Appendix I
GDT Reconstruction Code GDTAXI.F

This Fortran 77 program by J. R. Torczynski determines the radial conductivity distribution in a two-dimensional cylindrical domain from gamma-ray attenuation measurements along many parallel beam paths through the domain. A complete description of the algorithm may be found in Torczynski *et al.* (1997).

GDTAXI reads from and writes to the following files:

gdtaxi_inp.dat	general input parameters (input)
gdtaxi_ful.dat	photon count rate data from full column with no gas present (input)
gdtaxi_emp.dat	photon count rate data from empty column with no liquid present (input)
gdtaxi_flo.dat	photon count rate data from column during flow of interest (input)
gdtaxi_out.dat	general output parameters and coefficients of volume fraction profiles (output)
gdtaxi_gas.dat	gas volume fraction profile (output)
gdtaxi_liq.dat	liquid volume fraction profile (output)

Examples of some input file formats follow.

gdtaxi_inp.dat:

9.525	column inner radius (cm)
19.79	column x-midpoint (cm)
0.1	thickness of boundary layer within which data is discarded (cm)
0	clipping suppressed or enabled (0 or 1, respectively)
0.	lower value for clipped gas volume fraction (usually 0)
1.	upper value for clipped gas volume fraction (usually 1)
0.0E-06	time constant for nonlinear detector response (s), usually set to 0
4	degree of polynomial fit (even integer: 0 = constant, 2 = quadratic, 4 = quartic, etc.)

gdtaxi_out.dat echoes the contents of gdtaxi_inp.dat, then reports the column-averaged gas and liquid volume fractions, and finally prints the polynomial coefficients of the computed volume fraction distributions.

gdtaxi_ful.dat, gdtaxi_emp.dat, gdtaxi_flo.dat:

