Appendix E
EIT Reconstruction Code `FEMEIT.F`

The reconstruction code `FEMEIT`, written in Fortran 77 by J. R. Torczynski, generates and solves the finite-element representation of the voltage equations (Eq. 3.1) for two-dimensional arbitrary domains, including multiply connected domains and geometries with electrodes on the domain boundary or within the domain itself. Electrodes themselves are represented by mesh nodes, essentially mathematical points. `FEMEIT` uses global conductivity functions selected from a subroutine library and applies a Newton-Raphson algorithm to find the conductivity parameters that most closely reproduce the measured voltages. Conductivity functions in the library include a constant conductivity, a single insulating bubble, a series of radial insulating annuli centered at an arbitrary position, and conductivity distributions described by Cartesian and radial polynomials.

`FEMEIT` reads from and writes to the following files:

`femeit_inp.dat`	general input parameters (input)
`femeit_exp.dat`	file of experimental voltages (input)
`femeit_nod.dat`	finite element mesh information (input)
`femeit_elc.dat`	table associating electrodes with mesh nodes (input)
`femeit_log.dat`	convergence and conductivity parameters after each iteration (diagnostic output)
`femeit_con.dat`	conductivity values at mesh nodes (output)
`femeit_out.dat`	general output parameters (output)

Examples of some file formats follow.

`femeit_inp.dat`:

`1.`	Length scale
`0.001`	Initial value of liquid conductivity, σ_L
`50`	Maximum number of iterations
`0.8`	}
`0.1`	} Dampening coefficients in Newton-Raphson algorithm
`1.`	}
`0.00001`	Convergence criterion for conductivity parameters
`0.00001`	Convergence criterion for residuals of voltage equation
`4`	}
`4`	} Parameters to select conductivity function
`2`	}

0.3 } C_1

0.3 } C_2

0. } C_3 Initial values of conductivity parameters

0. } C_4

0.99 } K_1

0.03 } K_2

`femeit_out.dat` echoes the input values in `femeit_inp.dat`, then reports the converged conductivity parameters.

`femeit_exp.dat:`

The input file `femeit_exp.dat` is a primary output file from the EIT data acquisition codes, listed in Appendices B and C. Columns 1 through 3 identify the injection, ground and measurement electrodes, respectively; columns 4 through 6 list the magnitude, carrier component and quadrature component of the measured voltages.

```
 1  2  1  858.63  847.32  138.91
 1  2  2    0.18    0.18    0.02
 1  2  3  359.77  343.14  108.11
 .  .  .     .       .       .
 .  .  .     .       .       .
 .  .  .     .       .       .
```

`femeit_nod.dat:`

The first line contains the number of nodes in the finite-element representation of the domain. Subsequent lines are composed of three columns containing, in order, the number of each node, its x coordinate and its y coordinate. The xy coordinate system is defined so that a circular domain is centered on the origin and has a radius of 1 unit. After the node list, the number of elements in the mesh is given, then each element is listed along with the three nodes defining its boundaries (all elements are triangular).

```
441
  1   0.996917     7.84591E-02
  2   0.987688     0.156434
  3   0.972370     0.233445
  .       .            .
  .       .            .
  .       .            .
800
   1   80    1  152
   2    1    2   81
   3    2    3   82
   .    .    .    .
   .    .    .    .
   .    .    .    .
```

femeit_elc.dat:

