APPENDIX B

NUMERICAL CODE OF THE CR MODEL

```
program CR

      implicit none
      include 'dati.fi'
      include 'parametri_cr.fi'

      include 'oggetti.fi'

      integer  i,iter,ndataaus
      double precision fnor(ndatamx)
      double precision maxw,dn
      external maxw
      include 'dati_ar.fi'
      include 'valori.fi'

c     Lettura dei parametri e dei dati iniziali

      call read_qu
      call read_data
      call read_data_cr

c     Calcolo delle particelle totali

      do i=1,nion_ar
        N=N+N_ar(i)
```

```
      end do

c     Inizializzazione delle variabilli

      Te=Te0

c     Inizializzazione fdst: Maxwell alla temperatura Te

      do i=1,ndata
        fdst(i)=maxw(energ(i))
      end do

      call terminenoto

10    continue
      iter=1

20     continue

      write(*,*)'ITERAZIONE',iter

      call discretizzazione

c     Inserimento del termine di arricchimento degli integrali di collisione anelastici

      call anelastiche

c     Soluzione della EEDF

      call matrice
      call soluzione

      do i=1,ndata
        fnor(i)=energ(i)**(0.5d0)*fdst(i)
      end do
      call trapezi(ndata,energ,fnor,1,ndata,integrale)
      write(*,*) 'integrale',integrale

c     Verifica dei criteri di convergenza della EEDF
```

```
      call convergenza
      call temp

      if ((diffrel.gt.conv).or.
     1          (dabs((Te-Te0)/Te0).gt.convtemp)) then
        if (iter.lt.itermax) then
          write(*,*)'diffrel',diffrel
          write(*,*)'Te',Te
                Te0=Te
          iter=iter+1
          goto 20
        else
          write(*,*) 'superato limite di iterazioni!',diffrel
          goto 50
        end if
      else
        write(*,*)'ITERAZIONE',iter,'diffrel',diffrel
        write(*,*) 'temperatura',Te
      end if

 50   continue

c     Soluzione del modello CR

      nk(n_cr+1)=N_ar(3)
      call matrix_cr
      call soluzione_cr

c     Verifica dei criteri di convergenza per N_ar(2)

      dn=dabs((N_ar(2)-nk(1))/N_ar(2))
      if (dn.gt.convN) then
        N_ar(2)=nk(1)
        write(*,*) 'Popolazione ArII',N_ar(2)
        goto 10
      else
      end if

c     Calcolo dei parametri di trasporto
```

```
      call zero
      call fzerocalc
      call funocalc
      call drift
      call freq
      alfa=ni/Vd

c     Scrittura dei risultati

      call writeres
      call write_nk

      stop
      end

      subroutine anelastiche

      implicit none
      include 'oggetti.fi'
      include 'dati.fi'
      include 'dati_ar.fi'
      include 'valori.fi'

c     Eccitazione e-Ar1 dal livello 1

      call ter_ecc(necc1_ar1,ndecc1_ar1,qecc1_ar1,N_ar(1),
     1             tecc1_ar1,Iecc1_ar1)

c     Eccitazione e-Ar2 dal livello 1

      call ter_ecc(necc1_ar2,ndecc1_ar2,qecc1_ar2,N_ar(2),
     1             tecc1_ar2,Iecc1_ar2)

c     Ionizzazione e-Ar dal livello 1

      call ter_ion(nion_ar,ndion_ar,qion_ar,N_ar,eion_ar,
     1             tion_ar,Iion_ar)

c     Stampa dei termini
```

```
c          open(unit=10,file='tion_ar.out',status='unknown')
c          do j=1,2*nion_ar
c            write(10,50) j
c            do i=1,ndata
c              write(10,50) i,enereV(i),tion_ar(i,3*j-2),tion_ar(i,3*j-1),
c     1                                    tion_ar(i,3*j),Iion_ar(i,j)
c            end do
c          end do
c          close (10)

c50        format (i4,g13.5,3g16.8,i6)

c          open(unit=10,file='tecc_ar.out',status='unknown')
c          do j=1,necc1_ar1
c            write(10,50) j
c            do i=1,ndata
c              write(10,50) i,enereV(i),tecc1_ar1(i,3*j-2),tecc1_ar1(i,3*j-1),
c     1                                    tecc1_ar1(i,3*j),Iecc1_ar1(i,j)
c            end do
c          end do
c          close (10)

c50        format (i4,g13.5,3g16.8,i6)

           return
           end
```

