C

Fortran Codes for TM Scattering From Perfect Electric Conducting Cylinders

This appendix illustrates three distinct approaches for the implementation of the pulse basis–delta testing discretization of the EFIE for the TM scattering problem of Section 2.1. Section C.1 provides the specific implementation discussed in Section 2.1, which involved a "single-point" approximation for the off-diagonal matrix entries. Section C.2 illustrates the use of Romberg quadrature with the "singularity subtraction" approach for treating the diagonal matrix elements, as described at the end of Section 2.1. Finally, Section C.3 illustrates the use of a generalized Gaussian quadrature procedure (Section A.4) that can evaluate the singular diagonal matrix entries without special treatment.

The following FORTRAN subroutines are intended for illustration and generally are far from optimal. For instance, it should be more efficient to separately evaluate the real and imaginary parts of the matrix entries and reserve the use of the specialized Gaussian quadrature rule for the singular integrands only.

C.1 IMPLEMENTATION 1: SINGLE-POINT APPROXIMATION

The following FORTRAN computer program illustrates the specific implementation discussed in Section 2.1. The geometry is described by the input data file (CYLFIL), and some additional information is provided through the keyboard. A sample input file and a sample output file are provided after the program listing.

```
c     ptmege      moment-method analysis of pec cylinders for tm-pol
c                 plane-wave excitation -- program finds the current
c                 density and rcs
```

```
c
c                the E-field equation is discretized with pulse basis
c                functions and point-matching (approximate formulas
c                are taken from Harrington, Field Computation by
c                Moment Methods, Krieger reprint, 1982, pp 42 - 46)
c
c     a. f. peterson   feb 11, 1986
c
c     description of input file:
c
c     the model of the cylinder cross-section is read from file
c     'cylfil,' which should be free-formatted and contain the
c     following:
c
c       N
c       x(1) y(1) w(1)
c       x(2) y(2) w(2)
c       .
c       .
c       .
c       x(N) y(N) w(N)
c
c     where
c
c       N is the number of strips in the model
c       (x,y) is the location of the phase center of each strip in the
c             x-y plane
c       w is the width of each strip
c       (x, y, and w are in units of free-space wavelength)
c
c
c     parameter 'na' is the max order of the system matrix and thus
c     the largest size model that can be treated -- adjust accordingly
c
      parameter(na=300)
      complex z(na,na),v(na),j(na),sig,ezpjz
      real x(na),y(na),w(na),theta,arg,mag,rcs,phi,psi
      integer p(na)
c
      open(unit=1,file='cylfil')
      open(unit=4,file='outfil',status='new')
c
c     load model from file #1
c
      read(1,*) n
      do 10 i=1,n
   10 read(1,*) x(i),y(i),w(i)
c
```

```
c     fill impedance matrix
c
      do 20 k=1,n
      do 20 i=1,n
      z(k,i)=ezpjz(x(i),y(i),x(k),y(k),w(i))
   20 continue
      ijob = 0
c
c     generate incident field vector v
c
   30 write(*,*) 'give direction of incident plane wave in degrees'
      write(*,*) '(theta=0. means a wave propagating in the +x'
      write(*,*) 'direction; theta=90. means +y direction, relative'
      write(*,*) 'to the model)'
      read(*,*) theta
      write(4,*)
      write(4,*) 'angle of incidence is ',theta,' degrees'
      theta = .01745329*theta
      do 40 i=1,n
      arg=-6.2831853*(y(i)*sin(theta)+x(i)*cos(theta))
   40 v(i) = cmplx(cos(arg),sin(arg))
c
c     solve system by gaussian elimination
c
      call cgauss(z,na,n,v,j,ijob,p)
      if(ijob.eq.-1) stop
      ijob = 1
c
c     write data to file #4
c
      write(4,*)
      write(4,*) 'current density in magnitude - angle format'
      write(4,*)
      do 60 i=1,n
      mag=cabs(j(i))
      arg=atan2(aimag(j(i)),real(j(i)))*57.2957795
   60 write(4,*) i,mag,arg
      write(4,*)
c
c     compute bistatic scattering cross section
c
      write(*,*) 'give incremental angle for rsc scan in degrees'
      read(*,*) theta
      write(4,*) 'rcs in dB-wavelength'
      write(4,*) '     angle              rcs'
      write(4,*)
      psi=0.
   70 phi=psi*.01745329
```

```
      sig = (0.,0.)
      do 80 i=1,n
