .  THIS IS NOT AN ERROR
C                     CONDITION FOR THIS SUBROUTINE, BUT IT DOES
C                     INDICATE THAT CGESL OR CGEDI WILL DIVIDE BY ZERO
C                     IF CALLED.  USE  RCOND  IN CGECO FOR A RELIABLE
C                     INDICATION OF SINGULARITY.
C
C     LINPACK. THIS VERSION DATED 07/14/77 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS CAXPY,CSCAL,ICAMAX
C     FORTRAN ABS,AIMAG,CMPLX,REAL
C
C     INTERNAL VARIABLES
C
      COMPLEX T
      INTEGER ICAMAX,J,K,KP1,L,NM1
C
      COMPLEX ZDUM
      REAL CABS1
      CABS1(ZDUM) = ABS(REAL(ZDUM)) + ABS(AIMAG(ZDUM))
C
CCC   Gaussian elimination with partial pivoting
C
      INFO = 0
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 70
      DO 60 K = 1, NM1
         KP1 = K + 1
C
C        FIND L = PIVOT INDEX
C
         L = ICAMAX(N-K+1,A(K,K),1) + K - 1
         IPVT(K) = L
C
CCC      Zero pivot implies this column already triangularized
C
         IF (CABS1(A(L,K)) .EQ. 0.0E0) GO TO 40
C
CCC         Interchange if necessary
C
            IF (L .EQ. K) GO TO 10
               T = A(L,K)
               A(L,K) = A(K,K)
               A(K,K) = T
   10       CONTINUE
C
CCC         Compute multipliers
C
            T = -CMPLX(1.0E0,0.0E0)/A(K,K)
            CALL CSCAL(N-K,T,A(K+1,K),1)
C
CCC         Row elimination with column indexing
C
            DO 30 J = KP1, N
```

```
               T = A(L,J)
               IF (L .EQ. K) GO TO 20
                  A(L,J) = A(K,J)
                  A(K,J) = T
   20          CONTINUE
               CALL CAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
   30       CONTINUE
         GO TO 50
   40    CONTINUE
            INFO = K
   50    CONTINUE
   60 CONTINUE
   70 CONTINUE
      IPVT(N) = N
      IF (CABS1(A(N,N)) .EQ. 0.0E0) INFO = N
      RETURN
      END
C                                                                      C
C**********************************************************************C
C                                                                      C
      SUBROUTINE CGESL(A,LDA,N,IPVT,B,JOB)
C                                                                      C
C**********************************************************************C
C                                                                      C
C NAASA  2.1.044 CGESL    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
      INTEGER LDA,N,IPVT(1),JOB
      COMPLEX A(LDA,1),B(1)
C
C     CGESL SOLVES THE COMPLEX SYSTEM
C     A * X = B  OR  CTRANS(A) * X = B
C     USING THE FACTORS COMPUTED BY CGECO OR CGEFA.
C
C     ON ENTRY
C
C        A       COMPLEX(LDA, N)
C                THE OUTPUT FROM CGECO OR CGEFA.
C
C        LDA     INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C        IPVT    INTEGER(N)
C                THE PIVOT VECTOR FROM CGECO OR CGEFA.
C
C        B       COMPLEX(N)
C                THE RIGHT HAND SIDE VECTOR.
C
C        JOB     INTEGER
C                = 0         TO SOLVE  A*X = B ,
C                = NONZERO   TO SOLVE  CTRANS(A)*X = B  WHERE
C                            CTRANS(A)  IS THE CONJUGATE TRANSPOSE.
C
C     ON RETURN
C
C        B       THE SOLUTION VECTOR  X .
C
C     ERROR CONDITION
C
C        A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS A
C        ZERO ON THE DIAGONAL.  TECHNICALLY THIS INDICATES SINGULARITY
C        BUT IT IS OFTEN CAUSED BY IMPROPER ARGUMENTS OR IMPROPER
C        SETTING OF LDA .  IT WILL NOT OCCUR IF THE SUBROUTINES ARE
C        CALLED CORRECTLY AND IF CGECO HAS SET RCOND .GT. 0.0
C        OR CGEFA HAS SET INFO .EQ. 0 .
C
C     TO COMPUTE  INVERSE(A) * C  WHERE  C  IS A MATRIX
C     WITH  P  COLUMNS
C           CALL CGECO(A,LDA,N,IPVT,RCOND,Z)
C           IF (RCOND IS TOO SMALL) GO TO ...
```