These are primary output files from the LabView program that controls the GDT system. The first line of each file describes the horizontal and vertical motion of the source and detector during the scan; GDTAXI compares this information in all three files to verify that they are measurements of the same column geometry. The second line, which is not used by GDTAXI, reports parameters used by LabView to compute count rates from the output of the multichannel analyzer. The remaining lines of each input file contain gamma beam positions and the count rates at each position. Columns 1 and 2 list the horizontal and vertical coordinates, respectively, of the source and detector; column 3 contains the count rate at the detector at those coordinates. Columns 4 through 6 are not used.

```
10.780 1.000  18.000 26.650 0.000  0.000  0.000  0.000
60     280    480    4.000  5.000  0.100  0.100
10.78  26.65  8208.94       148.49 55.28  104.45
11.78  26.65  6295.64       114.12 55.17  104.74
12.78  26.65  5314.41        96.14 55.28  104.82
  .      .       .            .     .       .
  .      .       .            .     .       .
  .      .       .            .     .       .
28.78  26.65  8213.49       148.87 55.17  105.37
```

```
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      program gdtaxi
c
c     Revision 19990420
c
c *** Gamma-densitometry tomography axisymmetric reconstruction
c     using even radial polynomials.
c
      implicit double precision (a-h,o-z)
c
      parameter (nsm=1000)
      parameter (nfcnm=10)
      dimension spos(nsm), full(nsm), empt(nsm), flow(nsm)
      dimension snrm(nsm), void(nsm)
      dimension nexp(nfcnm), amat(nfcnm,nfcnm), bvec(nfcnm)
      dimension cmat(nfcnm,nfcnm)
      dimension ravf(nsm), axvf(nsm)
      dimension cravf(nfcnm), caxvf(nfcnm)
      dimension cramf(nfcnm), caxmf(nfcnm)
      dimension ccoeff(nfcnm,nfcnm), dcoeff(nfcnm,nfcnm)
c
 1001 format (1x,d18.12)
 1002 format (1x,i4)
 1003 format (1x,i4,5(1x,d11.5))
 1004 format (6(1x,d11.5))
 2000 format (1x,a)
 2001 format (1x,a12,d18.12)
 2002 format (1x,a12,i4)
c
c *** Read in geometric and fitting information.
c
```

```
      write (6,2000) 'Reading input parameters from gdtaxi_inp.dat'
      open (unit=24, status='old', file='gdtaxi_inp.dat')
      read (24,*) rinner
      read (24,*) scentr
      read (24,*) dxedge
      read (24,*) iclip
      read (24,*) cliplo
      read (24,*) cliphi
      read (24,*) tau
      read (24,*) nexpm
      write (6,2001) '   rinner = ', rinner
      write (6,2001) '   scentr = ', scentr
      write (6,2001) '   dxedge = ', dxedge
      write (6,2002) '   iclip  = ', iclip
      write (6,2001) '   cliplo = ', cliplo
      write (6,2001) '   cliphi = ', cliphi
      write (6,2001) '   tau    = ', tau
      write (6,2002) '   nexpm  = ', nexpm
      close (unit=24)
c
c *** Read in full, empty, flow scans.
c *** Do some error checking for consistent files.
c
      write (6,2000) 'Reading experimental data from '
      write (6,2000) '   gdtaxi_ful.dat gdtaxi_emp.dat gdtaxi_flo.dat'
      open (unit=21, status='old', file='gdtaxi_ful.dat')
      open (unit=22, status='old', file='gdtaxi_emp.dat')
      open (unit=23, status='old', file='gdtaxi_flo.dat')
      read (21,*) x1f, dxf, xnf, y1f, dyf, ynf
      read (22,*) x1e, dxe, xne, y1e, dye, yne
      read (23,*) x1b, dxb, xnb, y1b, dyb, ynb
      nxf = nint(xnf)
      nyf = nint(ynf)
      nxe = nint(xne)
      nye = nint(yne)
      nxb = nint(xnb)
      nyb = nint(ynb)
      if ((nyf.ne.0).or.(nye.ne.0).or.(nyb.ne.0)) then
         close (unit=21)
         close (unit=22)
         close (unit=23)
         write (6,*) '*** SCAN IS IN Y-DIRECTION ***'
         write (6,*) 'ful ', nyf, ' emp ', nye, ' flo ', nyb
         go to 998
         end if
      if ((nxf.ne.nxe).or.(nxf.ne.nxb)) then