subroutine convergenza

```
           implicit none
           include 'oggetti.fi'
      double precision f1,f2,diff
           integer i

      diffrel=0.d0

           do i=1,ndata
            f1=fdst0(i)
            f2=fdst(i)
            if ((f1.eq.0.d0).or.((f1.eq.0.d0).and.(f2.eq.0.d0))) then
```

```
            goto 10
           else
                   diff=dabs((f1-f2)/f1)
            if (diff.gt.diffrel) then
              diffrel=diff
       else
            end if
           end if
10   continue
        end do

    return
        end

    subroutine deltax(uist,kist,dxist)

        implicit none
    double precision uist,kist,dxist
        include 'oggetti.fi'
        include 'dati.fi'
        include 'valori.fi'

        dxist=2*Kist/Uist

        return
        end

    subroutine discretizzazione

        implicit none
        integer i,iener,ie,ind,iq
        double precision aist,d,uist,kist,dxist,gammaist,p,passo,var
        double precision eaus,faus,fint,f1aus,f2aus
        include 'oggetti.fi'
        double precision maxw,f1(ndatamx),f2(ndatamx)
        include 'dati.fi'
        include 'valori.fi'

c       Inizializzazione delle variabili ausiliarie
```

```
            ndata0=ndata
            do i=1,ndata0
              energ0(i)=energ(i)
              fdst0(i)=fdst(i)
              f1(i)=energ0(i)**(1.5d0)*fdst0(i)
              f2(i)=energ0(i)**(0.5d0)*fdst0(i)
            end do

c           Inizio della discretizzazione
c           Punto 1

      i=1
            iq=1
            write(*,*) 'discretizzazione in corso.....enereV',enereV(i)
            energ(i)=e*enereV(i)
            iener=1
            ind=1

      call trovab
      call trovaa(iq,energ(i),aist,d)
            call kappa(f1,energ(i),iener,aist,var,kist)
            call udi(f2,energ(i),iener,aist,d,var,uist)
      call gammacalc(f2,energ(i),iener,aist,d,gammaist)
            call deltax(uist,kist,dxist)

      A(i)=aist
            DA(i)=d
            U(i)=uist
            K(i)=kist
            dx(i)=dxist
            gamma(i)=gammaist

c           Punti 2,ndata-1

            iener=2

      do i=2,ndatamx-1

              p=parametro
20            continue
```

```
      passo=p*dabs(dx(i-1))

            if (i.gt.2) then
                  if (passo.gt.2.d0*(energ(i-1)-energ(i-2))) then
                    passo=2.d0*(energ(i-1)-energ(i-2))
                  else
                  end if
            else
            end if
            if (passo.gt.5.d-1*e) then
                  passo=5.d-1*e
            else
            end if
      energ(i)=energ(i-1)+passo

            do ie=iener,ndata0
              if (energ(i).le.energ0(ie)) then
                iener=ie
                    goto 5
              else
              end if
            end do
5           continue

      fint=fdst0(iener)-fdst0(iener-1)
            fint=fint/(energ0(iener)-energ0(iener-1))
            fint=fdst0(iener-1)+fint*(energ(i)-energ0(iener-1))

            eaus=energ0(iener)
            faus=fdst0(iener)
            f1aus=f1(iener)
            f2aus=f2(iener)
            energ0(iener)=energ(i)
            fdst0(iener)=fint
            f1(iener)=energ(i)**(1.5d0)*fdst0(iener)
            f2(iener)=energ(i)**(0.5d0)*fdst0(iener)

            call trovaa(iq,energ(i),aist,d)
            call kappa(f1,energ(i),iener,aist,var,kist)
            call udi(f2,energ(i),iener,aist,d,var,uist)
```

```
            call gammacalc(f2,energ(i),iener,aist,d,gammaist)
            call deltax(uist,kist,dxist)

            energ0(iener)=eaus
            fdst0(iener)=faus
            f1(iener)=f1aus
            f2(iener)=f2aus

            if (dabs(dxist).lt.passo) then
          p=parametro*p
          goto 20
            else
                  A(i)=aist
                  DA(i)=d
                  U(i)=uist
                  K(i)=kist
                  dx(i)=dxist
                  gamma(i)=gammaist
          enereV(i)=energ(i)/e
          if (enereV(i).gt.enermax) then
              goto 10
             else
                  end if
            end if

          end do

10    write (*,*) ' numero di punti =',i

          ndata=i

c         Punto ndata

          enereV(ndata+1)=enereV(ndata)
          enereV(ndata)=enermax
          energ(ndata)=enermax*e
          energ(ndata+1)=enereV(ndata+1)*e
          write(*,*) 'discretizzazione effettuata.....enereV',enereV(ndata)

          call trovaa(iq,energ(ndata),aist,d)
```

```
            call kappa(f1,energ(ndata),ndata0,aist,var,kist)
            call udi(f2,energ(ndata),ndata0,aist,d,var,uist)
            call gammacalc(f2,energ(ndata),ndata0,aist,d,gammaist)
            call deltax(uist,kist,dxist)

      A(ndata)=aist
            DA(ndata)=d
            U(ndata)=uist
            K(ndata)=kist
            dx(ndata)=dxist
            gamma(ndata)=gammaist

c           call salvarisultati
            return
            end

      subroutine drift

      implicit none
      integer i
            include 'oggetti.fi'
            double precision faus(ndatamx)
            include 'dati.fi'
            include 'valori.fi'

            Vd=0.d0
            do i=1,ndata
              faus(i)=energ(i)*funo(i)
            end do
      call trapezi(ndata,energ,faus,1,ndata,Vd)
            Vd=Vd*8.d0*pi/(3.d0*m**2.d0)
            Vd=dabs(Vd)

            return
            end

            subroutine eccdis_e_cr(c,f)

c           Questa subroutine calcola gli integrali di collisione:
c           - ecc/disec con elettroni: c/f
```

```
implicit none
include 'parametri_cr.fi'
include 'oggetti.fi'
integer ir,i,j,il,iq,ld
double precision c(n_cr,n_cr),f(n_cr,n_cr)
double precision eh,a0,es,ur,bs,faus,p,q,dp
double precision qu(ndata),qua(ndata),qus(ndata)
double precision a_a,a_p,a_s,x,y,z,ej,a_p1(n_cr)
double precision vaus1(ndatamx),vaus2(ndatamx),eaus(ndatamx)
include 'dati.fi'
include 'dati_ar.fi'
include 'valori.fi'

write(*,*)'Calcolo termini ecc/disec con e'

eh=13.601*e
a0=5.292e-11
bs=1.0d0
a_a=5.0d-1
a_s=2.0d-1
a_p1(17)=1.19d-1
a_p1(18)=8.50d-2
a_p1(19)=1.68d-1
a_p1(20)=5.18d-5
a_p1(21)=2.65d-1
a_p1(22)=8.40d-2
a_p1(23)=5.60d-2
a_p1(24)=1.15d-1
a_p1(25)=8.60d-2
a_p1(26)=1.39d-1
a_p1(27)=3.60d-2
a_p1(30)=3.05d-2
a_p1(31)=1.75d-1
a_p1(32)=1.63d-1
a_p1(33)=1.68d-1
do i=34,n_cr
  a_p1(i)=2.0d-1
end do
```

```
      ir=1

      do il=ir+1,n_cr

       iq=1
       es=(elev(il)-elev(ir))*e
       do j=1,ndata
        if (energ(j).le.es) then
                eaus(j)=energ(j)
                vaus1(j)=0.d0
                vaus2(j)=0.d0
        else
                i=j
                eaus(j-1)=es
                goto 10
        end if
       end do
10     continue
       if (fo(ir,il).ne.0.d0) then
              do j=i,ndata
          ur=energ(j)/es
                qua(j)=4*Pi*(a0**2)*((eh/es)**2)*fo(ir,il)*a_a*ur**(-2.d0)
                qua(j)=qua(j)*(ur-1)*dlog(1.25*bs*ur)
                qus(j)=4*Pi*(a0**2)*a_s*(1-1/(ur**2))*ur**(-3.d0)
                qu(j)=qua(j)+qus(j)
              end do
       else if  ((l(ir).ne.10).and.(l(il).ne.10)) then
              ld=abs(l(ir)-l(il))
              if (ld.eq.1) then
                do j=i,ndata
                        ur=energ(j)/es
                        qua(j)=4*Pi*(a0**2)*((eh/es)**2)*fo(ir,il)*ur**(-2.d0)
                        qua(j)=qua(j)*a_a*(ur-1)*dlog(1.25*bs*ur)
                        qus(j)=4*Pi*(a0**2)*a_s*(1-1/(ur**2))*ur**(-3.d0)
                        qu(j)=qua(j)+qus(j)
                end do
              else
                do j=i,ndata
                        ur=energ(j)/es
                  qu=4*Pi*(a0**2)*a_p1(il)*(1-1/ur)
```

```
            end do
            end if
          else
           p=2*(13.601/(elev(n_cr+1)-elev(ir)))**5.d-1
            q=2*(13.601/(elev(n_cr+1)-elev(il)))**5.d-1
            dp=q-p
            if (y/p.le.1.d0) then
             fo(ir,il)=1.52*p**(-5.d0)*q**(-3.d0)
             fo(ir,il)=fo(ir,il)*((p**(-2.d0)-q**(-2.d0))**(-3.d0))
            else
             fo(ir,il)=1.95*p**(-5.d0)*q**(-3.d0)
             fo(ir,il)=fo(ir,il)*((p**(-2.d0)-q**(-2.d0))**(-3.d0))
            end if
            do j=i,ndata
             ur=energ(j)/es
             qua(j)=4*Pi*(a0**2)*((eh/es)**2)*fo(ir,il)*ur**(-2.d0)
             qua(j)=qua(j)*a_a*(ur-1)*dlog(1.25*bs*ur)
             qus(j)=4*Pi*(a0**2)*a_s*(1-1/(ur**2))*ur**(-3.d0)
             qu(j)=qua(j)+qus(j)
            end do
          end if
          do j=i,ndata
            eaus(j)=energ(j)
            vaus1(j)=qu(j)*fdst(j)*eaus(j)
            ej=eaus(j)-es
            call taylor(ndata,energ,ej,iq,x,y,z)
           if (iq.eq.1) then
             faus=fdst(1)*y+fdst(2)*z
           else if (iq.eq.ndata) then
             faus=fdst(ndata-1)*x+fdst(ndata)*y
           else
             faus=fdst(iq-1)*x+fdst(iq)*y+fdst(iq+1)*z
           end if
            vaus2(j)=qu(j)*faus*eaus(j)
          end do

c         Eccitazione

          call trapezi(ndata,eaus,vaus1,1,ndata,c(ir,il))
          c(ir,il)=c(ir,il)*(2.d0/m)**5.d-1
```

```
c       Diseccitazione

        call trapezi(ndata,eaus,vaus2,1,ndata,f(ir,il))
        f(ir,il)=f(ir,il)*(2.d0/m)**5.d-1
        f(ir,il)=f(ir,il)*g(ir)/(2*g(il))

       end do

       a_p=2.0d-1

       do ir=2,n_cr-1
        do il=ir+1,n_cr

              iq=1
         es=(elev(il)-elev(ir))*e
         do j=1,ndata
          if (energ(j).le.es) then
                    eaus(j)=energ(j)
            vaus1(j)=0.d0
            vaus2(j)=0.d0
         else
            i=j
                    eaus(j-1)=es
            goto 20
          end if
         end do
20       continue
         if (fo(ir,il).ne.0.d0) then
              do j=i,ndata
           ur=energ(j)/es
                    qua(j)=4*Pi*(a0**2)*((eh/es)**2)*fo(ir,il)*a_a*ur**(-2.d0)
                    qua(j)=qua(j)*(ur-1)*dlog(1.25*bs*ur)
                    qus(j)=4*Pi*(a0**2)*a_s*(1-1/(ur**2))*ur**(-3.d0)
                    qu(j)=qua(j)+qus(j)
            end do
           else if    ((l(ir).ne.10).and.(l(il).ne.10)) then
            ld=abs(l(ir)-l(il))
            if (ld.eq.1) then
              do j=i,ndata
```

```
                    ur=energ(j)/es
                    qua(j)=4*Pi*(a0**2)*((eh/es)**2)*fo(ir,il)*ur**(-2.d0)
                    qua(j)=qua(j)*a_a*(ur-1)*dlog(1.25*bs*ur)
                    qus(j)=4*Pi*(a0**2)*a_s*(1-1/(ur**2))*ur**(-3.d0)
                    qu(j)=qua(j)+qus(j)
                  end do
               else
                 do j=i,ndata
                    ur=energ(j)/es
                   qu=4*Pi*(a0**2)*a_p*(1-1/ur)
                  end do
               end if
             else
          p=2*(13.601/(elev(n_cr+1)-elev(ir)))**5.d-1
               q=2*(13.601/(elev(n_cr+1)-elev(il)))**5.d-1
               dp=q-p
               if (y/p.le.1.d0) then
                 fo(ir,il)=1.52*p**(-5.d0)*q**(-3.d0)
                  fo(ir,il)=fo(ir,il)*((p**(-2.d0)-q**(-2.d0))**(-3.d0))
               else
                  fo(ir,il)=1.95*p**(-5.d0)*q**(-3.d0)
                  fo(ir,il)=fo(ir,il)*((p**(-2.d0)-q**(-2.d0))**(-3.d0))
               end if
               if (fo(ir,il).gt.1.d0) then
c                 write(*,*)'ATTENZIONE fo!!!!',fo(ir,il),ir,il
                  fo(ir,il)=fo(ir,il)*5.0d-6
               end if
               do j=i,ndata
                  ur=energ(j)/es
                  qua(j)=4*Pi*(a0**2)*((eh/es)**2)*fo(ir,il)*ur**(-2.d0)
                  qua(j)=qua(j)*a_a*(ur-1)*dlog(1.25*bs*ur)
                  qus(j)=4*Pi*(a0**2)*a_s*(1-1/(ur**2))*ur**(-3.d0)
                  qu(j)=qua(j)+qus(j)
               end do
             end if

             do j=i,ndata
               eaus(j)=energ(j)
               vaus1(j)=qu(j)*fdst(j)*eaus(j)
               ej=eaus(j)-es
```

```
                call taylor(ndata,energ,ej,iq,x,y,z)
            if (iq.eq.1) then
                faus=fdst(1)*y+fdst(2)*z
            else if (iq.eq.ndata) then
                faus=fdst(ndata-1)*x+fdst(ndata)*y
            else
                faus=fdst(iq-1)*x+fdst(iq)*y+fdst(iq+1)*z
            end if
              vaus2(j)=qu(j)*faus*eaus(j)
            end do

c       Eccitazione

          call trapezi(ndata,eaus,vaus1,1,ndata,c(ir,il))
          c(ir,il)=c(ir,il)*(2.d0/m)**5.d-1

c       Diseccitazione

          call trapezi(ndata,eaus,vaus2,1,ndata,f(ir,il))
          f(ir,il)=f(ir,il)*(2.d0/m)**5.d-1
          f(ir,il)=f(ir,il)*g(ir)/(2*g(il))

        end do
      end do

c       Stampa dei coefficienti

c       open (unit=10,file='c.out',status='unknown')
c         write(10,50) ((c(i,j),j=1,n_cr),i=1,n_cr)
c       close (10)

c       open (unit=10,file='f.out',status='unknown')
c         write(10,50) ((f(i,j),j=1,n_cr),i=1,n_cr)
c       close (10)

c50     format(78e13.4)

        return
        end
```

```
      subroutine HAA(i,ai,bi,ci)

            implicit none
      integer i
            double precision isx,idx,ai,bi,ci
            include 'oggetti.fi'

      isx=energ(i)-energ(i-1)
            idx=energ(i+1)-energ(i)

      ai=1.d0/(isx*(isx+idx))*(2.d0-U(i)/K(i)*idx)
            bi=-1.d0/(isx*idx)*(2.d0+U(i)/K(i)*(isx-idx)-gamma(i)/K(i)*isx*idx)
            ci=1.d0/(idx*(isx+idx))*(2+U(i)/K(i)*isx)

            return
            end

            subroutine emisspont_cr(ac)

c           Questa subroutine calcola gli integrali di collisione:
c           - emissione spontanea: ac

            implicit none
            integer ir,il
            include 'parametri_cr.fi'
            include 'oggetti.fi'
            double precision ac(n_cr,n_cr),es
            include 'dati.fi'
            include 'dati_ar.fi'
            include 'valori.fi'

            write(*,*)'Calcolo termini di emissione spontanea'

            do ir=1,n_cr-1
              do il=ir+1,n_cr
                es=(elev(il)-elev(ir))*e
                      ac(ir,il)=fo(ir,il)*(es**2)*g(ir)/g(il)
                      ac(ir,il)=ac(ir,il)/(1.5d0*(h*cl*1.0d2)**2)
              end do
            end do
```

```
      return
      end

  subroutine freq

implicit none
integer i,j,ii,iq
      include 'oggetti.fi'
      double precision mu(10),vaus(ndatamx)
      double precision qu,x,y,z
      include 'dati.fi'
      include 'dati_ar.fi'
      include 'valori.fi'

ni=0.d0
      do ii=1,nion_ar
       iq=1
       do j=1,ndion_ar(ii)
        x0(j)=qion_ar(j,2*ii-1)
        f0(j)=qion_ar(j,2*ii)
       end do
       mu(ii)=0.d0
       do i=1,ndata
        if (energ(i).le.eion_ar(ii)) then
                qu=0.d0
        else
                call taylor(ndion_ar(ii),x0,energ(i),iq,x,y,z)
                        if (iq.eq.1) then
                         qu=f0(1)*y+f0(2)*z
                        else if (iq.eq.ndion_ar(ii)) then
                         qu=f0(ndion_ar(ii)-1)*x+f0(ndion_ar(ii))*y
                        else
                         qu=f0(iq-1)*x+f0(iq)*y+f0(iq+1)*z
                        end if
               end if
        if (qu.lt.0.d0) then
          write(*,*) 'MMMMMMMMMMMMMMMMMM'
        else
        end if
```

```
                vaus(i)=qu*fzero(i)*energ(i)
          end do
          call trapezi(ndata,energ,vaus,1,ndata,mu(ii))
          mu(ii)=8.d0*Pi*N_ar(ii)*mu(ii)/(m*m)
        end do

        do i=1,nion_ar
          ni=ni+mu(i)
        end do

    return
        end
```

