Temperature-Dependent Thomas-Fermi Model

W. R. Johnson
Department of Physics 225 Nieuwland Science Hall
Notre Dame University, Notre Dame, IN 46556

March 20, 2002

Abstract

This is the listing of the principal subroutines in a FORTRAN program to solve the temperature-dependent Thomas-Fermi equation iteratively.

Usage

1. Compile the routine
   f90 -O thomas.f -o thomas

2. Run the routine using sample input
   thomas < cu.in > cu.out

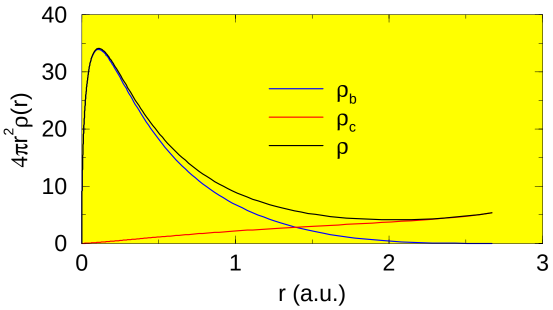
3. Examine the output files: cu.out with details of run, thomas.dat with atomic properties, and rho.dat with output suitable for graphic display. See the header of routine SELFC for a detailed description of these output in thomas.dat and rho.dat files.

4. sample input deck

```
Cu                    : atomic symbol (a4 format)
  29  63.55  8.92     : Z, Atomic Weight (gm/mol), density (gm/cc)
  -20.0  20.0         : mu0, mu1  (au) [bounds, mu0 < mu < mu1]
   1  10  1           : T2, T2, dT  (eV)
```

5. Routine has been checked out on the following:
   (a) Sun Blade 1000 workstation with f90
   (b) Dell 530 Linux (pentium iv xeon) with intel fortran compiler (ifc)
   (c) Dell PC (pentium iii) with Digital Fortran 5.5
   (d) SGI Origin with f90
   (e) LLNL open cluster (alpha) with f90

6. Sample graphical output from rho.dat for Cu at $T = 10\text{eV}$



Main Program (thomas)

```
      PROGRAM thomas
****************************************************************
*
*  Version: 03/20/02
*
*  Temperature-dependent Thomas-Fermi average-atom model
*
*  INPUT to program:    described in header of subroutine readat
*  OUTPUT from program: described in header of subroutine selfc
*
*  SUBROUTINES called:
*                 readat----
*                        purpose:  read in data from unit(5)
*                          input:  from unit 5 only
*                         output:
*                             jz  = nuclear charge Z
*                             aw  = atomic weight (gm/mol)
*                          rplas  = Wigner-Seitz radius (a0)
*                           amlo  = lower bound on mu (a.u.)
*                           amhi  = upper bound on mu (a.u.)
*                             T1  = initial temperature (eV)
*                             T2  = final temperature (eV)
*                             dT  = delta T (eV)
*
*                setgrd----
*                        purpose:  setup radial grid and initial potential
*                          input: (from readat)
*                             jz  = nuclear charge Z
*                             aw  = atomic weight (gm/mol)
*                           rplas = Wigner-Seitz radius (a0)
*                         output:
*                           mcav  = index of rplas on radial grid
*                             (through common)
*                           radial grid and z(r) initial charge distribution
*
*                 selfc----
*                          input: (from readat)
*                             jz  = nuclear charge Z
*                           amlo  = lower bound on mu (a.u.)
*                           amhi  = upper bound on mu (a.u.)
*                             T1  = initial temperature (eV)
*                             T2  = final temperature (eV)
*                             dT  = delta T (eV)
*                          input: (from setgrd)
*                            mcav = index of rplas on radial grid
*                         output: to unit(3) 'thomas.dat'
*                                 to unit(1) 'rho.dat'
*
*
******************************************************************************
      implicit doubleprecision(a-h,o-z)
      t0 = mclock()

***  initialize Gaussian coordinates for routine fdinc(aj,b,amu)

      call inint

***  read data for this run from unit 5

      call readat(jz,aw,rplas,amlo,amhi,T1,T2,dT)

***  set up radial grid and starting potential

      call setgrd(jz,aw,rplas,mcav,*911)

***  carry out self-consistent solution to TF equation.
      call selfc(jz,mcav,amlo,amhi,T1,T2,dT,*911)

      ttot = mclock()
      dt = (ttot-t0)/100d0
      write(6,1000) dt
 1000 format(/'  time =',f10.2,' sec')
      stop
 911  stop ' error exit'
      end
```