         close (unit=21)
         close (unit=22)
         close (unit=23)
         write (6,*) '*** SCANS HAVE DIFFERENT NUMBERS OF POINTS ***'
         write (6,*) 'ful ', nxf, ' emp ', nxe, ' flo ', nxb
         go to 998
         end if
      tol = 0.05 * rinner
      if ((abs(x1f-x1e).gt.tol).or.(abs(x1f-x1b).gt.tol)) then
c         close (unit=21)
c         close (unit=22)
c         close (unit=23)
         write (6,*) '*** SCANS HAVE DIFFERENT ORIGINS ***'
         write (6,*) 'ful ', x1f, ' emp ', x1e, ' flo ', x1b
c         go to 998
         end if
      if ((abs(dxf-dxe).gt.tol).or.(abs(dxf-dxb).gt.tol)) then
```

```
         close (unit=21)
         close (unit=22)
         close (unit=23)
         write (6,*) '*** SCANS HAVE DIFFERENT STEP SIZES ***'
         write (6,*) 'ful ', dxf, ' emp ', dxe, ' flo ', dxb
         go to 998
         end if
      if (nxf+1.gt.nsm) then
         close (unit=21)
         close (unit=22)
         close (unit=23)
         write (6,*) '*** NOT ENOUGH POINTS AVAILABLE ***'
         write (6,*) nxf, nsm
         go to 998
         end if
c
c *** Skip header information
c
      read (21,*) tmlive
      read (22,*) tmlive
      read (23,*) tmlive
c
c *** Read in data and compute the nominal rates.
c
      ns = nxf + 1
      do 100 is = 1, ns, 1
         read (21,*,end=998,err=998) sposx, sposy, frate
         read (22,*,end=998,err=998) sposx, sposy, erate
         read (23,*,end=998,err=998) sposx, sposy, brate
         spos(is) = sposx
         full(is) = frate
         empt(is) = erate
         flow(is) = brate
  100    continue
c
c *** Close files.
c
      close (unit=21)
      close (unit=22)
      close (unit=23)
c
c *** Compute best fit to ray averages.
c
      write (6,2000) 'Computing results'
      nfcn = nexpm / 2 + 1
      if (nfcn.gt.nfcnm) then
         close (unit=24)
         write (6,2000) '*** NOT ENOUGH FITTING FUNCTIONS AVAILABLE ***'
         write (6,2000) nfcn, nfcnm
         go to 998
         end if
      do 120 ifcn = 1, nfcn, 1
         nexp(ifcn) = 2 * ( ifcn - 1 )
  120    continue
c
c *** Correct the rates for nonlinear detector response.
c
      do 140 is = 1, ns, 1
         full(is) = full(is) / ( 1. - full(is) * tau )
         empt(is) = empt(is) / ( 1. - empt(is) * tau )
         flow(is) = flow(is) / ( 1. - flow(is) * tau )
  140    continue
c
```

```
c *** Take logarithms to get absorptions.
c
      do 150 is = 1, ns, 1
         full(is) = - log( full(is) )
         empt(is) = - log( empt(is) )
         flow(is) = - log( flow(is) )
  150    continue
c
c *** Compute left and right array bounds for inner diameter.
c
      s1 = scentr - rinner + dxedge
      s2 = scentr + rinner - dxedge
      ns1 = 1
      ns2 = 0
      do 160 is = 1, ns, 1
         if (spos(is).le.s1) ns1 = ns1 + 1
         if (spos(is).lt.s2) ns2 = ns2 + 1
  160    continue
c
c *** Compute the normalized position and the average void fraction on rays.
c
      do 180 is = 1, ns, 1
         snrm(is) = ( spos(is) - scentr ) / rinner
         void(is) = 0.
         if ((is.ge.ns1).and.(is.le.ns2)) then
            void(is) = ( full(is) - flow(is) ) / ( full(is) - empt(is) )
            if (iclip.ne.0) void(is) = max(min(cliphi,void(is)),cliplo)
            end if
  180    continue
c
c *** Compute matrix and vector for least-squares fit of data.
c
      do 200 ifcn1 = 1, nfcn, 1
         bvec(ifcn1) = 0.
      do 200 ifcn2 = 1, nfcn, 1
         amat(ifcn1,ifcn2) = 0.
  200    continue
c
      do 260 is = ns1, ns2, 1
      do 240 ifcn1 = 1, nfcn, 1
         fcn1 = 1.
         if (nexp(ifcn1).ne.0) fcn1 = snrm(is)**nexp(ifcn1)
         bvec(ifcn1) = bvec(ifcn1) + void(is) * fcn1
      do 220 ifcn2 = 1, nfcn, 1
         fcn2 = 1.
         if (nexp(ifcn2).ne.0) fcn2 = snrm(is)**nexp(ifcn2)