The first line contains the number of electrodes; the remaining lines list each electrode and its node number in the finite element mesh.

```
16
   1   5
   2  10
   3  15
   .   .
   .   .
   .   .
  16  80
```

```
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      program femeit

c
      implicit double precision (a-h,o-z)
c
      parameter (nnodem=441, nelemm=800, ngparm=15, nvert=3)
      parameter (nconnm=9)
      parameter (nprobm=16)
      parameter (npgm=2*ngparm)
      dimension xnode(nnodem), ynode(nnodem)
      dimension xelem(nelemm), yelem(nelemm)
      dimension condnd(nnodem), dconnd(nnodem)
      dimension ndelem(nelemm,nvert)
      dimension xvert(nvert), yvert(nvert)
      dimension ivert1(nvert), ivert2(nvert)
      dimension a1(nvert), ax(nvert), ay(nvert)
      dimension gigjda(nelemm,nvert,nvert)
      dimension cdnode(nnodem,0:ngparm), cdelem(nelemm,0:ngparm)
      dimension cdfcn(0:ngparm)
      dimension cond(ngparm), dcon(ngparm)
      dimension pg(npgm)
      dimension ndprob(nprobm), vnrm(nprobm,nprobm,nprobm)
      dimension femmat(nnodem,nnodem), feminv(nnodem,nnodem)
      dimension fempar(nnodem,nconnm,0:ngparm)
      dimension nconn(nnodem), nd21(nnodem,nconnm)
      dimension soln(nprobm), sjac(nprobm,ngparm)
      dimension psoln(nprobm,nprobm), psjac(nprobm,nprobm,ngparm)
      dimension amat(ngparm,ngparm), bvec(ngparm)
      data ivert1 / 2, 3, 1/
      data ivert2 / 3, 1, 2/
c
      wtints = 1.D+00 / dfloat(nvert+1)
      wtintc = 1.D+00 - 1.D+00 / dfloat(nvert+1)
c
 1001 format (i4,3(1x,e7.2),1x,e11.6,5(1x,e7.2))
 1002 format (1x,e11.5)
 1003 format (1x,i4)
 1004 format (2(1x,e11.5),i4)
 1005 format (i4,6(1x,e11.5))
 1006 format (3x,e11.5)
c
```

```
c *** Read in the mesh, first dimension, then nodes, then elements.
c
      write (6,*) 'Reading file femeit_nod.dat...'
      open (unit=21, status='unknown', file='femeit_nod.dat')
      read (21,*,end=998,err=998) nnode
      nnode1 = nnode - 1
      if (nnode.gt.nnodem) then
         write (6,*) 'Max no. nodes exceeded: ', nnode, nnodem
         close (unit=21)
         go to 998
         end if
      do 010 in = 1, nnode, 1
         read (21,*,end=998,err=998) nmnode, xnode(in), ynode(in)
         if (nmnode.ne.in) then
            write (6,*) 'Nodes not numbered in correct order.'
            close (unit=21)
            go to 998
            end if
  010    continue
      read (21,*,end=998,err=998) nelem
      if (nelem.gt.nelemm) then
         write (6,*) 'Max no. elems exceeded: ', nelem, nelemm
         close (unit=21)
         go to 998
         end if
      do 020 ie = 1, nelem, 1
         read (21,*,end=998,err=998) nmelem, (ndelem(ie,kv),kv=1,nvert)
         if (nmelem.ne.ie) then
            write (6,*) 'Elements not numbered in correct order.'
            close (unit=21)
            go to 998
            end if
  020    continue
      close(unit=21)
c
c *** Read in conductivity fitting information.
c
      write (6,*) 'Reading file femeit_inp.dat...'
      open (unit=22, status='unknown', file='femeit_inp.dat')
      read (22,*,end=998,err=998) slen
      read (22,*,end=998,err=998) sigma0
      read (22,*,end=998,err=998) niter
      read (22,*,end=998,err=998) damp0
      read (22,*,end=998,err=998) damp1
      read (22,*,end=998,err=998) damp2
      read (22,*,end=998,err=998) tolc
      read (22,*,end=998,err=998) tolr
      read (22,*,end=998,err=998) kctype
      read (22,*,end=998,err=998) ngpar