subroutine funocalc

```
        implicit none
        include 'dati.fi'
        include 'oggetti.fi'
    integer i
        double precision caus
        include 'valori.fi'

        caus=e*campo

        do i=1,ndata
          funo(i)=caus*dfzero(i)/A(i)-alfa*fzero(i)/A(i)
    end do

        return
        end
```

subroutine fzerocalc

```
C******calcolo fzero e la sua derivata rispetto alla energia**********************************

    implicit none
        include 'dati.fi'
        include 'oggetti.fi'
    double precision a1
        double precision denerdx,denersx,dener
```

```
      integer i
      include 'valori.fi'

            a1=4.d0*Pi*dsqrt(2.d0)/m**1.5d0

            do i=1,ndata
              fzero(i)=fdst(i)/a1
      end do

c           Derivata dfzero numerica

      dfzero(1)=(fzero(2)-fzero(1))/(energ(2)-energ(1))
            do i=2,ndata-1
              denerdx=energ(i+1)-energ(i)
              denersx=energ(i)-energ(i-1)
              dener=denerdx+denersx
              dfzero(i)=(fzero(i+1)-fzero(i))*denersx/(denerdx*dener)+
     1       (fzero(i)-fzero(i-1))*denerdx/(denersx*dener)
            end do
            dfzero(ndata)=(fzero(ndata)-fzero(ndata-1))
     1/(energ(ndata)-energ(ndata-1))

            return
            end

      subroutine gammacalc(f,ener,ien,aist,d,gammaist)

            implicit none
      double precision caus1,caus2,caus3,ener,aist,d,gammaist
      double precision vaus1,vaus2,vaus3,vaus4,vaus5
      integer i,j,ien,iq
            double precision qu,x,y,z
            include 'dati.fi'
            include 'oggetti.fi'
            double precision f(ndatamx)
            include 'dati_ar.fi'
            include 'valori.fi'

      caus1=(2.d0/3.d0)*alfa*e*campo/m**2.d0
            caus2=-(2.d0/3.d0)*(alfa/m)**2.d0
```

```
      caus3=-4.d0/(m*m_ar)

      vaus1=caus1*(1.d0/aist-(ener/aist**2.d0)*d)
      vaus2=caus2*ener/aist
      vaus3=caus3*(2.d0*ener*aist+ener*ener*d)

c     Contributo di Fokker-Plank

    vaus4=-f(ien)*B

c     Somma dei contributi

      gammaist=vaus1+vaus2+vaus3+vaus4

c     Termine di impoverimento della ionizzazione dell'Argon

      do i=1,nion_ar
       iq=1
       do j=1,ndion_ar(i)
        x0(j)=qion_ar(j,2*i-1)
        f0(j)=qion_ar(j,2*i)
       end do
       if (ener.le.eion_ar(i)) then
            qu=0.d0
       else
            call taylor(ndion_ar(i),x0,ener,iq,x,y,z)
             if (iq.eq.1) then
                  qu=f0(1)*y+f0(2)*z
             else if (iq.eq.ndion_ar(i)) then
                  qu=f0(ndion_ar(i)-1)*x+f0(ndion_ar(i))*y
             else
                  qu=f0(iq-1)*x+f0(iq)*y+f0(iq+1)*z
             end if
       end if
       if (qu.lt.0.d0) then
        write(*,*) 'gamma.f_ion',qu,i,ener
       else
       end if
       vaus5=qu*2.d0*N_ar(i)/(m*m)*ener
       gammaist=gammaist+vaus5
```

```
      end do

c     Termine di impoverimento dell'eccitazione dell'ArI

      vaus5=0.d0
      do i=1,necc1_ar1
       iq=1
       do j=1,ndecc1_ar1(i)
        x0(j)=qecc1_ar1(j,2*i-1)
        f0(j)=qecc1_ar1(j,2*i)
       end do
       if (ener.le.qecc1_ar1(1,2*i-1)) then
            qu=0.d0
       else
            call taylor(ndecc1_ar1(i),x0,ener,iq,x,y,z)
             if (iq.eq.1) then
                  qu=f0(1)*y+f0(2)*z
             else if (iq.eq.ndecc1_ar1(i)) then
                  qu=f0(ndecc1_ar1(i)-1)*x+f0(ndecc1_ar1(i))*y
             else
                  qu=f0(iq-1)*x+f0(iq)*y+f0(iq+1)*z
             end if
       end if
       if (qu.lt.0.d0) then
        write(*,*) 'gamma.f_ecc_Ar1',qu,i,ener
       else
       end if
       vaus5=qu*2.d0*N_ar(1)/(m*m)*ener
       gammaist=gammaist+vaus5
      end do

c     Termine di impoverimento dell'eccitazione dell'ArII

      vaus5=0.d0
      do i=1,necc1_ar2
       iq=1
       do j=1,ndecc1_ar2(i)
        x0(j)=qecc1_ar2(j,2*i-1)
        f0(j)=qecc1_ar2(j,2*i)
       end do
```

```
        if (ener.le.qecc1_ar2(1,2*i-1)) then
                qu=0.d0
        else
                call taylor(ndecc1_ar2(i),x0,ener,iq,x,y,z)
                 if (iq.eq.1) then
                        qu=f0(1)*y+f0(2)*z
                 else if (iq.eq.ndecc1_ar2(i)) then
                        qu=f0(ndecc1_ar2(i)-1)*x+f0(ndecc1_ar2(i))*y
                 else
                        qu=f0(iq-1)*x+f0(iq)*y+f0(iq+1)*z
                 end if
        end if
        if (qu.lt.0.d0) then
          write(*,*) 'gamma.f_ecc_Ar2',qu,i,ener
        else
        end if
        vaus5=qu*2.d0*N_ar(2)/(m*m)*ener
        gammaist=gammaist+vaus5
      end do

      return
      end
```