   80 sig = sig + w(i)*cexp(cmplx(0.,6.2831853*(x(i)*cos(phi)+
     +                y(i)*sin(phi))))*j(i)
      rcs = 10.*alog10(cabs(sig)**2 * 222936.04)
      write(4,*) psi,rcs
      psi = psi + theta
      if (psi.lt.360.) go to 70
c
c     check for additional incident angles
c
      write(*,*) 'want additional incident angles (1=y)'
      read(*,*) i
      if(i.eq.1) go to 30
      write(*,*)
      write(*,*) 'data from this run placed in file: outfil'
      stop
      end
c
c     ------------------------------------------------
c
      complex function ezpjz(xs,ys,xo,yo,ws)
c
c     negative scattered electric field (Ez) at location
c     (xo,yo) due to a unit source (Jz) of width ws at (xs,ys)
c
      complex h02
      wsot=ws*0.1
      arg=sqrt((xo-xs)**2+(yo-ys)**2)
         if(arg.lt.wsot) then
c
c        assume observer at center of source cell
c
         ezpjz=591.7661*ws*cmplx(1.,-.636619772*alog(1.029217*ws))
c
         else
c
c        observer not on source cell; use single-point approximation
c
         arg=arg*6.283185308
         ezpjz=591.7661*ws*h02(arg)
         endif
      return
      end
c
c     ------------------------------------------------
c
c     ZERO-ORDER HANKEL FUNCTION (2ND KIND) FROM ABR & STEGUN PP 369
```

```
c
      COMPLEX FUNCTION H02(A)
      IF(A.GT.3.000) GOTO 5
      BJ=(A/3.00)**2
      BJ=1.0+BJ*(-2.2499997+BJ*(1.2656208+BJ*(-.3163866+BJ*
     &    (.0444479+BJ*(-.0039444+BJ*.00021)))))
      BY=(A/3.00)**2
      BY=2.0/3.1415926*ALOG(A/2.)*BJ+.36746691+BY*(.60559366+BY
     &    *(-.74350384+BY*(.25300117+BY*(-.04261214+BY*(.00427916-
     &    BY*.00024846)))))
      GOTO 10
    5 BJ=3.00/A
      F0=.79788456+BJ*(-.00000077+BJ*(-.00552740+BJ*(-.00009512
     &    +BJ*(.00137237+BJ*(-.00072805+BJ*.00014476)))))
      T0=A-.78539816+BJ*(-.04166397+BJ*(-.00003954+BJ*(.00262573
     &    +BJ*(-.00054125+BJ*(-.00029333+BJ*.00013558)))))
      BY=SQRT(A)
      BJ=F0*COS(T0)/BY
      BY=F0*SIN(T0)/BY
   10 H02=CMPLX(BJ,-BY)
      RETURN
      END
c
c     ------------------------------------------------
c
      subroutine cgauss(a,ia,n,b,x,ijob,p)
c
c     gaussian elimination to solve the complex system ax=b
c
c     (perform forward elimination with scaled partial pivoting on the
c     matrix 'a,' then use back substitution to solve)
c
c     the right-hand side 'b' is destroyed in the process
c
c     ia = row dimension of 'a' as specified in the calling program
c     n = order of the system
c     ijob = 0  means to perform elimination and solve the system
c     ijob = 1  means that forward elimination has already been
c               performed on 'a' by a previous call, so just solve
c     ijob = -1  on return means that 'a' is singular
c     p = array to hold pivot indices
c
c     A. F. Peterson    Feb 11 1986
c     (adapted from the text 'numerical mathematics and computing'
c     by cheney and kincaid,  brooks / cole publishing, 1980)
c
      complex a(ia,n),b(n),x(n),sum
      real c,cmax
```

```
      integer p(n),pk,ia,n,ijob,i,j,k
c
      if(ijob.eq.1) go to 5
c
c     initialize pivot indices and scale factors used when pivoting
c
      do 1 i = 1,n
      p(i) = i
      x(i) = (0.,0.)
      do 1 j = 1,n
    1 x(i) = cmplx(amax1(cabs(x(i)),cabs(a(i,j))),0.)
c
c     loop through one column at a time, sweeping out
c
      do 4 k = 1,n-1
c
c     find the maximum entry of column 'k,' (each normalized to row
c     maximum), ignoring those rows that have already been assigned
c     pivots
c
      cmax = 0.
      do 2 i = k,n
      c=cabs(a(p(i),k))/cabs(x(p(i)))
      if(c.le.cmax) go to 2
      j = i
      cmax = c
    2 continue
c
      if(cmax) 9,9,3
c
c     set pivot index for column 'k'
c
    3 pk = p(j)
      p(j) = p(k)
      p(k) = pk
c
c     perform forward elimination, storing multipliers in 'a'
c
      do 4 i = k+1,n
      sum = a(p(i),k)/a(pk,k)
      a(p(i),k) = sum
      do 4 j = k+1,n
    4 a(p(i),j) = a(p(i),j) - sum*a(pk,j)
c
c     perform forward elimination on the right-hand side 'b'
c
    5 do 6 j = 1,n-1
      do 6 i = j+1,n
```

```
    6 b(p(i)) = b(p(i)) -a(p(i),j)*b(p(j))
c
c     solve by back substitution
c
      x(n) = b(p(n))/a(p(n),n)
      do 8 i = 1,n-1
      sum = b(p(n-i))
      do 7 j = n-i+1,n
    7 sum = sum - a(p(n-i),j)*x(j)
    8 x(n-i) = sum/a(p(n-i),n-i)
      return
    9 write(*,*) 'matrix is singular'
      ijob = -1
      return
      end
```