      read (22,*,end=998,err=998) npg
      call cdchk(kctype,ngpar,npg,ngparx,npgx,ichk)
      if ((ngpar.ne.ngparx).or.(npg.ne.npgx).or.(ichk.ne.1)) then
         write (6,*) 'No. parameters not as expected: '
         write (6,*) 'If ichk = 0, model not available: ', kctype
         write (6,*) 'ngpar, ngparx ', ngpar, ngparx
         write (6,*) 'npg,   npgx   ', npg,   npgx
         close (unit=22)
         go to 998
         end if
      if (ngpar.gt.ngparm) then
         write (6,*) 'Max no. gpars exceeded: ', ngpar, ngparm
         close (unit=22)
         go to 998
```

```
         end if
      do 030 ig = 1, ngpar, 1
         read (22,*,end=998,err=998) cond(ig)
  030    continue
      if (npg.gt.npgm) then
         write (6,*) 'Max no. gfcn pars exceeded: ', npg, npgm
         close (unit=22)
         go to 998
         end if
      if ((npg.gt.0).and.(npg.le.npgm)) then
         read (22,*,end=998,err=998) (pg(ipg),ipg=1,npg,1)
         end if
      close (unit=22)
c
c *** Read in electrode numbers and nodes.
c
      write (6,*) 'Reading file femeit_elc.dat...'
      open (unit=23, status='unknown', file='femeit_elc.dat')
      read (23,*,end=998,err=998) nprob
      if (nprob.gt.nprobm) then
         write (6,*) 'Max no. electrodes exceeded: ', nprob, nprobm
         close (unit=23)
         go to 998
         end if
      nexpt = nprob * (nprob - 1) / 2
      do 040 ip = 1, nprob, 1
         read (23,*,end=998,err=998) iprob, ndprob(ip)
         if (iprob.ne.ip) then
            write (6,*) 'Electrodes out of order:', ip, iprob
            close (unit=23)
            go to 998
            end if
  040    continue
      close (unit=23)
c
c *** Read in experimental data and normalize voltage by current.
c
      write (6,*) 'Reading file femeit_exp.dat...'
      open (unit=24, status='unknown', file='femeit_exp.dat')
      svcex2 = 0.D+00
      do 070 ip1 = 1, nprob-1, 1
      do 070 ip2 = ip1+1, nprob, 1
         wtotal = 0.D+00
         svcexa = 0.D+00
         svcexb = 0.D+00
         do 060 ip = 1, nprob, 1
            curr12 = 1.D+00
            read (24,*) ipa, ipb, ipc, vmagn, vcarr, vquad
*           vcex = vmagn * sigma0 * slen / curr12
            wtex = wt(ip,ip1,ip2)
            wtotal = wtotal + wtex
            svcexa = svcexa + wtex * vcex
            svcexb = svcexb + wtex * vcex ** 2
            vnrm(ip,ip1,ip2) = vcex
  060       continue
         svcexa = svcexa / wtotal
         svcexb = svcexb / wtotal
         svcex2 = svcex2 + svcexb - svcexa ** 2
  070    continue
      svcex1 = sqrt(svcex2 / dfloat(nexpt))
      close (unit=24)
c
c *** Compute nontrivial node pairs.
```

```
c
      write (6,*) 'Computing nontrivial node pairs...'
      do 080 in1 = 1, nnode, 1
      do 080 in2 = 1, nnode, 1
         femmat(in1,in2) = 0.D+00
  080    continue
c
      do 110 ie = 1, nelem, 1
         do 100 iv1 = 1, nvert, 1
            in1 = ndelem(ie,iv1)
         do 090 iv2 = 1, nvert, 1
            in2 = ndelem(ie,iv2)
            femmat(in1,in2) = 1.D+00
  090       continue
  100       continue
  110    continue
c
      do 130 in1 = 1, nnode, 1
         nconn(in1) = 0
      do 120 in2 = 1, nnode, 1
         if (femmat(in1,in2).gt.0.5D+00) then
            nconn(in1) = nconn(in1) + 1
            nd21(in1,nconn(in1)) = in2
            if (nconn(in1).gt.nconnm) then
               write (6,*) '*** INSUFFICIENT LINKS ***'
               go to 998
               end if