block data routine

```
      block data
***********************************************************************************
*
*    Useful constants:
*
*    alpha = 1/fine structure constant
*       pi = 3.1415...
*     bohr = bohr radius (Angstrom)
*       ev = Ry constant in electron volts
*      ryd = atomic unit in 1/cm
*
*********************************************************************************
      implicit doubleprecision(a-h,o-z)
      common/phycon/alpha,pi,ev,bohr,ryd
      data alpha /   137.035 989 5 d0/
      data pi    / 3.141 592 653 589 793 d0/
      data bohr  / 0.529 177 249 d0/
      data ev    /13.605 698 1 d0/
      data ryd   /   219 474.631 42 d0/
      end
```

First Subroutine (readat)

```
      subroutine readat(jz,aw,rplas,amlo,amhi,T1,T2,dT)
**********************************************************************
*
*  Read in data and calculate plasma radius
*
*  input card #1  ident (a4) atomic symbol
*  input card #2  jz = nuclear charge
*                 aw = atomic wgt
*                 den= density in gm/cc
*  input card #3  bounds on mu in (a.u.)  amlow < mu < amhi
*  input card #4   T(initial), T(final), delta T (all in eV)
*
*
*    OUTPUT:  jz = Z (nuclear charge)
*             aw = A (atomic weight)
*           rplas= R (Wigner-Seitz plasma radius)
*           amlo =  lower bound on mu (a.u.)
*           amhi =  upper bound on mu  (a.u.)
*           T1   =  initial temperature (eV)
*           T2   =  final temperature (eV)
*           dT   =  delta T (eV)
*
*    INPUT: only from unit 5 as given below
*
*******************************************************************
      implicit doubleprecision(a-h,o-z)
      character*4 ident
      character*8 version
      common/phycon/alpha,pi,ev,bohr,ryd
      data avag/6.022d-01/
      data version/'03/14/02'/

***  read ident (a4)

      read(5,1000)  ident
 1000 format(a)

***  read Z, A, density

      read(5,*)  jz,aw,den

*** Vol/mol = A/rho (cc/mol)

      vmol = aw/den

***  Vol/atom = Vol/mol / Atom/mol = vmol/avag (Angstrom^3/atom)
***  (A^3/cc = 10^24)

      vatom = vmol/avag

***  vatom = (4*pi/3)*Ratom^3

      ratom = (vatom*3d0/pi/4d0)**(1d0/3d0)

***  convert to a0 units

      rplas = ratom/bohr

      write(6,1010) ident,version,jz,aw,den,rplas

 1010 format(8x,' Thomas-Fermi for ',a,6x,'Version:',a//
     1       '               Z =',i4  ,'         A=',f7.2/
     2       '         density =',f6.2,'  rplas =',f7.4)

***  read bounds for mu in (a.u.)
      read(5,*) amlo, amhi
      write(6,1030) amlo,amhi
 1030 format('   Bounds: ',f10.1,' <  mu < ',f10.1 )

***  read  temperatures: Tini, Tfin, delta T (all in eV)

      read(5,*) T1,T2,DT

      write(6,1040) T1,T2,DT
 1040 format('  T1 =',f12.6,'  T2 =',f12.6,'  DT =',f12.6/)

      return
      end
```