```
C           DO 10 J = 1, P
C              CALL CGESL(A,LDA,N,IPVT,C(1,J),0)
C        10 CONTINUE
C
C     LINPACK. THIS VERSION DATED 07/14/77 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LABS.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS CAXPY,CDOTC
C     FORTRAN CONJG
C
C     INTERNAL VARIABLES
C
      COMPLEX CDOTC,T
      INTEGER K,KB,L,NM1
C
      NM1 = N - 1
      IF (JOB .NE. 0) GO TO 50
C
C        JOB = 0 , SOLVE  A * X = B
C        FIRST SOLVE  L*Y = B
C
         IF (NM1 .LT. 1) GO TO 30
         DO 20 K = 1, NM1
            L = IPVT(K)
            T = B(L)
            IF (L .EQ. K) GO TO 10
               B(L) = B(K)
               B(K) = T
   10       CONTINUE
            CALL CAXPY(N-K,T,A(K+1,K),1,B(K+1),1)
   20    CONTINUE
   30    CONTINUE
C
C        NOW SOLVE  U*X = Y
C
         DO 40 KB = 1, N
            K = N + 1 - KB
            B(K) = B(K)/A(K,K)
            T = -B(K)
            CALL CAXPY(K-1,T,A(1,K),1,B(1),1)
   40    CONTINUE
      GO TO 100
   50 CONTINUE
C
C        JOB = NONZERO, SOLVE  CTRANS(A) * X = B
C        FIRST SOLVE  CTRANS(U)*Y = B
C
         DO 60 K = 1, N
            T = CDOTC(K-1,A(1,K),1,B(1),1)
            B(K) = (B(K) - T)/CONJG(A(K,K))
   60    CONTINUE
C
C        NOW SOLVE CTRANS(L)*X = Y
C
         IF (NM1 .LT. 1) GO TO 90
         DO 80 KB = 1, NM1
            K = N - KB
            B(K) = B(K) + CDOTC(N-K,A(K+1,K),1,B(K+1),1)
            L = IPVT(K)
            IF (L .EQ. K) GO TO 70
               T = B(L)
               B(L) = B(K)
               B(K) = T
   70       CONTINUE
   80    CONTINUE
   90    CONTINUE
  100 CONTINUE
      RETURN
      END
C                                                                       C
```

```
C*********************************************************************C
C                                                                     C
      SUBROUTINE CAXPY(N,CA,CX,INCX,CY,INCY)
C                                                                     C
C*********************************************************************C
C                                                                     C
C NAASA  1.1.014 CAXPY    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
C     CONSTANT TIMES A VECTOR PLUS A VECTOR.
C     JACK DONGARRA, LINPACK, 6/17/77.
C
      COMPLEX CX(1),CY(1),CA
      INTEGER I,INCX,INCY,IX,IY,N
C
      IF(N.LE.0)RETURN
      IF (ABS(REAL(CA)) + ABS(AIMAG(CA)) .EQ. 0.0 ) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GOTO 20
C
CCC      Code for unequal increments or equal increments
CCC      Not equal to 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        CY(IY) = CY(IY) + CA*CX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
CCC      Code for both increments equal to 1
C
   20 DO 30 I = 1,N
        CY(I) = CY(I) + CA*CX(I)
   30 CONTINUE
      RETURN
      END
C                                                                     C
C*********************************************************************C
      COMPLEX FUNCTION CDOTC(N,CX,INCX,CY,INCY)
C*********************************************************************C
C                                                                     C
C NAASA  1.1.012 CDOTC    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
C     FORMS THE DOT PRODUCT OF TWO VECTORS, CONJUGATING THE FIRST
C     VECTOR.
C     JACK DONGARRA, LINPACK,  6/17/77.
C
      COMPLEX CX(1),CY(1),CTEMP
      INTEGER I,INCX,INCY,IX,IY,N
C
      CTEMP = (0.0,0.0)
      CDOTC = (0.0,0.0)
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GOTO 20
C
CCC      Code for unequal increments or equal increments
CCC      Not equal to 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        CTEMP = CTEMP + CONJG(CX(IX))*CY(IY)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      CDOTC = CTEMP
      RETURN
```