Sample input data (CYLFIL) for circular cylinder,
one wavelength circumference:

```
          10
   0.1591549       0.0000000       0.1000000
   0.1287590       9.3548924E-02   0.1000000
   4.9181573E-02   0.1513653       0.1000000
  -4.9181588E-02   0.1513653       0.1000000
  -0.1287591       9.3548916E-02   0.1000000
  -0.1591549      -1.3913767E-08   0.1000000
  -0.1287590      -9.3548939E-02   0.1000000
  -4.9181599E-02  -0.1513653       0.1000000
   4.9181603E-02  -0.1513653       0.1000000
   0.1287591      -9.3548872E-02   0.1000000
```

Corresponding output data: (OUTFIL)

```
 angle of incidence is   0.0000000     degrees

 current density in magnitude - angle format

 1   8.1690575E-04   152.8625
 2   9.1447576E-04  -143.4977
 3   2.1823230E-03  -69.01913
 4   3.8987177E-03  -12.58591
 5   5.6035798E-03   27.17590
 6   6.3566919E-03   41.39040
 7   5.6035747E-03   27.17587
```

```
 8   3.8987170E-03  -12.58589
 9   2.1823235E-03  -69.01912
10   9.1447704E-04  -143.4976

 rcs in dB-wavelength
      angle              rcs

   0.0000000       2.831829
    30.00000       2.015279
    60.00000      -0.033761
    90.00000      -1.874606
    120.0000      -2.312734
    150.0000      -2.139701
    180.0000      -2.048646
    210.0000      -2.139700
    240.0000      -2.312733
    270.0000      -1.874606
    300.0000      -0.033763
    330.0000       2.015276
-------------------------------------------------------------
```