Second Subroutine (setgrd)

```
      subroutine setgrd(jz,aw,rplas,mcav,*)
      implicit doubleprecision(a-h,o-z)
***********************************************************************
*
*  routine to set up grid and nuclear potential
*  revised May 26, 2000 for SCF in plasma
*  version 03/14/02
*
***********************************************************************
      parameter(NGP=500)
      common/radial/r(NGP),rp(NGP),rpor(NGP),h,max
      common/charge/znuc(NGP),z(NGP)
     &      /phycon/alpha,pi,ev,bohr,ryd
      data nmax /NGP/,hdef/0.03125/,rdef/5e-4/

      r0  = rdef
      h   = hdef
      max = nmax

      write(6,1000) r0,h,max
 1000 format(/' initial grid parameters:'
     &       /' r0=',f13.5,'  h=',f9.5,'   max=',i6)

**********    set up grid

      r(1)=0.0
      rp(1)=r0
      rpor(1)=0.0
      DO i=2,max
```

```
           rp(i)=dexp((i-1)*h)
           r(i)=r0*(rp(i)-1d0)
           rp(i)=r0*rp(i)
           rpor(i)=rp(i)/r(i)
      END DO

***  determine nearest point to plasma radius

      DO i = 1,max
        IF(r(i).lt.rplas) THEN
          mless = i
        END IF
      END DO
      mgrt = mless + 1
      dless = rplas - r(mless)
      dgrt = r(mgrt) - rplas
      IF(dgrt.lt.dless) THEN
        rnew = r(mgrt)
        mcav = mgrt
      ELSE
        rnew = r(mless)
        mcav = mless
      END IF

***  readjust r0 so that rplas = r(mcav)
      r00 = rplas/(dexp(h*(mcav-1))-1d0)

***  find new grid

      write(6,1010) r0,r00,mcav
 1010 format(' modified r0:'
     &       /' orig=',1p,e15.5,'  new=',e15.5,'  mcav=',i6/)

      rp(1)=r00
      DO i=2,max
         rp(i)=dexp((i-1)*h)
         r(i)=r00*(rp(i)-1d0)
         rp(i)=r00*rp(i)
         rpor(i)=rp(i)/r(i)
      END DO

***  verify that rplas = r(mcav)
      write(6,1020) mcav,r(mcav),rplas
 1020 format('  r(',i3,') =',f9.4/
     1       '  Rplas  =',f9.4/)
```

```
***  calculate nuclear rms radius (Johnson & Soff) ADNDT 33, 405 (1985)

      rnuc = 0.836d0 * aw**(1d0/3d0) + 0.570d0
      cnuc =sqrt(5d0/3d0)*rnuc
      c = 1d-5*cnuc/bohr

***  fill in nuclear potential and Coulomb potential

      DO i=1,max
        IF(r(i).lt.c) THEN
          z(i)=jz*r(i)*(1.5-0.5*(r(i)/c)**2)/c
          inuc=i
        ELSE
          z(i)=jz
        END IF
      END DO

      tnuc = 0
      write(6,1030) rnuc,cnuc,tnuc,inuc
 1030 format('  r(rms) =',f6.4,'  c =',f6.4,'   t =',f4.2/
     &       '  i(nuc) =',i5/)

****  add a square box potential to neutralize things outside cavity

      DO i = 1,mcav
         znuc(i) = z(i)
         z(i) = z(i) - jz*(1.5d0 - 0.5d0*(r(i)/rplas)**2)*r(i)/rplas
      END DO
      DO i = mcav+1,max
         znuc(i) = z(i)
         z(i) = 0d0
      END DO

***  initialize arrays for subroutine yfun

      call inidat

      return

 901  return 1
      end
```