```
C
CCC      Code for both increments equal to 1
C
   20 DO 30 I = 1,N
        CTEMP = CTEMP + CONJG(CX(I))*CY(I)
   30 CONTINUE
      CDOTC = CTEMP
      RETURN
      END
C                                                                      C
C**********************************************************************C
C                                                                      C
      SUBROUTINE  CSCAL(N,CA,CX,INCX)
C                                                                      C
C**********************************************************************C
C                                                                      C
C NAASA  1.1.019 CSCAL    FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
C     SCALES A VECTOR BY A CONSTANT.
C     JACK DONGARRA, LINPACK,  6/17/77.
C
      COMPLEX CA,CX(1)
      INTEGER I,INCX,N,NINCX
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
CCC      Code for increment not equal to 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        CX(I) = CA*CX(I)
   10 CONTINUE
      RETURN
C
CCC      Code for increment equal to 1
C
   20 DO 30 I = 1,N
        CX(I) = CA*CX(I)
   30 CONTINUE
      RETURN
      END
C                                                                      C
C**********************************************************************C
C                                                                      C
      SUBROUTINE  CSSCAL(N,SA,CX,INCX)
C                                                                      C
C**********************************************************************C
C                                                                      C
C NAASA  1.1.018 CSSCAL   FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
C     SCALES A COMPLEX VECTOR BY A REAL CONSTANT.
C     JACK DONGARRA, LINPACK, 6/17/77.
C
      COMPLEX CX(1)
      REAL SA
      INTEGER I,INCX,N,NINCX
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
CCC      Code for increment not equal to 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        CX(I) = CMPLX(SA*REAL(CX(I)),SA*AIMAG(CX(I)))
   10 CONTINUE
      RETURN
C
CCC      Code for increment equal to 1
C
   20 DO 30 I = 1,N
```

```
        CX(I) = CMPLX(SA*REAL(CX(I)),SA*AIMAG(CX(I)))
   30 CONTINUE
      RETURN
      END
C                                                                      C
C**********************************************************************C
      INTEGER FUNCTION ICAMAX(N,CX,INCX)
C**********************************************************************C
C                                                                      C
C NAASA  1.1.021 ICAMAX   FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
C     FINDS THE INDEX OF ELEMENT HAVING MAX. ABSOLUTE VALUE.
C     JACK DONGARRA, LINPACK, 6/17/77.
C
      COMPLEX CX(1)
      REAL SMAX
      INTEGER I,INCX,IX,N
      COMPLEX ZDUM
      REAL CABS1
      CABS1(ZDUM) = ABS(REAL(ZDUM)) + ABS(AIMAG(ZDUM))
C
      ICAMAX = 1
      IF(N.LE.1)RETURN
      IF(INCX.EQ.1)GOTO 20
C
CCC      Code for increment not equal to 1
C
      IX = 1
      SMAX = CABS1(CX(1))
      IX = IX + INCX
      DO 10 I = 2,N
         IF(CABS1(CX(IX)).LE.SMAX) GO TO 5
         ICAMAX = I
         SMAX = CABS1(CX(IX))
    5    IX = IX + INCX
   10 CONTINUE
      RETURN
C
CCC      Code for increment equal to 1
C
   20 SMAX = CABS1(CX(1))
      DO 30 I = 2,N
         IF(CABS1(CX(I)).LE.SMAX) GO TO 30
         ICAMAX = I
         SMAX = CABS1(CX(I))
   30 CONTINUE
      RETURN
      END
C                                                                      C
C**********************************************************************C
      REAL FUNCTION SCASUM(N,CX,INCX)
C**********************************************************************C
C                                                                      C
C NAASA  1.1.010 SCASUM   FTN-A 05-02-78     THE UNIV OF MICH COMP CTR
C
C     TAKES THE SUM OF THE ABSOLUTE VALUES OF A COMPLEX VECTOR AND
C     RETURNS A SINGLE PRECISION RESULT.
C     JACK DONGARRA, LINPACK, 6/17/77.
C
      COMPLEX CX(1)
      REAL STEMP
      INTEGER I,INCX,N,NINCX
C
      SCASUM = 0.0E0
      STEMP = 0.0E0
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GOTO 20
C
CCC      Code for increment not equal to 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
```

```
         STEMP = STEMP + ABS(REAL(CX(I))) + ABS(AIMAG(CX(I)))
   10 CONTINUE
      SCASUM = STEMP
      RETURN
C
CCC      Code for increment equal to 1
C
   20 DO 30 I = 1,N
         STEMP = STEMP + ABS(REAL(CX(I))) + ABS(AIMAG(CX(I)))
   30 CONTINUE
      SCASUM = STEMP
      RETURN
      END
```