C.2 IMPLEMENTATION 2: ROMBERG QUADRATURE

In this example, subroutine EZPJZ is replaced by a routine that employs Romberg integration (Section A.1) to evaluate the matrix entries. An additional parameter Ω_n is required to describe the orientation of each cell, as defined in Section 2.2. In addition, the desired integration accuracy *rerr* is provided from the main routine.

```
      complex function ezpjz(xs,ys,xo,yo,ws,omegs,rerr)
c
c     negative scattered electric field (Ez) at location
c     (xo,yo) due to a unit source (Jz) of width ws and
c     orientation omegs at (xs,ys)
c
c     computed using Romberg quadrature to accuracy rerr
c
      complex zrom1d,zint1,zint2,r(15)
      common xc,yc,com,som
      external zint1,zint2
c
      aerr=0.000001
      wso2=0.5*ws
      wsot=0.1*ws
      arg=sqrt((xo-xs)**2+(yo-ys)**2)
      if(arg.lt.wsot) then
```

```
c
c        assume observer at center of source cell; add singular
c        part of integral (missing from zint1) back in
c
         ezpjz=zrom1d(zint1,0.,wso2,rerr,aerr,15,1,r,ier)+
     +   cmplx(0.,-.636619772*alog(2.058434105*wso2))*wso2
         ezpjz=ezpjz*(2.,0.)
c
         else
c
c        observer not on source cell
c
         xc=xo-xs
         yc=yo-ys
         com=cos(omegs)
         som=sin(omegs)
         ezpjz=zrom1d(zint2,-wso2,wso2,rerr,aerr,15,1,r,ier)
         endif
      ezpjz=591.7661*ezpjz
      return
      end
c
c     ------------------------------------------------------
c
      complex function zint1(u)
c
c     the complex-valued integrand for the electric field
c     (Ez) produced by a unit source (Jz) when the observer
c     IS located on the source cell
c
c     (the singular part of the integrand is removed)
c
      complex h02
c
      arg=6.283185308*u
      if(arg.lt.0.00001) then
      zint1=(1.,0.)
      else
      zint1=h02(arg)+(0.,.6366197723)*alog(5.595404025*u)
      endif
      return
      end
c
c     ------------------------------------------------------
c
      complex function zint2(u)
c
c     the complex-valued integrand for the electric field
```

```
c     (Ez) produced by a unit source (Jz) when the observer
c     is NOT located on the source cell
c
      complex h02
      common xc,yc,com,som
c
      delx=xc-u*com
      dely=yc-u*som
      arg=6.283185308*sqrt((delx)**2+(dely)**2)
      zint2=h02(arg)
      return
     end
c
c     ------------------------------------------------------
c
      complex function zrom1d(f,a,b,rerr,aerr,maxp,minp,r,ier)
c
c  returns estimate of the integral of a complex-valued function 'f'
c  over the interval (a,b) on the real axis -- Romberg integration
c
c     'f' is an external complex-valued function of the form 'f(x)'
c         where 'x' is real-valued
c         ('f' must be declared 'external' in the calling program)
c  'rerr' is the desired relative accuracy (rerr must be between
c         0.00001 and 1.0)
c  'aerr' is the absolute error desired -- if the estimate falls
c         below this amount, the algorithm will terminate