subroutine intnorm

```
      implicit none
      double precision mu
      include 'oggetti.fi'
      integer i

integrale=0.d0
      do i=2,ndata
  mu=fdst(i-1)*dsqrt(energ(i-1))+fdst(i)*dsqrt(energ(i))
        mu=mu*(energ(i)-energ(i-1))/2.d0
        integrale=integrale+mu
end do

      return
      end
```

```
      subroutine ionric_e_cr(s,q)

c     Questa subroutine calcola gli integrali di collisione:
c     - ioniz/ricomb con elettroni: s/o

      implicit none
      integer i,j,iq,ir,iel
      include 'parametri_cr.fi'
      include 'oggetti.fi'
      double precision es,faus,a1,b1,eh,a0,ur
      double precision vaus1(ndatamx),vaus2(ndatamx),eaus(ndatamx)
      double precision qu,x,y,z,ej
      double precision s(n_cr),q(n_cr)
      include 'dati.fi'
      include 'dati_ar.fi'
      include 'valori.fi'

      write(*,*)'Calcolo termini ioniz/ricomb con e'

      eh=13.601*e
      a0=5.292e-11
      do j=1,ndion_ar(2)
       x0(j)=qion_ar(j,2*2-1)
       f0(j)=qion_ar(j,2*2)
      end do
      iel=1

      do ir=1,n_cr
       iq=1
       es=(elev(n_cr+1)-elev(ir))*e
       do j=1,ndata
        if (energ(j).le.es) then
              eaus(j)=energ(j)
              vaus1(j)=0.d0
              vaus2(j)=0.d0
        else
              i=j
              eaus(j-1)=es
              goto 10
        end if
```

```
      end do
10    continue
      do j=i,ndata
            eaus(j)=energ(j)
            ur=eaus(j)/es
       if (ir.eq.1) then
             a1=3.5d-1
             b1=1.5
            else if (ir.le.37) then
             a1=4.0d-1
             b1=4.0
            else
             a1=6.7d-1
             b1=1.0
            end if
            qu=4*Pi*(a0**2)*((eh/es)**2)*iel*a1*ur**(-2.d0)
            qu=qu*(ur-1)*dlog(1.25*b1*ur)
            vaus1(j)=qu*fdst(j)*eaus(j)
            ej=eaus(j)-es
            call taylor(ndata,energ,ej,iq,x,y,z)
       if (iq.eq.1) then
             faus=fdst(1)*y+fdst(2)*z
       else if (iq.eq.ndata) then
             faus=fdst(ndata-1)*x+fdst(ndata)*y
       else
             faus=fdst(iq-1)*x+fdst(iq)*y+fdst(iq+1)*z
       end if
            vaus2(j)=qu*faus*eaus(j)
      end do

c     Ionizzazione

      call trapezi(ndata,eaus,vaus1,1,ndata,s(ir))
      s(ir)=s(ir)*(2.d0/m)**5.d-1

c     Ricombinazione

      call trapezi(ndata,eaus,vaus2,1,ndata,q(ir))
      q(ir)=q(ir)*(2.d0/m)**5.d-1
      q(ir)=q(ir)*g(ir)/(2*g(n_cr+1))
```

```
          q(ir)=q(ir)*((h*h)/(2*Pi*m*kboltz*Te))**1.5d0

        end do

        return
        end

      subroutine kappa(fa,ener,ien,aist,var,kist)

      implicit none
        integer i,ien
        include 'dati.fi'
        include 'oggetti.fi'
      double precision var1,var2,var,vaus1,vaus2,vaus3,ener,aist,kist
        double precision fa(ndatamx)
      include 'dati_ar.fi'
      include 'valori.fi'

        vaus1=-(2.d0/3.d0)*(e*campo/m)**2.d0*ener/aist
        vaus2=-kboltz*Th/(m*m_ar)*aist*4.d0*ener*ener

c       Contributo di Fokker-Plank

        if (ien.eq.1) then
          var1=0.d0
        else
       call trapezi(ndata0,energ0,fa,1,ien,var1)
        end if
        if (ien.eq.ndata0) then
          var2=0.d0
        else
          call trapezi(ndata0,energ0,fdst0,ien,ndata0,var)
       var2=var*energ0(ien)**(1.5d0)
        end if
      vaus3=(-2.d0/3.d0)*B*(var1+var2)

c       Somma totale dei contributi

      kist=vaus1+vaus2+vaus3
```

```
      return
      end

      subroutine matrice

      implicit none
      integer i,j,jj,iaus,g,ik,icont,ntot
      integer min,jtemp
      parameter (icont=100)
      integer jaus(icont)
      double precision elem1,elem2,elem3,atemp
      double precision aaus(icont)
      include 'dati.fi'
      include 'oggetti.fi'
      include 'dati_ar.fi'
      include 'valori.fi'

c     Gli elementi della matrice sono memorizzati nei seguenti vettori:
c     AA    vettore degli elementi non nulli
c     IA    indice di riga
c     JA    indice di colonna
c     Gli elementi sono memorizzati riga per riga

      do i=1,ndatamx
       AA(i)=0.d0
       IA(i)=0
       JA(i)=0
      end do

c     Numero dei processi anelastici totali considerati

      ntot=necc1_ar1+necc1_ar2+2*nion_ar

c     La prima riga della matrice contiene la condizione di normalizzazione

      AA(1)=5.d-1*dsqrt(energ(1))*(energ(2)-energ(1))
      IA(1)=1
      JA(1)=1
      do i=2,ndata-1
       AA(i)=5.d-1*dsqrt(energ(i))*(energ(i+1)-energ(i-1))
```

```
      IA(i)=1
      JA(i)=i
    end do
    AA(ndata)=5.d-1*dsqrt(energ(ndata))*(energ(ndata)-energ(ndata-1))
    IA(ndata)=1
    JA(ndata)=ndata

c   Inserimento dei coefficienti della matrice da 2 a ndata-1

    iaus=ndata
    iaus=iaus+1

    do i=2,ndata-1

c   Inserimento dei termini della diagonale principale e delle 2 co-diagonali

      call HAA(i,elem1,elem2,elem3)
      AA(iaus)=elem1
      IA(iaus)=i
      JA(iaus)=i-1
      iaus=iaus+1
      AA(iaus)=elem2
      IA(iaus)=i
      JA(iaus)=i
      iaus=iaus+1
      AA(iaus)=elem3
      IA(iaus)=i
      JA(iaus)=i+1

c   Inserimento dei termini di arricchimento delle collisioni anelastiche

      iaus=iaus+1
      ik=1
      do j=1,3*ntot
            aaus(j)=0.d0
            jaus(j)=0
      end do

c   Eccitazione e-ArI
```

```
      do j=1,necc1_ar1
            g=Iecc1_ar1(i,j)
            if (g.ne.0) then
             aaus(ik)=tecc1_ar1(i,3*j-2)
             jaus(ik)=g-1
             ik=ik+1
             aaus(ik)=tecc1_ar1(i,3*j-1)
             jaus(ik)=g
             ik=ik+1
             if (g.eq.ndata) then
                  continue
             else
               aaus(ik)=tecc1_ar1(i,3*j)
               jaus(ik)=g+1
               ik=ik+1
             end if
            else
             goto 10
            end if
   end do
10      continue

c       Eccitazione e-ArII

      do j=1,necc1_ar2
            g=Iecc1_ar2(i,j)
            if (g.ne.0) then
             aaus(ik)=tecc1_ar2(i,3*j-2)
             jaus(ik)=g-1
             ik=ik+1
             aaus(ik)=tecc1_ar2(i,3*j-1)
             jaus(ik)=g
             ik=ik+1
             if (g.eq.ndata) then
                  continue
             else
               aaus(ik)=tecc1_ar2(i,3*j)
               jaus(ik)=g+1
               ik=ik+1
             end if
```

```
                else
                  goto 20
                end if
      end do
20          continue

c           Ionizzazioni e-Ar

            do j=1,2*nion_ar
                g=Iion_ar(i,j)
                if (g.ne.0) then
                  aaus(ik)=tion_ar(i,3*j-2)
                  jaus(ik)=g-1
                  ik=ik+1
                  aaus(ik)=tion_ar(i,3*j-1)
                  jaus(ik)=g
                  ik=ik+1
                  if (g.eq.ndata) then
                        continue
                  else
                    aaus(ik)=tion_ar(i,3*j)
                    jaus(ik)=g+1
                    ik=ik+1
                  end if
                else
                  goto 30
                end if
            end do
30          continue

c           Ordinamento crescente del vettore aaus in base all'indice di colonna jaus

            do jj=1,ik-2
              min=jj
                do j=jj+1,ik-1
                  if (jaus(j).lt.jaus(min)) then
                        min=j
                end if
              end do
                if (min.gt.jj) then
```

```
                jtemp=jaus(min)
                atemp=aaus(min)
                jaus(min)=jaus(jj)
                aaus(min)=aaus(jj)
                jaus(jj)=jtemp
                aaus(jj)=atemp
              else
              end if
          end do

c         Costruisco il vettore AA sommando i termini che hanno l'indice di colonna comune

          do ik=1,3*ntot
              if (jaus(ik).ne.0) then
                AA(iaus)=aaus(ik)
                IA(iaus)=i
                JA(iaus)=jaus(ik)
                do j=ik+1,3*ntot
                  if (jaus(j).eq.jaus(ik)) then
                      AA(iaus)=AA(iaus)+aaus(j)
                      jaus(j)=0
                    else
            end if
                  end do
                  iaus=iaus+1
                  goto 40
              else
              end if
40            continue
          end do

    end do

c         Ultima riga

          call HAA(ndata,elem1,elem2,elem3)
          AA(iaus)=elem1
          IA(iaus)=ndata
    JA(iaus)=ndata-1
          iaus=iaus+1
```

```
      AA(iaus)=elem2
            IA(iaus)=ndata
      JA(iaus)=ndata

            nzero=iaus

c           Stampa dei vettori

c           open (unit=10,file='matrix.out',status='unknown')
c             write(10,48) 'elementi non nulli = ',nzero
c             write(10,49) 'IA','JA','AA'
c             do i=1,nzero
c               write(10,50) IA(i),JA(i),AA(i)
c             end do
c           close (10)

c148        format(a20,i6)
c149        format(2a6,a14)
c150        format(2i6,g16.8)

            return
            end

            subroutine matrix_cr

            implicit none
            integer i,j,k
            include 'parametri_cr.fi'
            include 'dati.fi'
            double precision aus1,aus2,aus3
            double precision ac(n_cr,n_cr),la(n_cr,n_cr),c(n_cr,n_cr)
            double precision s(n_cr),q(n_cr),lr(n_cr),r(n_cr),f(n_cr,n_cr)

c           Gli elementi della matrice sono memorizzati nei seguenti vettori:
c           a_cr      vettore degli elementi non nulli
c           Ia_cr     indice di riga
c           Ja_cr     indice di colonna

c           Inizializzazione delle variabili
```

```
      do i=1,n_cr
       do j=1,n_cr
                a_cr(i,j)=0.d0
       end do
      end do

      do i=1,n_cr
       lr(i)=1
       sc_cr=0.d0
      end do

      do i=1,n_cr
       do j=1,n_cr
        la(i,j)=1
       end do
      end do

      alfa_cr=0.d0
      s_cr=0.d0

c     Calcolo degli integrali di collisione

      call ionric_e_cr(s,q)
      call ricrad_cr(r)
      call eccdis_e_cr(c,f)
      call emisspont_cr(ac)

      write(*,*)'Calcolo matrice'

c     Prima riga: condizione di neutralità del plasma

      do j=1,n_cr
       a_cr(1,j)=1
      end do

c     Righe 2-n_cr: rate equations

      do k=2,n_cr

       do j=1,k-1
```

```
          a_cr(k,j)=ne*c(j,k)
         end do

         aus1=0.d0
         aus2=0.d0
         do j=1,k-1
          aus1=aus1+ne*f(j,k)+ac(j,k)*la(j,k)
         end do
         do j=k+1,n_cr
          aus2=aus2+ne*c(k,j)
         end do
         aus3=ne*s(k)
         a_cr(k,k)=-(aus1+aus2+aus3)

         do j=k+1,n_cr
          a_cr(k,j)=ne*f(k,j)+ac(k,j)*la(k,j)
         end do

        end do

c       Termine noto

        tn_cr(1)=ne-2*nk(n_cr+1)
        do k=2,n_cr
         tn_cr(k)=-nk(n_cr+1)*(ne*ne*q(k)+ne*lr(k)*r(k))
        end do

c       Calcolo di alfa_cr e sc_cr

        do j=1,n_cr
         alfa_cr=alfa_cr+ne*q(j)+lr(j)*r(j)
         sc_cr(j)=ne*s(j)
        end do
        alfa_cr=ne*nk(n_cr+1)*alfa_cr

c       Stampa della matrice

c       open (unit=10,file='matrix_cr.out',status='unknown')
c        write(10,50) ((a_cr(i,j),j=1,n_cr+1),i=1,n_cr+1)
c       close (10)
```

```
c50     format(79e13.4)

        return
        end

    double precision function maxw(x)

        implicit none
        double precision x
        include 'dati.fi'

        costmaxw=2.d0/(dsqrt(Pi)*(kboltz*Te)**1.5d0)
    maxw=costmaxw*dexp(-x/(kboltz*Te))

    return
        end

        subroutine read_qu

        implicit none
    integer i,j
        include 'dati.fi'
        include 'oggetti.fi'
        include 'dati_ar.fi'

c       Lettura delle sezioni d'urto momentum transfer e-ArI

        open(unit=10,file='qel-ar.txt',status='old',readonly)
          read(10,*) ndel_ar
          do j=1,ndel_ar
                read(10,*) qel_ar(j,1),qel_ar(j,2)
            qel_ar(j,1)=qel_ar(j,1)*e
            qel_ar(j,2)=qel_ar(j,2)*1.d-20
          end do

c       Lettura di tutte le sezioni d'urto di ionizzazione e-Ar: 1-i

        open (unit=10,file='qion-ar.txt',status='old',readonly)
          read(10,*) nion_ar
```

```
      do i=1,nion_ar
       read(10,*) ndion_ar(i)
            do j=1,ndion_ar(i)
              read(10,*) qion_ar(j,2*i-1),qion_ar(j,2*i)
        qion_ar(j,2*i-1)=qion_ar(j,2*i-1)*e
        qion_ar(j,2*i)=qion_ar(j,2*i)*1.d-20
            end do
      end do
     close (10)

c     Lettura delle sezioni d'urto di eccitazione e-ArI: 1-n

     open (unit=10,file='qecc1-ar1.txt',status='old',readonly)
      read(10,*) necc1_ar1
      do i=1,necc1_ar1
       read(10,*) ndecc1_ar1(i)
            do j=1,ndecc1_ar1(i)
              read(10,*) qecc1_ar1(j,2*i-1),qecc1_ar1(j,2*i)
        qecc1_ar1(j,2*i-1)=qecc1_ar1(j,2*i-1)*e
        qecc1_ar1(j,2*i)=qecc1_ar1(j,2*i)*1.d-20
            end do
      end do
     close (10)

c     Lettura delle sezioni d'urto di eccitazione e-ArII: 1-n

     open (unit=10,file='qecc1-ar2.txt',status='old',readonly)
      read(10,*) necc1_ar2
      do i=1,necc1_ar2
       read(10,*) ndecc1_ar2(i)
            do j=1,ndecc1_ar2(i)
              read(10,*) qecc1_ar2(j,2*i-1),qecc1_ar2(j,2*i)
        qecc1_ar2(j,2*i-1)=qecc1_ar2(j,2*i-1)*e
        qecc1_ar2(j,2*i)=qecc1_ar2(j,2*i)*1.d-20
            end do
      end do
     close (10)

     return
     end
```

```
      subroutine read_data

      implicit none
  integer i,j
      include 'dati.fi'
      include 'oggetti.fi'
      include 'dati_ar.fi'

c     Lettura dei parametri di ingresso

      open (unit=10,file='dati.txt',status='old',readonly)
       read(10,*) Te0
       read(10,*) Th
       read(10,*) campo
       read(10,*) Ne
       do i=1,nion_ar
        read(10,*) N_ar(i)
       end do
      close (10)

c     Lettura del vettore iniziale delle energie [J]

      open(unit=10,file='fdata.txt',status='old',readonly)
       read (10,*) ndata
       do i=1,ndata
            read(10,*) energ(i)
    end do
      close(10)

      return
      end

      subroutine read_data_cr

      implicit none
  integer i,j
      include 'parametri_cr.fi'
      include 'oggetti.fi'
      include 'dati.fi'
```

```
      include 'valori.fi'

c     Lettura dei dati dei livelli energetici

      open (unit=10,file='dati_liv.txt',status='old',readonly)
       do i=1,n_cr+1
            read(10,*) elev(i),g(i),l(i)
    end do
      close (10)

c     Lettura della f osc

      open (unit=10,file='freq_osc.txt',status='old',readonly)
       do i=1,n_cr-1
            do j=i+1,n_cr
             read(10,*) fo(i,j)
    end do
      end do
      close (10)

      return
      end

      subroutine ricrad_cr(r)

c     Questa subroutine calcola gli integrali di collisione:
c     - ricombinazione radiativa: r

      implicit none
      integer ir,j,i,iq
      include 'oggetti.fi'
      include 'parametri_cr.fi'
      double precision x,y,z,ej,es,faus,eh
      double precision r(n_cr),vaus(ndatamx),eaus(ndatamx)
      double precision qu(ndatamx)
      include 'dati.fi'
      include 'valori.fi'

      write(*,*)'Calcolo termini di ricombinazione radiativi'
```

```
      eh=13.601*e
      do ir=1,n_cr
       es=(elev(n_cr+1)-elev(ir))*e
       do j=1,ndata
        if (energ(j).le.es) then
              eaus(j)=energ(j)
              vaus(j)=0.d0
        else
              i=j
              eaus(j-1)=es
              goto 10
        end if
       end do
10     continue
       if (ir.eq.1) then
        do j=i,ndata
              if (energ(j).le.2*eh) then
                    qu(j)=3.5d-21
              else
                    qu(j)=2.8d-20*(eh/energ(j))**3
              end if
             end do
       else if (ir.ge.2.and.ir.le.16) then
        do j=i,ndata
              if (energ(j).le.5.9d-1*eh) then
                    qu(j)=2.0d-22*0.06
              else
                    qu(j)=7.91d-22*0.06*((es/eh)**2.5d0)*(eh/energ(j))**3
              end if
             end do
       else
        do j=i,ndata
              qu(j)=7.91d-22*0.1*((es/eh)**2.5d0)*(eh/energ(j))**3
             end do
       end if
       do j=i,ndata
              eaus(j)=energ(j)
              iq=1
        ej=energ(j)-es
        call taylor(ndata,energ,ej,iq,x,y,z)
```

```
            if (iq.eq.1) then
                  faus=fdst(1)*y+fdst(2)*z
            else if (iq.eq.ndata) then
                  faus=fdst(ndata-1)*x+fdst(ndata)*y
            else
                  faus=fdst(iq-1)*x+fdst(iq)*y+fdst(iq+1)*z
            end if
            vaus(j)=qu(j)*faus*eaus(j)*eaus(j)
          end do
          call trapezi(ndata,eaus,vaus,1,ndata,r(ir))
          r(ir)=r(ir)*(dsqrt(2.d0)/(cl*cl))*m**(-1.5d0)
          r(ir)=r(ir)*g(ir)/(2*g(n_cr+1))
        end do

        return
        end
```