Principal Subroutine (selfc)

```
      subroutine selfc(jz,mcav,amlo,amhi,T1,T2,dT,*)
***************************************************************************
*
*    Self-consistent solution to TF equation
*
*    INPUT :  jz = Z (nuclear charge)
*           mcav = index if rplas on radial grid [rplas = r(mcav)]
*           amlo =  lower bound on mu (a.u.)
*           amhi =  upper bound on mu  (a.u.)
*           T1   =  initial temperature (eV)
*           T2   =  final temperature (eV)
*           dT   =  delta T (eV)
*
*   OUTPUT:  file 'thomas.dat' connected here as unit(3)
*           for T = T1..T2 in step dT
*     the file contains in each record: (13f12.4)
*           TkeV = temperature  (keV)
*           aAng = R (Angstrom)
*         amukev = mu (keV)
*          PMbar = P (Mbar)
*          fpotn = E(pot)/ZkT
*          fkinn = E(kin)/ZkT
*          fpvn  = PV/ZkT
*            sok = S/k
*            tm1 = 5/2 PV (keV)
*            tm2 = 1/6 E(e-nuc) (keV)
*            tm3 = 7/6 E(e-e) (keV)
*            tm4 = - Z mu (keV)
*          tskev = TS (keV)
*
*            file 'rho.dat'  connected here as unit(1)
*        this is a summary file prepared to give the density as a function
*        of r suitable for graphic display:
*        for each i from 1 to mcv, we give in each record (1p,4e16.6)
*           r(i)    =  ith radial grid point
*           rhob(i) =  bound component of radial density
*           rhoc(i) =  continuum component of radial density
*           rho(i)  =  radial density
*
*********************************************************************
      parameter(NGP=500)
      implicit doubleprecision(a-h,o-z)

      common/radial/r(NGP),rp(NGP),rpor(NGP),h,max
```

```
     &      /phycon/alpha,pi,ev,bohr,ryd
     &      /charge/znuc(NGP),z(NGP)
     &      /density/rho(NGP)
      common/argment/a2,a3,a4,i5
      dimension u(NGP),v(NGP),y(NGP),rhoc(NGP)

      external fnorm
      data  NTMX /200/, epps /1d-9/
      data relerr /1d-12/, abserr /1d-12/, ferr /1d-14/
      data a2kv /0.027211d0/, a2Mbar /294.2101d0/

      q = jz

      DO i = 1,max
        u(i) = 0d0
        v(i) = 0d0
        rho(i) = 0d0
      END DO

****  convert temperatures to a.u.

      T1au =  T1/(2*ev)
      T2au =  T2/(2*ev)
      dTau =  dT/(2*ev)

      IF(T2au.eq.0d0.or.DTau.eq.0d0) THEN
          NTMP = 1
      ELSE
          NTMP = 1 + nint((T2au-T1au)/DTau)
      END IF

      open(unit=3,file=’thomas.dat’,form=’formatted’,status=’unknown’,
     1      position=’append’)

      DO NT = 1,NTMP

         Temp = T1au + dTau * (NT-1)

         write(6,1000) 2 * ev * Temp
 1000    format(’    kT =’,f12.5,’  eV’)

***  rho ==> 4 \pi r^2 rho    of notes

          frfac = 2d0*sqrt((2d0*Temp)**3)/pi

***  this starts an iteration loop over the effective potential
```

```
          delm = 1.0d0
          aold = 0d0

***   set up the four common arguments of fnorm(t) [q,Temp,frfac,mcav]
          a2 = q
          a3 = Temp
          a4 = frfac
          i5 = mcav

***    start of scf iteration loop

          DO ipot = 1,NTMX

            IF(ipot.lt.4) THEN
                dell = 1d-3
                ain1 = amlo
                ain2 = amhi
             ELSE IF(delm.lt.0.2d0) THEN
                ddd = 100*dmax1(epps,delm)
                ain1 = amu + ddd
                ain2 = amu - ddd
                dell = 1d-9
             ELSE
                ain1 = amu + 10*delm
                ain2 = amu - 10*delm
                dell = 1d-6
             END IF

*** solve    Z = int 4 pi r^2 rho(mu) for mu

             t = hybrid(fnorm,ain1,ain2,dell)

             amu = t

             itot = ipot

***  test convergence

            delm = abs(1d0-aold/amu)

            aold = amu

***  escape when delm < epss

            if(delm.lt.epps) go to 910
```

```
***  calculate the Hartree screening potential for this case

            l = 0
            call yfun(rho,y,l,mcav,*901)

***  use a 50-50 admixture to accelerate convergence

            DO i = 1,mcav
              z(i) = 0.5 * (z(i) + znuc(i) - y(i)*r(i))
            END DO

            DO i = mcav+1,max
               z(i) = 0d0
            END DO

****  end ipot loop

         END DO

***   end of iteration loop for a single temperature

 910  continue

***   summarize the iteration solution

       write(6,1010) itot
 1010  format('  Converged after ',i3,' loops')

***  calculate the continuum (E>0) contribution to the density.

        cau = amu/Temp
        ORD = 0.5d0
        u(1) = 0d0
        DO i = 2,mcav
          b = z(i)/(r(i)*Temp)
          x = cau + b
          if(b.lt.0.and.i.eq.mcav) then
            b = 0.0
          end if

***  fdinc(aj,b,amu) is the incomplete fermi-dirac integral

          rhoc(i) = fdinc(ORD, B, X)
          rhoc(i) = frfac * rhoc(i) * r(i)**2
          u(i) = rhoc(i) * rp(i)
          v(i) = (rho(i)-rhoc(i)) * rp(i)
        END DO
        encon = rint(u,1,mcav,7,h)
        enbnd = rint(v,1,mcav,7,h)
        Zi = rhoc(mcav)*r(mcav)/3

        write(6,1020) enbnd,encon,Zi
 1020   format('  Nbound =',f10.6,5x,'Ncont =',f12.6,
     2          '  Zion  =',f12.6)

****  end of loop: now, evaluate the pressure (au)

      xp = amu/Temp

      pfac = sqrt((2d0*Temp)**5)/(6*pi**2)

***  fd(aj,amu) is the fermi-dirac integral of (j,mu)

      ORD3 = 1.5d0
      fdd =  fd(ORD3,xp)

      Press = pfac *  fdd

      write(6,1030) Temp,amu,Press
 1030 format(4x,'kT =',f12.6,8x,'mu =',f12.6,6x,
     1           'P =',f12.6,' a.u.'/)

***  convert to practical units and write on unit 3

      TkeV = Temp*a2kv
      amukev = amu*a2kv
      PMbar = a2Mbar * Press

***  entropy and internal energy.

***  calculate Een and Eee

      DO i = 1,mcav
        u(i) = - znuc(i) * rho(i) * rpor(i)
        v(i) = y(i) * rho(i) * rp(i)
      END DO
      Een = rint(u,1,mcav,7,h)
      Eee = 0.5d0 * rint(v,1,mcav,7,h)

      vol = 4 * pi * r(mcav)**3 / 3d0
      entr = 2.5d0 * Press * vol + (Een+7*Eee)/6d0 - amu * jz

      ekin = 0.5 * (3 * Press * vol - Een - Eee)
      epot = Een + Eee

      ekinn = a2kv * ekin
      epotn = a2kv * epot
      epvn  = a2kv * Press * vol
      anmu = a2kv * jz * amu
      etotkv = a2kv * (ekin+epot)
      tskev = a2kv * entr
      tm1 = a2kv * 2.5d0 * Press * vol
      tm2 = a2kv * Een / 6d0
      tm3 = a2kv * Eee * 7d0/6d0
      tm4 = -anmu

      aAng = bohr * r(mcav)
      fkinn = ekin/(jz*Temp)
      fpotn = epot/(jz*Temp)
      fpvn  = Press * vol /(jz*Temp)
      sok = entr/Temp

      write(3,2000) TkeV,aAng,amukev,PMbar,fpotn,fkinn,fpvn,sok,
     1              tm1,tm2,tm3,tm4,tskev

 2000 format(13f12.4)

      END DO

      close(unit=3)

**** write output to a file for graphics in the final case

      open(unit=1,file='rho.dat',form='formatted',status='unknown')

      DO i = 1,mcav
       write(1,3000) r(i),rho(i)-rhoc(i),rhoc(i),rho(i)
 3000  format(1p,4e16.6)
      END DO

      return
 901  stop
      end
```