            end if
  120    continue
  130    continue
c
c *** Compute element quantities: Int. grad phi_i . grad phi_j dV.
c
      write (6,*) 'Computing element quantities...'
      inotcc = 0
      do 170 ie = 1, nelem, 1
         xsum = 0.D+00
         ysum = 0.D+00
         do 140 iv = 1, nvert, 1
            xvert(iv) = xnode(ndelem(ie,iv))
            yvert(iv) = ynode(ndelem(ie,iv))
            xsum = xsum + xvert(iv)
            ysum = ysum + yvert(iv)
  140       continue
         xelem(ie) = xsum / dfloat(nvert)
         yelem(ie) = ysum / dfloat(nvert)
         size = 0.D+00
         do 150 iv = 1, nvert, 1
            a1(iv) = xvert(ivert1(iv)) * yvert(ivert2(iv))
     1             - xvert(ivert2(iv)) * yvert(ivert1(iv))
            ax(iv) = yvert(ivert1(iv)) - yvert(ivert2(iv))
            ay(iv) = xvert(ivert2(iv)) - xvert(ivert1(iv))
            size = size + 0.5D+00 * a1(iv)
  150       continue
         if (size.le.0) then
            write (6,*) 'Nodes not counterclockwise for elt: '
            write (6,*) ie, size
            inotcc = 1
            if (size.eq.0) go to 998
            end if
         do 160 iv1 = 1, nvert, 1
         do 160 iv2 = 1, nvert, 1
            gigj = ax(iv1) * ax(iv2) + ay(iv1) * ay(iv2)
```

```
            gigjda(ie,iv1,iv2) = gigj * 0.25D+00 / size
  160       continue
  170    continue
      if (inotcc.gt.0) go to 998
c
c *** Begin major iteration loop.
c
      write (6,*) 'Beginning major iteration loop...', niter
      write (6,*) 'Writing file femeit_log.dat...'
      write (6,*) ' '
      write (6,*)  ' it  tolc    snorm   sdcon   scond       '//
     1             'tolr    snvec   sbvec   rmsnrm  corr'
      open (unit=25, status='unknown', file='femeit_log.dat')
      write (25,*) ' it  tolc    snorm   sdcon   scond       '//
     1             'tolr    snvec   sbvec   rmsnrm  corr'
      do 500 it = 1, niter, 1
c
c *** Compute average conductivity and derivatives for each element.
c
      do 190 in = 1, nnode, 1
         xn = xnode(in)
         yn = ynode(in)
         call cdsub(xn,yn,slen,kctype,npg,pg,ngpar,cond,cdfcn)
         do 180 ig = 0, ngpar, 1
            cdnode(in,ig) = cdfcn(ig)
  180       continue
  190       continue
c
      do 198 ie = 1, nelem, 1
         xe = xelem(ie)
         ye = yelem(ie)
         call cdsub(xe,ye,slen,kctype,npg,pg,ngpar,cond,cdfcn)
         do 196 ig = 0, ngpar, 1
            cdelem(ie,ig) = cdfcn(ig)
  196       continue
  198    continue
c
      do 220 ie = 1, nelem, 1
      do 210 ig = 0, ngpar, 1
         cdfcn(ig) = 0.D+00
         do 200 iv = 1, nvert, 1
            cdfcn(ig) = cdfcn(ig) + cdnode(ndelem(ie,iv),ig)
  200       continue
         cdfcn(ig) = cdfcn(ig) / dfloat(nvert)
         cdelem(ie,ig) = wtintc * cdelem(ie,ig) + wtints * cdfcn(ig)
  210    continue
  220    continue
c
c *** Assemble the derivative FEM matrices.
c
      do 230 in1 = 1, nnode, 1
      do 230 ic = 1, nconnm, 1
      do 230 ig = 0, ngpar, 1
         fempar(in1,ic,ig) = 0.D+00
  230    continue
c
      do 250 ie = 1, nelem, 1
      do 250 iv1 = 1, nvert, 1
         in1 = ndelem(ie,iv1)
         nconn1 = nconn(in1)
      do 250 iv2 = 1, nvert, 1
         in2 = ndelem(ie,iv2)
         ic = 0
```

```
         do 240 ic0 = 1, nconn1, 1
            if (nd21(in1,ic0).eq.in2) ic = ic0
  240       continue
        do 250 ig = 0, ngpar, 1
           fempar(in1,ic,ig) = fempar(in1,ic,ig)
     1        + cdelem(ie,ig) * gigjda(ie,iv1,iv2)