         amat(ifcn1,ifcn2) = amat(ifcn1,ifcn2) + fcn1 * fcn2
  220    continue
  240    continue
  260    continue
c
c *** Solve the linear system.
c
      do 280 ifcn1 = 1, nfcn, 1
         cravf(ifcn1) = bvec(ifcn1)
         do 270 ifcn2 = 1, nfcn, 1
            cmat(ifcn1,ifcn2) = amat(ifcn1,ifcn2)
  270       continue
  280    continue
c
      call gauss1(cmat,nfcn,nfcnm,cravf)
c
c *** Compute the c and d coefficients needed to transform.
```

```
c
      do 300 ifcn1 = 1, nfcn, 1
      do 300 ifcn2 = 1, nfcn, 1
         ccoeff(ifcn1,ifcn2) = cfcn(ifcn1-1,ifcn2-1)
         dcoeff(ifcn1,ifcn2) = dfcn(ifcn1-1,ifcn2-1)
  300    continue
c
c *** Convert the ray averaged void fraction coefficients
c *** into radial void fraction coefficients.
c
      do 340 ifcn1 = 1, nfcn, 1
         caxvf(ifcn1) = 0.
      do 320 ifcn2 = 1, nfcn, 1
         caxvf(ifcn1) = caxvf(ifcn1) + ccoeff(ifcn1,ifcn2)*cravf(ifcn2)
  320    continue
  340    continue
c
c *** Calculate the ray averaged void fraction fit
c *** and the radial void fraction fit.
c
      do 380 is = 1, ns, 1
         ravf(is) = 0.
         axvf(is) = 0.
         if ((is.ge.ns1).and.(is.le.ns2)) then
            do 360 ifcn = 1, nfcn, 1
               fcn = snrm(is) ** nexp(ifcn)
               ravf(is) = ravf(is) + cravf(ifcn) * fcn
               axvf(is) = axvf(is) + caxvf(ifcn) * fcn
  360          continue
            end if
  380    continue
c
c *** Compute area-averaged void fraction and 1 - void fraction.
c
      avggas = 0.
      do 390 ifcn = 1, nfcn, 1
         avggas = avggas + caxvf(ifcn) * 2. / (2. + dfloat(nexp(ifcn)))
  390    continue
      avgliq = 1. - avggas
      write (6,2001) '   avggas = ', avggas
      write (6,2001) '   avgliq = ', avgliq
c
c *** Compute 1 - void fraction fit.
c
      do 400 ifcn = 1, nfcn, 1
         caxmf(ifcn) = - caxvf(ifcn)
         cramf(ifcn) = - cravf(ifcn)
         if (ifcn.eq.1) then
            caxmf(ifcn) = caxmf(ifcn) + 1.
            cramf(ifcn) = cramf(ifcn) + 1.
            end if
  400    continue
      do 410 ifcn = 1, nfcn, 1
         write (6,2001) '   caxvfi = ', caxvf(ifcn)
  410    continue
      do 420 ifcn = 1, nfcn, 1
         write  (6,1003) nexp(ifcn), caxvf(ifcn), cravf(ifcn),
     1                   caxmf(ifcn), cramf(ifcn)
  420    continue
c
c *** Write results to output files.
c
      write (6,2000) 'Writing output parameters to gdtaxi_out.dat'
```

```
      open (unit=30, status='unknown', file='gdtaxi_out.dat')
      write (30,1001) rinner
      write (30,1001) scentr
      write (30,1001) dxedge
      write (30,1002) iclip
      write (30,1001) cliplo
      write (30,1001) cliphi
      write (30,1001) tau
      write (30,1002) nexpm
      write (30,1001) avggas
      write (30,1001) avgliq
      do 430 ifcn = 1, nfcn, 1
         write (30,1001) caxvf(ifcn)
  430    continue
      do 440 ifcn = 1, nfcn, 1
         write (30,1003) nexp(ifcn), caxvf(ifcn), cravf(ifcn),
     1                   caxmf(ifcn), cramf(ifcn)
  440    continue
      close (unit=30)
c
      write (6,2000) 'Writing profiles to gdtaxi_gas.dat gdtaxi_liq.dat'
      open (unit=28, status='unknown', file='gdtaxi_gas.dat')
      open (unit=29, status='unknown', file='gdtaxi_liq.dat')
      do 450 is = 1, ns, 1
         if ((is.ge.ns1).and.(is.le.ns2)) then
            sc = spos(is) - scentr
            vmix = 1. - void(is)
            ramf = 1. - ravf(is)
            axmf = 1. - axvf(is)
            write (28,1004) snrm(is), void(is), ravf(is), axvf(is),
     1                      spos(is), sc
            write (29,1004) snrm(is), vmix,     ramf,     axmf,
     1                      spos(is), sc
            end if
  450    continue
      close (unit=28)
      close (unit=29)
c
      write (6,*) ' '
      write (6,1004) (caxvf(i),i=1,nfcn,1)