Normalization Function (fnorm)

```
      doubleprecision function fnorm(amu)
      implicit doubleprecision(a-h,o-z)
***************************************************************
*
*  This routine evaluates the function
*
*                   fnorm(mu) = Norm(mu) - Z
*  where
*                   Norm = Int_0^\R 4\pi r^2 rho(i)
*  with
*                   rho(i) = frfac * I_1/2[(mu-V(r))/kT]
*
*  the parameters (Z,T,frfac, and mcav) are transferred
*  from routine selfc through the common block
*         common/argment/q,Temp,frfac,mcav
*
***************************************************************
      parameter(NGP=500,RELERR=1d-14)
      common/radial/r(NGP),rp(NGP),rpor(NGP),h,max
     &      /charge/znuc(NGP),z(NGP)
     &      /density/rho(NGP)
      common/argment/q,Temp,frfac,mcav
      dimension u(NGP)

***  fill in the radial density rho(r(i))
        ord = 0.5d0
        u(1) = 0d0
        DO i = 2,mcav
          x = (amu+z(i)/r(i))/Temp
          IF(x.gt.10000d0) THEN
            rho(i) = 2d0*dsqrt(x**3)/3d0
          ELSE
            rho(i) = fd(ord,x)
          END IF
          rho(i) = frfac * rho(i) * r(i)**2
          u(i) = rho(i) * rp(i)
        END DO

***    here is the integral of the density
        Zf = rint(u,1,mcav,7,h)
        fnorm = Zf - q

        return
        end
```