  250    continue
c
c *** Assemble FEM matrix.
c
        do 260 in1 = 1, nnode, 1
        do 260 in2 = 1, nnode, 1
           femmat(in1,in2) = 0.D+00
           feminv(in1,in2) = 0.D+00
  260    continue
c
        do 280 in1 = 1, nnode, 1
           nconn1 = nconn(in1)
        do 280 ic = 1, nconn1, 1
           in2 = nd21(in1,ic)
           femmat(in1,in2) = fempar(in1,ic,0)
           feminv(in1,in2) = femmat(in1,in2)
  280    continue
c
c *** Invert FEM matrix.
c
        call zmtinv(femmat,feminv,nnode1,rcond,info)
c
c *** Loop over electrode-electrode combinations to calculate
c *** partial solution vector and portion of electrode Jacobian.
c     Could speed up using psoln(ipk,ipm) = psoln(ipm,ipk)
c     and psjac(ipk,ipm,ig) = psjac(ipm,ipk,ig).
c
        do 380 ipm = 1, nprob, 1
           inm = ndprob(ipm)
        do 380 ipk = 1, nprob, 1
           ink = ndprob(ipk)
c
           psoln(ipm,ipk) = feminv(ink,inm)
c
        do 360 ig = 1, ngpar, 1
           psj = 0.D+00
        do 340 in1 = 1, nnode1, 1
           nconn1 = nconn(in1)
        do 340 ic = 1, nconn1, 1
           in2 = nd21(in1,ic)
           psj = psj+feminv(ink,in1)*fempar(in1,ic,ig)*feminv(in2,inm)
  340    continue
           psjac(ipm,ipk,ig) = - psj
  360    continue
c
  380    continue
c
c ** Calculate the least-squares matrix and vector.
c
        do 390 ig1 = 1, ngpar, 1
           bvec(ig1) = 0.D+00
        do 390 ig2 = 1, ngpar, 1
           amat(ig1,ig2) = 0.D+00
  390    continue
c
c *** Calculate the least-squares matrix and vector.
c     Could speed up using amat(ig2,ig1) = amat(ig1,ig2).
```

```
c
      svrms2 = 0.D+00
      do 460 ipm = 1, nprob-1, 1
      do 460 ipn = ipm+1, nprob, 1
      do 410 ipk = 1, nprob, 1
         soln(ipk) = psoln(ipm,ipk) - psoln(ipn,ipk)
      do 400 ig1 = 1, ngpar, 1
         sjac(ipk,ig1) = psjac(ipm,ipk,ig1) - psjac(ipn,ipk,ig1)
  400    continue
  410    continue
         wtotal = 0.D+00
         svrmsa = 0.D+00
         svrmsb = 0.D+00
      do 450 ipk = 1, nprob, 1
         bterm = vnrm(ipk,ipm,ipn) - soln(ipk)
         wtipk = wt(ipk,ipm,ipn)
         wtotal = wtotal + wtipk
         svrmsa = svrmsa + wtipk * bterm
         svrmsb = svrmsb + wtipk * bterm * bterm
      do 440 ipl = 1, nprob, 1
         wtipl = wt(ipl,ipm,ipn)
         wtterm = wtipk * wtipl
      do 430 ig1 = 1, ngpar, 1
         sjterm = wtterm * ( sjac(ipk,ig1) - sjac(ipl,ig1) )
         bvec(ig1) = bvec(ig1) + sjterm * bterm
      do 420 ig2 = 1, ngpar, 1
         amat(ig1,ig2) = amat(ig1,ig2) + sjterm * sjac(ipk,ig2)
  420    continue
  430    continue
  440    continue
  450    continue
         svrmsa = svrmsa / wtotal
         svrmsb = svrmsb / wtotal
         svrms2 = svrms2 + svrmsb - svrmsa ** 2
  460    continue
      svrms1 = sqrt(svrms2 / dfloat(nexpt))
      rmsnrm = svrms1 / svcex1
c
c *** Solve matrix equation for conductivity parameter increments.
c
      call zlnsol(amat,ngpar,bvec,dcon,indwrn)
c
c *** Determine nodal conductivity, change thereof, damping parameter.
c     Update conductivity. Compute changes. Test tolerances.
c
      damp = damp0
      do 480 in = 1, nnode, 1
         condnd(in) = cdnode(in,0)
         dconnd(in) = 0.D+00
         do 470 ig = 1, ngpar, 1
            dconnd(in) = dconnd(in) + cdnode(in,ig) * dcon(ig)
  470       continue
         damp = max(damp,abs(dconnd(in)/condnd(in)))
  480    continue
      corr = max(damp1,min(damp2,damp0/damp))
c
      scond = 0.D+00