      write (6,*) ' '
c
c *** completed, stop.
c
      go to 999
  998 write (6,2000) '*** ABNORMAL STOP ***'
  999 stop 'gdtaxi'
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      function cfcn(m,n)
c
c *** Computes backward transformation coefficients.
c
      implicit double precision (a-h,o-z)
c
      cfcn = 0.
      if (m.le.n) then
         m2 = 2 * m
         n2 = 2 * n
         n2m2= n2 - m2
         nm = n - m
```

```
         fact = - dfloat(2 * m + 1) / dfloat(2**n2 * ( n2m2 - 1 ) )
         bin1 = bico(n2m2,nm)
         bin2 = bico(m2,m)
         cfcn = fact * bin1 * bin2
         end if
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      function dfcn(n,m)
c
c *** Computes forward transformation coefficients.
c
      implicit double precision (a-h,o-z)
c
      dfcn = 0.
      if (n.le.m) then
         m2 = 2 * m
         n2 = 2 * n
         m2n2= m2 - n2
         mn = m - n
         fact = dfloat(2**n2) / dfloat(2 * m + 1)
         bin1 = bico(m2n2,mn)
         bin2 = bico(m2,m)
         dfcn = fact * bin1 / bin2
         end if
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      subroutine gauss1(a,n,np,b)
c
c *** Gauss-Jordan elimination with full pivoting (Numerical Recipes).
c
      implicit double precision (a-h,o-z)
c
      parameter (nmax=10)
      dimension a(np,np), b(np)
      dimension ipiv(nmax), indxr(nmax), indxc(nmax)
c
      do 0100 j = 1, n, 1
         ipiv(j) = 0
 0100    continue
c
      do 0700 i = 1, n, 1
         big = 0.
         do 0250 j = 1, n, 1
            if (ipiv(j).ne.1) then
               do 0200 k = 1, n, 1
                  if (ipiv(k).eq.0) then
                     if (abs(a(j,k)).ge.big) then
                        big = abs(a(j,k))
                        irow = j
                        icol = k
                        end if
                  else if (ipiv(k).gt.1) then
                     pause 'Singular matrix'
                     end if
 0200                continue
```

```
            end if
 0250       continue
        ipiv(icol) = ipiv(icol) + 1
c
        if (irow.ne.icol) then
           do 0300 l = 1, n, 1
              dum = a(irow,l)
              a(irow,l) = a(icol,l)
              a(icol,l) = dum
 0300         continue
           dum = b(irow)
           b(irow) = b(icol)
           b(icol) = dum
           end if
        indxr(i) = irow
        indxc(i) = icol
        if (a(icol,icol).eq.0.) pause 'Singular matrix'
        pivinv = 1. / a(icol,icol)
        a(icol,icol) = 1.
        do 0400 l = 1, n, 1
           a(icol,l) = a(icol,l) * pivinv
 0400      continue
        b(icol) = b(icol) * pivinv
        do 0600 ll = 1, n, 1
           if (ll.ne.icol) then
              dum = a(ll,icol)
              a(ll,icol) = 0.
              do 0500 l = 1, n, 1
                 a(ll,l) = a(ll,l) - a(icol,l) * dum
 0500            continue
              b(ll) = b(ll) - b(icol) * dum
              end if
 0600      continue
 0700    continue
c
      do 0900 l = n, 1, -1
         if (indxr(l).ne.indxc(l)) then
            do 0800 k = 1, n, 1
               dum = a(k,indxr(l))
               a(k,indxr(l)) = a(k,indxc(l))
               a(k,indxc(l)) = dum
 0800          continue
            end if
 0900    continue
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      function bico(n,k)
c
c *** Binomial coefficient.
c
      implicit double precision (a-h,o-z)
c
      bico = anint(exp(factln(n)-factln(k)-factln(n-k)))
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
```

```
      function factln(n)
c
c *** Logarithm of factorial.
c
      implicit double precision (a-h,o-z)
c
      sum = 0.
      do 0100 i = 1, n, 1
         sum = sum + log(dfloat(i))
 0100    continue
c
      factln = sum
c
      return
      end
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
```