subroutine salvarisultati

```
        implicit none
    integer i
        include 'oggetti.fi'
        include 'dati.fi'
        include 'valori.fi'

    open(unit=10,file='coeff.out',status='unknown')
      do i=1,ndata
                  write(10,99) i,energ(i)/e,K(i),U(i),gamma(i),dabs(dx(i))
          end do
    close (unit=10)
    open(unit=10,file='a_funz.out',status='unknown')
      do i=1,ndata
                  write(10,98) energ(i)/e,A(i),DA(i)
          end do
    close (unit=10)

98      format (3g16.6)
99      format (i5,5g16.6)

        end
```

```
      subroutine soluzione_cr

      implicit none
      include 'parametri_cr.fi'
      integer ipath,i,j
  parameter  (ipath=1)

      write(*,*)'Calcolo soluzione'

c     call dlslrg (n_cr,a_cr,n_cr,tn_cr,ipath,nk)
      call dlsarg (n_cr,a_cr,n_cr,tn_cr,ipath,nk)

c     Calcolo flussi di popolazione percentuali

      do i=2,n_cr
       do j=1,n_cr
            fp(i,j)=-1.0d2*((a_cr(i,j)*nk(j))/(a_cr(i,i)*nk(i)))
       end do
      end do

c     Calcolo flusso

      do j=1,n_cr
       s_cr=s_cr+sc_cr(j)*nk(j)
      end do
      fs=s_cr-alfa_cr

      return
      end

  subroutine soluzione

      implicit none
      include 'oggetti.fi'
  integer iparam(6),ipath,i
      double precision rparam(5)
      external dlslxg,dl4lxg

      call dl4lxg(iparam,rparam)
```

```
      ipath=1
      iparam(5)=10000000

      call dlslxg(ndata,nzero,AA,IA,JA,tn,ipath,iparam,rparam,fdst)

c     open (unit=10,file='fdist1.out',status='unknown')
c      do i=1,ndata
c        write(10,10) i,energ(i),fdst(i)
c      end do
c     close (10)

c10   format(i6,2g16.8)

      return
      end
```