      sdcon = 0.D+00
      sbvec = 0.D+00
      do 490 ig = 1, ngpar, 1
         cond(ig) = cond(ig) + corr * dcon(ig)
         scond = scond + cond(ig)*cond(ig)
         sdcon = sdcon + dcon(ig)*dcon(ig)
```

```
         sbvec = sbvec + bvec(ig)*bvec(ig)
  490    continue
       scond = sqrt( scond / dfloat(ngpar) )
       sdcon = sqrt( sdcon / dfloat(ngpar) )
       snorm = sdcon / scond
       sbvec = sqrt( sbvec / dfloat(ngpar) )
       if (it.eq.1) svscal = sbvec
       snvec = sbvec / svscal
       write (25,1001) it, tolc, snorm, sdcon, scond,
      1                tolr, snvec, sbvec, rmsnrm, corr
       write (6,1001)  it, tolc, snorm, sdcon, scond,
      1                tolr, snvec, sbvec, rmsnrm, corr
       if ((snorm.lt.tolc).and.(snvec.lt.tolr)) go to 800
c
c *** End major iteration loop.
c
  500 continue
      write (6,*) 'Exceeded iteration limit without convergence.'
c
c *** Write solution and close files.
c
  800 continue
       close (unit=25)
       write (6,*) ' '
c
       write (6,*) 'Writing file femeit_out.dat...'
       open (unit=26, status='unknown', file='femeit_out.dat')
       write (26,1002) slen
       write (26,1002) sigma0
       write (26,1003) niter
       write (26,1002) damp0
       write (26,1002) damp1
       write (26,1002) damp2
       write (26,1002) tolc
       write (26,1002) tolr
       write (26,1003) kctype
       write (26,1003) ngpar
       write (26,1003) npg
       do 810 ig = 1, ngpar, 1
          write (26,1004) cond(ig), dcon(ig), ig
  810     continue
       if (npg.gt.0) then
          do 820 ipg = 1, npg, 1
             write (26,1006) pg(ipg)
  820        continue
          end if
       close (unit=26)
c
       write (6,*) 'Writing file femeit_con.dat...'
       do 840 in = 1, nnode, 1
          xn = xnode(in)
          yn = ynode(in)
          call cdsub(xn,yn,slen,kctype,npg,pg,ngpar,cond,cdfcn)
          condnd(in) = cdfcn(0)
          dconnd(in) = 0.D+00
          do 830 ig = 1, ngpar, 1
             dconnd(in) = dconnd(in) + cdnode(in,ig) * dcon(ig)
  830        continue
          damp = max(damp,abs(dconnd(in)/condnd(in)))
  840     continue
       open (unit=27, status='unknown', file='femeit_con.dat')
       do 850 in = 1, nnode, 1
          write (27,1005) in, condnd(in), dconnd(in)
```

```
  850    continue
      close (unit=27)
c
c *** Completed, stop.
c
      go to 999
  998 write (6,*) '*** ABNORMAL STOP ***'
  999 stop 'femeit'
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      subroutine cdchk(k,ng,np,ngx,npx,ichk)
c
      implicit double precision (a-h,o-z)
c
c *** Set default to abort.
c
      ngx = -1
      npx = -1
      ichk = 0
c
c *** Cartesian polynomials.
c
      if (k.eq.0) then
         ngx = max(0,ng)
         npx = 2 * ngx
         ichk = 1
         end if
c
c *** Radial polynomials and angular sine and cosine.
c
      if (k.eq.1) then
         ngx = max(0,ng)
         npx = 2 * ngx
         ichk = 1
         end if
c
c *** Piecewise linear interpolation over ng radial annuli.
c
      if (k.eq.2) then
         ngx = max(1,ng)
         npx = ngx
         ichk = 1
         end if
c
c *** Bubble centered at origin.
c
      if (k.eq.3) then
         ngx = 2
         npx = 2
         ichk = 1
         end if
c
c *** Bubble centered arbitrarily.
c
      if (k.eq.4) then
         ngx = 4
         npx = 2
         ichk = 1
         end if
c
c *** Bubble centered arbitrarily with known conductivity.
```

```
c