Other utility routines included in package

- `hybrid(f,x1,x2,delx)` Function to find the zero of a function $f(x)$ in the interval $[x1, x2]$ to accuracy delx. A call to the function returns the value of the zero.

- `mclock()` Function to give time in 1/100th second. Usage: `t0 = mclock()` at beginning of routine and `t1 = mclock()` at end of routine. The difference `t1-t0` is the elapsed time in 1/100 sec.

- `rint (f,na,nb,nq,h)` Function to evaluate integral of an evenly spaced function $f(t(i))$ from point $t(n_a)$ to $t(n_b)$ using an n_q-point integration scheme. h is the spacing interval.

  $$\mathrm{rint}(f, n_a, n_b, n_q, h) = \int_{n_a h}^{n_b h} f(t(i))\, dt$$

  (Written by C. C. J. Roothaan.)

- `yfun(x,y,l,m,*)` Evaluates the Slater multipole (l) potential of array $x(r)$ on radial grid $r(i)$ and returns potential as the array $y(i) = v_l[x(i), r(i)]$. Calls auxiliary routine `yint (v,w,y,z,m,h)`. The subroutine `inidat()` must be called once just after the radial grid is set up and before the first call to `yfun` to initialize arrays used in `yfun`.

- `fd (xnu, alpha)` Fermi-Dirac function

  $$\mathrm{fd}(\nu, \alpha) = \int_0^\infty \frac{dy\ y^\nu}{1 + \exp(y - \alpha)}$$

  Adapted from the routine AADU by L. W. Fullerton, Comput. Phys. Commun. **39**, 181 (1986).

- `fdinc (aj,b,amu)` Incomplete Fermi-Dirac function

  $$\mathrm{fdinc}(j, b, \mu) = \int_b^\infty \frac{dy\ y^j}{1 + \exp(y - \mu)}$$

  Use 50-point Gaussian integral to evaluate difference with complete integral for $b < |\mu|$ or to evaluate the integral itself for $b > |\mu|$. At the start of the program, before any call to `fdint`, one must call routine `inint` once only. The program `inint` calls `setgau(xm,wm,n)` to initialize Gaussian points and weights.