subroutine taylor(nqu,x,ener,ie,a,b,c)

```
c     Subroutine che calcola i tre coefficienti A,B,C derivanti dalla approssimazione di
c     Taylor del secondo ordine. Viene utilizzata per ogni funzione f generica
c     (sez d'urto, funzione di distribuzione...)
c     nqu: nr dei dati
c     x: vettore delle x
c     ener: valore di x al quale è richiesta la funzione
c     ie:   indice del punto più vicino (output)
c     a,b,c: i tre coefficienti di uscita

      implicit none
      integer nqu,i,ie,ix
  double precision ener,a,b,c,de,sx,dx,sum
      double precision x(nqu)

      a=0.d0
      b=0.d0
      c=0.d0
      ix=ie
      do i=ix,nqu
       if (ener.le.x(i)) then
            if (i.gt.1) then
         if((x(i)-ener).lt.(ener-x(i-1))) then
```

```
                ie=i
                    else
                     ie=i-1
                    end if
                    goto 5
              else
                    ie=1
              end if
                  goto 5
            end if
          end do
5         continue
          if (ie.eq.1) then
            b=1.d0-(ener-x(ie))/(x(ie+1)-x(ie))
            c=(ener-x(ie))/(x(ie+1)-x(ie))
          else if (ie.eq.nqu) then
            b=1.d0+(ener-x(ie-1))/(x(ie)-x(ie-1))
            a=-(ener-x(ie-1))/(x(ie)-x(ie-1))
          else
            sx=x(ie)-x(ie-1)
            dx=x(ie+1)-x(ie)
            sum=dx+sx
            de=ener-x(ie)
       a=-(dx/(sx*sum))*de+(1.d0/(sx*sum))*de*de
            b=1.d0+(1.d0/sx-1.d0/dx)*de-(1.d0/(dx*sx))*de*de
       c=(sx/(dx*sum))*de+(1.d0/(dx*sum))*de*de
          end if

          return
          end
```