c  'maxp' specifies the maximum number of function evaluations
c         to be (1+2**(maxp-1))
c  'minp' specifies the minimum number of function evaluations
c         to be (1+2**(minp-1)) -- this may be necessary if the
c         integrand oscillates several times within the interval.
c     'r' is a complex array of length 'maxp' (workspace provided by
c         main routine). On return, contains the last row of the
c         Romberg extrapolation array.
c   'ier' on return is set to zero if accuracy met
c         on return is 1 if rerr is out of range
c         on return is 2 if maxp is less than 1 or minp .gt. maxp
c         on return is 3 if accuracy is not met
c
c  author: A. F. Peterson
c
c  last date revised: Dec 18, 1986
c
      complex r(maxp),f,zsum,rlast
      if((maxp.lt.1) .or. (minp.gt.maxp)) then
         ier=2
         zrom1d=(0.,0.)
```

```
            go to 30
            endif
         if((rerr.lt.0.00001).or.(rerr.gt.1.)) then
            ier=1
            zromld=(0.,0.)
            go to 30
            endif
         ier=0
         zsum=0.5*(f(a)+f(b))
         r(1)=zsum*(b-a)
         n=1
c
c  refine estimate of integral using trapezoid rule
c
         do 20 k=2,maxp
         rlast=r(1)
         del=(b-a)/float(n)
         x=a-0.5*del
         do 10 i=1,n
   10    zsum=zsum+f(x+del*float(i))
         n=n*2
         r(k)=0.5*del*zsum
c
c  update Romberg extrapolation array
c
         w=4.
         do 15 j=2,k
         r(k+1-j)=(w*r(k+2-j)-r(k+1-j))/(w-1)
         w=4.*w
   15    continue
c
         zromld=r(1)
         if(k.lt.minp) go to 20
c
c  check integral estimate:
c
c    if change in estimate is within desired range, or if the
c    estimate falls below the absolute error desired, return
c
         if(cabs(zromld).lt.aerr) return
         if(cabs((zromld-rlast)/zromld).lt.rerr) return
   20    continue
         ier=3
c
c  error
c
   30    write(*,*) 'error in zromld -- ier = ',ier
         return
```

```
      end
```

```
------------------------------------------------------------
Output (OUTFIL) based on Romberg quadrature with minimum
requested accuracy of rerr=0.001:


 angle of incidence is   0.0000000     degrees

 current density in magnitude - angle format

 1  7.8412844E-04   149.6074
 2  8.8237022E-04  -143.5423
 3  2.1982782E-03  -68.95123
 4  3.9386787E-03  -12.91290
 5  5.6149564E-03   26.84530
 6  6.3408921E-03   41.16503
 7  5.6149517E-03   26.84531
 8  3.9386805E-03  -12.91293
 9  2.1982780E-03  -68.95126
10  8.8236929E-04  -143.5423

 rcs in dB-wavelength
      angle              rcs

   0.0000000       2.858046
    30.00000       2.047373
    60.00000       2.1480292E-02
    90.00000      -1.791418
    120.0000      -2.243502
    150.0000      -2.101166
    180.0000      -2.022754
    210.0000      -2.101165
    240.0000      -2.243500
    270.0000      -1.791417
    300.0000       2.1478748E-02
    330.0000       2.047372
------------------------------------------------------------
```