      if (k.eq.5) then
         ngx = 3
         npx = 3
         ichk = 1
         end if
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      subroutine cdsub(x,y,sl,k,np,p,ng,cn,cdfcn)
c
c *** Determines conductivity and derivatives at (x,y).
c     Function is type k, with np internal parameters in p,
c     ng fitting parameters in cn, and length scale sl.
c     cdfcn(0) = conductivity
c     cdfcn(ig) = d conductivity / d cn(ig)
c
      implicit double precision (a-h,o-z)
      dimension p(np)
      dimension cn(ng)
      dimension cdfcn(0:ng)
c
      zero = 0.D+00
      unit = 1.D+00
      two2 = 2.D+00
      pi = 2.D+00 * asin(1.D+00)
c
      radius = sqrt(x**2+y**2)
      angle = 0.D+00
      if ((x.ne.zero).or.(y.ne.zero)) angle = atan2(y,x)
      xsl = x / sl
      ysl = y / sl
      rsl = radius / sl
      do 010 ig = 0, ng, 1
         cdfcn(ig) = 0.D+00
  010    continue
c
c *** Cartesian polynomials.
c     p(2*ig-1) = m = x exponent, p(2*ig) = n = y exponent.
c     cn(ig) is coefficient multiplying xsl**m * ysl**n.
c
      if (k.eq.0) then
         cdfcn(0) = 0.D+00
         do 100 ig = 1, ng, 1
            m = nint(p(2*ig-1))
            xfcn = 1.D+00
            if (m.ne.0) xfcn = xsl ** m
            n = nint(p(2*ig))
            yfcn = 1.D+00
            if (n.ne.0) yfcn = ysl ** n
            cdfcn(ig) = xfcn * yfcn
            cdfcn(0) = cdfcn(0) + cn(ig) * cdfcn(ig)
  100       continue
         end if
c
c *** Radial polynomials and angular sine and cosine.
c     p(2*ig-1) = m = r exponent, p(2*ig) = n = theta harmonic.
c     If n < 0, cosine. If n > 0, sine. If n = 0, unity.
c     cn(ig) is coefficient multiplying rsl**m * trig(n*theta).
c
```

```
      if (k.eq.1) then
         cdfcn(0) = 0.D+00
         do 110 ig = 1, ng, 1
            m = nint(p(2*ig-1))
            rfcn = 1.D+00
            if (m.ne.0) rfcn = rsl ** m
            n = nint(p(2*ig))
            zn = dfloat(abs(n))
            tfcn = 1.D+00
            if (n.lt.0) tfcn = cos(zn*angle)
            if (n.gt.0) tfcn = sin(zn*angle)
            cdfcn(ig) = rfcn * tfcn
            cdfcn(0) = cdfcn(0) + cn(ig) * cdfcn(ig)
  110       continue
         end if
c
c *** Piecewise linear interpolation over ng radial annuli.
c     p(ig) are the right-hand radii of the annuli, ng values.
c     cn(ig) are the values at these radii.
c     If they existed, p(0) = 0 (i.e. r=0), and cn(0) = cn(1)
c     (i.e. the value at r = 0 is taken to be cn(1)).
c
      if (k.eq.2) then
         iannu = 0
         do 140 ig = ng, 1, -1
            cdfcn(ig) = 0.D+00
            rrght = p(ig) / sl
            if (rsl.le.rrght) iannu = ig
  140       continue
         if (iannu.eq.0) iannu = ng
         if (iannu.eq.1) then
            frght = 1.D+00
            cdfcn(0) = frght * cn(1)
            cdfcn(1) = frght
            end if
         if (iannu.ne.1) then
            ig = iannu
            rleft = p(ig-1) / sl
            rrght = p(ig) / sl
            frght = ( rsl - rleft ) / ( rrght - rleft )
            frght = max(zero,min(unit,frght))
            fleft = 1.D+00 - frght
            cdfcn(0) = fleft * cn(ig-1) + frght * cn(ig)
            cdfcn(ig-1) = fleft
            cdfcn(ig) = frght
            end if
         end if
c
c *** Bubble model: nearly zero inside, nearly constant outside.
c     Smooth transition between 2 constant regions: 1 circular
c     region embedded eccentrically in another circular region.
c     cn(1) is overall amplitude.