subroutine temp

```
          implicit none
          include 'dati.fi'
          include 'oggetti.fi'
          integer i
     double precision faus(ndatamx)
          include 'valori.fi'
```

```
      Te=0.d0
            do i=1,ndata
              faus(i)=energ(i)*dsqrt(energ(i))*fdst(i)
            end do
      call trapezi(ndata,energ,faus,1,ndata,Te)
      Te=2.d0/3.d0*Te/kboltz

            return
            end

      subroutine ter_ecc(necc,ndecc,qecc,ds,tecc,Iecc)

c           Questa subroutine genera le matrici terecc e Iecc.
c           Iecc: matrice degli indici di colonna dei punti traslati riferiti a be
c           tecc: matrice dei punti traslati ae,be,ce derivanti dall'approssimazione di Taylor
c                       del 2° ordine.
c     ii rappresenta l'indice di eccitazione

            implicit none
      integer i,j,ii,icol,iq,necc
      double precision caus,qu,x,y,z,ds,es,ae,be,ce
            include 'dati.fi'
            include 'oggetti.fi'
            double precision tecc(ndatamx,3*necc),qecc(nqmax,2*necc)
            integer IND(ndatamx),Iecc(ndatamx,necc),ndecc(necc)
            include 'valori.fi'

c           Inizializzazione della matrice

            do i=1,ndatamx
              do j=1,3*necc
                    tecc(i,j)=0.d0
              end do
            end do

            do i=1,ndatamx
              do j=1,necc
                    Iecc(i,j)=0
              end do
            end do
```

```
      do ii=1,necc
       do j=1,ndecc(ii)
        x0(j)=qecc(j,2*ii-1)
        f0(j)=qecc(j,2*ii)
       end do
       j=1
       iq=1
  do i=1,ndata
        es=energ(i)+qecc(1,2*ii-1)
        if (es.gt.enermax*e) then
          goto 10
        else
              call taylor(ndata,energ,es,j,x,y,z)
              Iecc(i,ii)=j
              ae=x
              be=y
              ce=z
              call taylor(ndecc(ii),x0,es,iq,x,y,z)
              if (iq.eq.1) then
                qu=f0(1)*y+f0(2)*z
              else if (iq.eq.ndecc(ii)) then
                qu=f0(ndecc(ii)-1)*x+f0(ndecc(ii))*y
              else
                qu=f0(iq-1)*x+f0(iq)*y+f0(iq+1)*z
              end if
              if (qu.lt.0.d0) then
                write(*,*) 'terecc.f',ii
              else
              end if
              icol=3*ii
          tecc(i,icol-2)=ae*qu*es*ds*2.0/(k(i)*m*m)
          tecc(i,icol-1)=be*qu*es*ds*2.0/(k(i)*m*m)
          tecc(i,icol)=ce*qu*es*ds*2.0/(k(i)*m*m)
        end if
  end do
10       continue
      end do

      return
```

```
      end

      subroutine ter_ion(nion,ndion,qion,ds,eion,tion,Iion)

c     Questa subroutine genera le matrici terion e Iion.
c     Iion: matrice degli indici di colonna dei punti traslati riferiti a be
c     tion: matrice dei punti traslati ae,be,ce derivanti dall'approssimazione di Taylor
c           del 2° ordine.
c     Per ogni grado di ionizzazione nion ci sono 2 colonne relative alla ripartizione della
c     energia dopo l'urto.
c    ii rappresenta l'indice di ionizzazione
c    id rappresenta l'indice di ripartizione dell'energia dopo l'urto:
c    id=2*ii-1          delta
c     id=2*ii                1-delta

      implicit none
      integer i,j,ii,id,icol,iq,nion
      double precision caus,qu,x,y,z,ds(nion),eion(nion)
      double precision ae,be,ce,es
      logical status
      include 'dati.fi'
      include 'oggetti.fi'
      double precision tion(ndatamx,6*nion),qion(nqmax,2*nion)
      integer IND(ndatamx),Iion(ndatamx,2*nion),ndion(nion)
      include 'valori.fi'

c     Inizializzazione della matrice

      do i=1,ndatamx
        do j=1,6*nion
          tion(i,j)=0.d0
        end do
      end do

      do i=1,ndatamx
        do j=1,2*nion
          Iion(i,j)=0.d0
        end do
      end do
```

```
      status=.true.

      do ii=1,nion
       do j=1,ndion(ii)
        x0(j)=qion(j,2*ii-1)
        f0(j)=qion(j,2*ii)
       end do
20     if (status.eq..true.) then
        id=1
       else
            id=0
       end if
       iq=1
       j=1
   do i=1,ndata
        es=energ(i)/delta+eion(ii)
        if (es.gt.enermax*e) then
         goto 10
        else
            call taylor(ndata,energ,es,j,x,y,z)
            Iion(i,ii)=j
            ae=x
            be=y
            ce=z
            call taylor(ndion(ii),x0,es,iq,x,y,z)
            if (iq.eq.1) then
             qu=f0(1)*y+f0(2)*z
            else if (iq.eq.ndion(ii)) then
             qu=f0(ndion(ii)-1)*x+f0(ndion(ii))*y
            else
             qu=f0(iq-1)*x+f0(iq)*y+f0(iq+1)*z
            end if
            if (qu.lt.0.d0) then
             write(*,*) 'terion.f',qu,ii
            else
            end if
            icol=3*(2*ii-id)
       tion(i,icol-2)=ae*qu*es*2.0*ds(ii)/(k(i)*delta*m*m)
       tion(i,icol-1)=be*qu*es*2.0*ds(ii)/(k(i)*delta*m*m)
       tion(i,icol)=ce*qu*es*2.0*ds(ii)/(k(i)*delta*m*m)
```

```
            end if
      end do
10          continue
            if (status.eq..true.) then
              status=.false.
              delta=1.d0-delta
              goto 20
            else
              status=.true.
                    delta=1.d0-delta
            end if
            continue
          end do




          return
          end

    subroutine terminenoto

          implicit none
    integer i,j,g
          include 'dati.fi'
          include 'oggetti.fi'
          include 'valori.fi'

          tn(1)=1.d0
          do i=2,ndatamx
            tn(i)=0.d0
          end do

    return
          end

    subroutine trapezi(nd,en,fa,imin,imax,integral)

          implicit none
          double precision integral,mu
```

```
      integer imin,imax,i,nd
      double precision fa(nd),en(nd)

      mu=0.d0
      integral=0.d0
      do i=imin+1,imax
        mu=(fa(i)+fa(i-1))*(en(i)-en(i-1))/2.d0
        integral=integral+mu
  end do

      return
      end

      subroutine trovaa(iq,ener,aist,d)

c     Questa subroutine calcola il valore di A e della sua derivata in forma numerica

      implicit none
      integer i,iq
  double precision aist,d,ener
      double precision x,y,z,qu
      double precision c1,c2,b0,sez,d1,d2,dsez(1),V(1)
      include 'oggetti.fi'
  double precision vaus(n_ionmax),daus(n_ionmax)
      logical status
      external dsplez
      include 'dati.fi'
      include 'dati_ar.fi'
      include 'valori.fi'

      aist=0.d0
      d=0.d0

c     Urti e-neutro

      if (status.eq..false.) then
        do i=1,ndel_ar
          x0(i)=qel_ar(i,1)
          f0(i)=qel_ar(i,2)
              status=.true.
```

```
            end do
          else
          end if

c         iq=1
      call taylor(ndel_ar,x0,ener,iq,x,y,z)
          if (iq.eq.1) then
            qu=f0(1)*y+f0(2)*z
          else if (iq.eq.ndel_ar) then
            qu=f0(ndel_ar-1)*x+f0(ndel_ar)*y
          else
            qu=f0(iq-1)*x+f0(iq)*y+f0(iq+1)*z
          end if
          if (qu.lt.0.d0) then
            write(*,*) 'MMMMMMMMMMMMMMMMMM'
          else
          end if
      vaus(1)=N_ar(1)*qu

c         Derivata della sdu e-ArI

          V(1)=ener
      call dsplez(ndel_ar,x0,f0,1,1,1,V,dsez)
          daus(1)=N_ar(1)*dsez(1)

c         Urti coulombiani e-ione

      ldebye=dsqrt(eps0*kboltz*Te/(Ne*e**2.d0))

          do i=2,nion_ar

c         Calcolo dei contributi di A

            c1=i*(e**2.d0)/(8*Pi*eps0)
            b0=c1/ener
            c2=1+(ldebye/b0)**2.d0
            vaus(i)=dlog(dsqrt(c2))
        vaus(i)=4*Pi*b0**2.d0*vaus(i)
        vaus(i)=N_ar(i)*vaus(i)
```

```
c          Calcolo della derivata analitica

             d1=-b0**2*dlog(c2)
             d2=(ldebye**2)/c2
             daus(i)=4*Pi*(d1+d2)/ener
       daus(i)=N_ar(i)*daus(i)

          end do

c          Calcolo A(ener) e la sua derivata d(ener)

          do i=1,nion_ar
            aist=aist+vaus(i)
            d=d+daus(i)
          end do

c          Derivata di A in forma numerica

c     DA(1)=(A(2)-A(1))/(ener(2)-ener(1))
c          do i=2,ndata-1
c             denerdx=ener(i+1)-ener(i)
c             denersx=ener(i)-ener(i-1)
c             dener=denerdx+denersx
c             DA(i)=(A(i+1)-A(i))*denersx/(denerdx*dener)+
c     1         (A(i)-A(i-1))*denerdx/(denersx*dener)
c          end do
c          DA(ndata)=(A(ndata)-A(ndata-1))/(ener(ndata)-ener(ndata-1))

      return
          end

      subroutine trovab

          implicit none
          double precision c,gammaee
          include 'dati.fi'
          include 'oggetti.fi'
          include 'valori.fi'

      c=(12.d0*Pi*(eps0*kboltz*Te0)**(3.d0/2.d0))/(dsqrt(Ne)*e**3.d0)
```

```
      gammaee=4.d0*Pi*((e**2.d0)/(4.d0*Pi*eps0*m))**2.d0*dlog(c)
            B=gammaee*Ne

            return
            end

      subroutine udi(fa,ener,ien,aist,d,var,uist)

            implicit none
            integer i,ien
            double precision result3,result4
            double precision dueterzi,ener,aist,d,uist
            parameter (dueterzi=2.d0/3.d0)
            double precision caus1,caus2,caus3,caus4,caus,uA
      double precision vaus1,vaus2,vaus3,vaus4,vaus5,var
            include 'oggetti.fi'
            include 'dati.fi'
            double precision fa(ndatamx)
            include 'dati_ar.fi'
      include 'valori.fi'

            caus=e*campo/m
            caus1=-dueterzi*caus*caus
            caus2=(4.d0/3.d0)*(alfa*e*campo/(m*m))
            caus3=4.d0/(m*m_ar)
            caus4=kboltz*Th*caus3

            uA=ener*aist
            vaus1=caus1*((1.d0/aist)-(ener/(aist*aist))*d)
            vaus2=caus2*ener/aist
            vaus3=-caus3*ener*uA
            vaus4=-caus4*(d*ener*ener+2.d0*uA)

c           Contributo di Fokker-Plank

            if (ien.eq.1) then
              result4=0.d0
            else
              call trapezi(ndata0,energ0,fa,1,ien,result4)
            end if
```

```
      result3=dsqrt(energ0(ien))*var

      vaus5=-B*(result3+result4)

c     Somma totale dei contributi

      uist=vaus1+vaus2+vaus3+vaus4+vaus5

      return
      end

      subroutine write_nk

    implicit none
      include 'parametri_cr.fi'
      integer i,j

c     Stampa delle popolazioni

      open (unit=10,file='n40_1.out',status='unknown')
        write(10,*) fs
        write(10,9) 'Livello [eV]','nk','n/g'
        do i=1,n_cr+1
          write(10,10) elev(i),nk(i),dlog(nk(i)/g(i))
        end do
      close (10)

c     Stampa dei flussi

      open (unit=10,file='fp40_1.out',status='unknown')
        write(10,50) ((fp(i,j),j=1,n_cr),i=2,n_cr)
      close (10)

9     format(a13,2a15)
10    format(3g16.8)
50    format(78e13.4)

      return
      end
```