C.3 IMPLEMENTATION 3: GENERALIZED GAUSSIAN QUADRATURE

In this example, subroutine EZPJZ is replaced by a routine that employs a generalized Gaussian quadrature rule incorporating logarithmic singularities (Section A.4).

```
      complex function ezpjz(xs,ys,xo,yo,ws,omegs)
c
c     negative scattered electric field (Ez) at location
c     (xo,yo) due to a unit source (Jz) of width ws and
c     orientation omegs at (xs,ys)
c
c     computed using 15-point lin/log Gaussian quadrature,
c     so no need to separately evaluate the logarithmic
c     singularity
c
      complex QGLL15,zint1,zint2
      common xc,yc,com,som
      external zint1,zint2
c
      wso2=0.5*ws
      wsot=0.1*ws
      arg=sqrt((xo-xs)**2+(yo-ys)**2)
      if(arg.lt.wsot) then
c
c        assume observer at center of source cell
c
         ezpjz=QGLL15(zint1,0.,wso2)
         ezpjz=ezpjz*(2.,0.)
c
         else
c
c        observer not on source cell
c
         xc=xo-xs
         yc=yo-ys
         com=cos(omegs)
         som=sin(omegs)
         ezpjz=QGLL15(zint2,-wso2,wso2)
         endif
      ezpjz=(591.7661,0.)*ezpjz
      return
      end
c
c     ------------------------------------------------------
c
      complex function zint1(u)
c
c     the complex-valued integrand for the electric field
c     (Ez) produced by a unit source (Jz) when the observer
c     IS located on the source cell
c
      complex h02
c
```

```
      arg=6.283185308*u
      zint1=h02(arg)
      return
      end
c
c     ------------------------------------------------------
c
      complex function zint2(u)
c
c     the complex-valued integrand for the electric field
c     (Ez) produced by a unit source (Jz) when the observer
c     is NOT located on the source cell
c
      complex h02
      common xc,yc,com,som
c
      delx=xc-u*com
      dely=yc-u*som
      arg=6.283185308*sqrt((delx)**2+(dely)**2)
      zint2=h02(arg)
      return
      end
C
C-----------------------------------------------------------------------
C
        complex function QGLL15(zfun,a,b)
C
C       This function computes the integral of the complex function
C       'zfun' from a to b with 15-point Gaussian quadrature using
C       a mixture of polynomials and natural logarithms as basis
C       functions.
C       reference: Ma, Rokhlin, and Wandzura, "Generalized Gaussian
C       Quadrature Rules for Systems of Arbitrary Functions,"
C       Research Report, Yale University.
C
C       A. W. Mathis and A. F. Peterson, Feb 17, 1997
C
        complex asum,zfun
        real x(15), w(15)
        integer n
        external zfun
C
        data x/.105784548458629e-3,.156624383616782e-2,
     +          .759521890320709e-2,.228310673939862e-1,
     +          .523886301568200e-1,.100758685201213e0,
     +          .170740768849943e0,.262591206118993e0,
     +          .373536505184558e0,.497746358414533e0,
     +          .626789031392373e0,.750516103461408e0,
```

```
     +          .858255335207861e0,.940141291212346e0,
     +          .988401595986342e0/
        data w/.403217724648460e-3,.3062978434787e-2,
     +          .978421211876615e-2,.215587522255813e-1,
     +          .383230673708892e-1,.588981990263004e-1,
     +          .811170299392595e-1,.102122101972069e0,
     +          .118789059030401e0,.128210316446694e0,
     +          .128163327417093e0,.117489465888492e0,
     +          .963230185695904e-1,.661345398318934e-1,
     +          .296207140035355e-1/
C
        xlength = b-a
        asum = (0.,0.)
C
        do 10 n = 1,15
            asum = w(n)*zfun(xlength*x(n)+a)+asum
   10   continue
        QGLL15 = xlength*asum
        return
        end
```

Output (OUTFIL) based on 15-point lin/log Gaussian quadrature:

```
 angle of incidence is   0.0000000     degrees

 current density in magnitude - angle format

 1  7.8408106E-04   149.6063
 2  8.8233891E-04  -143.5416
 3  2.1982689E-03  -68.95104
 4  3.9386880E-03  -12.91325
 5  5.6149126E-03   26.84476
 6  6.3407957E-03   41.16441
 7  5.6149075E-03   26.84473
 8  3.9386861E-03  -12.91325
 9  2.1982710E-03  -68.95101
10  8.8234048E-04  -143.5417

 rcs in dB-wavelength
      angle              rcs

   0.0000000        2.858009
    30.00000        2.047342
    60.00000        2.1467930E-02
```

```
90.00000       -1.791422
120.0000       -2.243534
150.0000       -2.101223
180.0000       -2.022818
210.0000       -2.101220
240.0000       -2.243532
270.0000       -1.791422
300.0000       -2.1467414E-02
330.0000        2.047340
```