c     cn(2) is normalized radius for changeover.
c     Center: (0,0) for k = 3, (cn(3),cn(4)) for k = 4.
c     p(1) is degree of variation, strictly 0 < p(1) < 1.
c     p(2) is normalized width of changeover, strictly p(2) > 0.
c
      if (k.eq.3) then
         arg1 = ( rsl - cn(2) ) / p(2)
         arg2 = ( rsl + cn(2) ) / p(2)
         cdfcn(1) = 1.D+00 + 0.5D+00 * p(1) * ( tanh(arg1)-tanh(arg2) )
         cdfcn(0) = cn(1) * cdfcn(1)
         sech12 = 1.D+00 / cosh(arg1)**2
```

```
         sech22 = 1.D+00 / cosh(arg2)**2
         cdfcn0 = cn(1) * 0.5D+00 * p(1) / p(2)
         cdfcn(2) = - cdfcn0 * ( sech12 + sech22 )
         end if
c
      if (k.eq.4) then
         rlr = sqrt((xsl-cn(3))**2+(ysl-cn(4))**2)
         arg1 = ( rlr - cn(2) ) / p(2)
         arg2 = ( rlr + cn(2) ) / p(2)
         cdfcn(1) = 1.D+00 + 0.5D+00 * p(1) * ( tanh(arg1)-tanh(arg2) )
         cdfcn(0) = cn(1) * cdfcn(1)
         sech12 = 1.D+00 / cosh(arg1)**2
         sech22 = 1.D+00 / cosh(arg2)**2
         cdfcn0 = cn(1) * 0.5D+00 * p(1) / p(2)
         cdfcn(2) = - cdfcn0 * ( sech12 + sech22 )
         cdfcn(3) = 0.D+00
         cdfcn(4) = 0.D+00
         if (rlr.ne.0.D+00) then
            cdfcn5 = cdfcn0 * ( sech22 - sech12 )
            cdfcn(3) = ( xsl - cn(3) ) / rlr * cdfcn5
            cdfcn(4) = ( ysl - cn(4) ) / rlr * cdfcn5
            end if
         end if
c
      if (k.eq.5) then
         rlr = sqrt((xsl-cn(2))**2+(ysl-cn(3))**2)
         arg1 = ( rlr - cn(1) ) / p(2)
         arg2 = ( rlr + cn(1) ) / p(2)
         cdfcn(0) = p(3) * (1.D+00+0.5D+00*p(1)*(tanh(arg1)-tanh(arg2)))
         sech12 = 1.D+00 / cosh(arg1)**2
         sech22 = 1.D+00 / cosh(arg2)**2
         cdfcn0 = p(3) * 0.5D+00 * p(1) / p(2)
         cdfcn(1) = - cdfcn0 * ( sech12 + sech22 )
         cdfcn(2) = 0.D+00
         cdfcn(3) = 0.D+00
         if (rlr.ne.0.D+00) then
            cdfcn5 = cdfcn0 * ( sech22 - sech12 )
            cdfcn(2) = ( xsl - cn(2) ) / rlr * cdfcn5
            cdfcn(3) = ( ysl - cn(3) ) / rlr * cdfcn5
            end if
         end if
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      function wt(i,i1,i2)
c
      implicit double precision (a-h,o-z)
      wt = 1.D+00
      if ((i.eq.i1).or.(i.eq.i2)) wt = 0.D+00
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      subroutine zmtinv(amat,ainv,n,rcond,info)
c
c *** Drives SLATEC routines DPOCO, DPODI.
c
      implicit double precision (a-h,o-z)
```

```
      parameter (nnodem=441)
      parameter (lda=nnodem)
      dimension amat(lda,lda), ainv(lda,lda), work(lda)
      dimension det(2)
c
      do 100 i = 1, lda, 1
      do 100 j = 1, lda, 1
         ainv(i,j) = amat(i,j)
  100    continue
c
      call dpoco(ainv,lda,n,rcond,work,info)
c
      call dpodi(ainv,lda,n,det,11)
c
      do 200 i = 1, lda, 1
      do 200 j = 1, i, 1
         ainv(i,j) = ainv(j,i)
  200    continue
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      subroutine zlnsol(amat,n,bvec,xvec,ind)
c
c *** Drives SLATEC routine DGEFS.
c
      implicit double precision (a-h,o-z)
      parameter (ngparm=15)
      parameter (lda=ngparm)
      dimension amat(lda,lda), bvec(lda), xvec(lda)
      dimension awrk(lda,lda), work(lda), iwork(lda)
c
      do 100 i = 1, lda, 1
         xvec(i) = bvec(i)
      do 100 j = 1, lda, 1
         awrk(i,j) = amat(i,j)
  100    continue
c
      call dgefs(awrk,lda,n,xvec,1,ind,work,iwork)
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
```