subroutine writeres

```
      implicit none
      include 'oggetti.fi'
      include 'dati.fi'
      integer i
      double precision maxw
      external maxw
      include 'dati_ar.fi'

      open (unit=10,file='f40_1.out',status='unknown')
       write(10,11) 'Temp. pesanti = ',Th, '°K'
       write(10,11) 'Campo elettrico = ',campo, 'V/m'
       write(10,12) 'Densità = ',N, 'm-3'
       write(10,13) 'Rapporto E/N = ',campo/N,'V*m^2',
     1          '(',campo*1.d21/N,'Td)'
       write(10,11) 'Temperatura elettronica = ',Te, '°K'
       write(10,12) 'Velocità di drift = ',Vd,'m/s'
       write(10,12) 'Freq. di ioniz. tot. = ',ni,'1/s'
       write(10,12) 'Coeff. di ioniz. = ',alfa,'1/m'
       write(10,12) 'alfa/Densità = ',alfa/N,'m^2'
       write(10,12) 'N ArI = ',N_ar(1),'m^-3'
       write(10,12) 'N ArII = ',N_ar(2),'m^-3'
       write(10,12) 'N ArIII = ',N_ar(3),'m^-3'
       write(10,9) 'Punto','Ener[J]','fdst','Maxwell','f/maxw','f1',
     1                                  'f1/f0'
       do i=1,ndata
        write(10,10) i,enereV(i),fdst(i),maxw(energ(i)),
     1  fdst(i)/maxw(energ(i)),funo(i),dabs(funo(i)/fzero(i))
       end do
      close (10)

9     format(a6,5a14,a20)
10    format(i6,6g16.8)
11    format(a27,f7.0,a3)
12    format(a27,g10.3,a5)
13    format(a27,g10.3,a7,1x,a,f6.1,a4)

      return
      end
```

subroutine zero

```
      implicit none
      double precision maxw
      external maxw
      integer i
      include  'oggetti.fi'

      do i=1,ndata
        if ((fdst(i)).eq.(0.d0))  then
     goto 10
    else
         end if
      end do

10    write(*,*)'la funzione si annulla a',enereV(i),'elettronvolt'
      do i=1,ndata
        if ((fdst(i))/maxw(energ(i)).lt.1.d-10)  then
     goto 20
    else
         end if
      end do

20    write(*,*)'la funzione si abbassa a',enereV(i),'elettronvolt'

      return
      end
```

dati.fi

```
      double precision m,e,Th,Te,Te0,Ne,eps0,Pi,kboltz,h,cl,
     1                 costmaxw,ldebye,b,enermax,delta,azero,Ryd,k1,
     1    campo,alfa,N

      common /dati/m,e,Th,Te,Te0,Ne,eps0,Pi,kboltz,h,cl,
     1                 costmaxw,ldebye,b,enermax,delta,azero,Ryd,k1,
     1    campo,alfa,N
```

dati_ar.fi

```
      integer ndel_ar,nion_ar,necc1_ar1,necc1_ar2

      double precision qion_ar(nqmax,2*n_ionmax)
      double precision tion_ar(ndatamx,6*n_ionmax)
      double precision qecc1_ar1(nqmax,2*n_eccmax)
      double precision tecc1_ar1(ndatamx,3*n_eccmax)
      double precision qecc1_ar2(nqmax,2*n_eccmax)
      double precision tecc1_ar2(ndatamx,3*n_eccmax)
      double precision n_ar(n_ionmax),eion_ar(n_ionmax)
      double precision qel_ar(nqmax,2)
      double precision m_ar

      integer ndion_ar(n_ionmax)
      integer Iion_ar(ndatamx,2*n_ionmax)
      integer ndecc1_ar1(n_eccmax)
      integer Iecc1_ar1(ndatamx,n_eccmax)
      integer ndecc1_ar2(n_eccmax)
      integer Iecc1_ar2(ndatamx,n_eccmax)

    common /t_coll_ar/tion_ar,tecc1_ar1,tecc1_ar2
      common /it_coll_ar/Iion_ar,Iecc1_ar1,Iecc1_ar2
      common /q_ar/qel_ar,qion_ar,qecc1_ar1,qecc1_ar2
      common /nd_ar/ndel_ar,ndion_ar,ndecc1_ar1,ndecc1_ar2
      common /nqdata_ar/nion_ar,necc1_ar1,necc1_ar2
      common /dens_ar/n_ar,m_ar
      common /stati_ar/eion_ar

      eion_ar(1)=15.7596d0*e
      eion_ar(2)=27.629d0*e
      eion_ar(3)=40.74*e
      m_ar=39.948*1.66d-27
```

oggetti.fi

```
      integer ndatamx,n_ionmax,nqmax,n_eccmax,ndd
      integer ndata,nzero,ndata0

      parameter (ndatamx=10000,ndd=20)
      parameter (nqmax=80,n_ionmax=3,n_eccmax=20)
```

```
      double precision energ(ndatamx),energ0(ndatamx),
     1                 A(ndatamx),DA(ndatamx),K(ndatamx),parametro
     double precision U(ndatamx),dx(ndatamx),gamma(ndatamx)
     double precision AA(ndd*ndatamx),tn(ndatamx),integrale,nuovoalfa
      double precision fdst(ndatamx),enereV(ndatamx),Vd,ni,Df(ndatamx),
     1  dfzero(ndatamx),fzero(ndatamx),diffrel,conv,convint,
     1  funo(ndatamx),fdst0(ndatamx),convtemp,convN
      double precision x0(nqmax),f0(nqmax)

      integer IA(ndd*ndatamx),JA(ndd*ndatamx),kr,itermax

      common /crsectint/ndata,ndata0
      common /crsectdp/energ,A,DA,enereV,energ0
      common /risultati/K,U,dx,gamma,integrale,Vd,ni,nuovoalfa
      common /mtx_dp/aa,fdst,tn,fzero,fdst0,diffrel,convN,
     1   conv,convint,parametro,convtemp,dfzero,funo,Df
      common /mtx_int/ia,ja,kr,itermax,nzero
```

parametric_cr.fi

```
      integer ntot_cr,n_cr
      parameter (n_cr=78,ntot_cr=(n_cr+1)*(n_cr+1))

      integer g(n_cr+1),l(n_cr+1)
      double precision elev(n_cr+1),nk(n_cr+1)
      double precision tn_cr(n_cr),fo(n_cr,n_cr),fp(n_cr,n_cr)
      double precision a_cr(n_cr,n_cr),fs
      double precision alfa_cr,s_cr,sc_cr(n_cr)

      common /dati_liv/elev,g,l,tn_cr,fo
      common /matrice_cr/a_cr,sc_cr
      common /risultati_cr/nk,fp,fs,alfa_cr,s_cr
```

valori.fi

```
      e=1.60217733d-19
      eps0=8.854187817d-12
      m=9.11e-31
      h=6.626e-34
```

```
      kboltz=1.380658d-23
Pi=3.141592654
      cl=2.998e8

      enereV(1)=1.0d-3
      enermax=1.0d2
parametro=2.0d-2

      delta=0.5d0
      alfa=0.d0

      conv=5.0d-1
      convtemp=1.d-4
      convN=1.0d-3
      itermax=1000

      azero=0.529d-10
      Ryd=13.6d0*e
      k1=Pi*(2.d0*azero*Ryd)**2.d0
```