Clog Doc: ccoeff1.0.tex

C-Coefficient in Clog

Robert Singleton

C-Coefficient in Clog

Robert L Singleton Jr
School of Mathematics
University of Leeds
LS2 9JT

(Dated: 3/7/2020)

Abstract

Physics documentation for the BPS stopping power in the code Clog.

Contents

I.   GENERAL ANALYTIC EXPRESSIONS FOR THE BPS A-COEFFICIENTS

Suppose we have a plasma with various species labeled by an index b at distinct temperatures T_b, number densities n_b, and species masses m_b.[1] Temperature will be measured in energy units, and we denote the inverse temperature by $\beta_b = 1/T_b$. Electrostatic units will be rationalized cgs. The projectile will have mass m_a, charge e_a and energy $E_a = \frac{1}{2} m_a v_a^2$. The BPS $\mathcal{C}^{\ell\ell}$-coefficients take the form

$$C_{ab}^{\ell\ell} = \left(C_{ab,\text{R}}^{\ell\ell <} + C_{ab,\text{S}}^{\ell\ell\,\text{C}} \right) + C_{ab}^{\ell\ell\,\Delta Q} \tag{1.1}$$

with

$$C_{ab,\text{R}}^{\ell\ell <} = \frac{e_a^2}{4\pi} \frac{i}{2\pi} \int_{-1}^{1} \frac{d\cos\theta}{\cos\theta} \frac{\rho_b(v_a \cos\theta)}{\rho_{\text{total}}(v_a \cos\theta)} F(v_a \cos\theta) \ln\left\{ \frac{F(v_a \cos\theta)}{K^2} \right\}, \tag{1.2}$$

$$C_{ab,\text{S}}^{\ell\ell\,\text{C}} = \frac{e_a^2 \kappa_b^2}{4\pi} \left(\frac{\beta_b m_b}{2\pi} \right)^{1/2} v_a \int_0^1 du\, u^{-1/2} \exp\left\{ -\frac{1}{2} \beta_b m_b v_a^2 u \right\}$$
$$\left[-\ln\left(\beta_b \frac{e_a e_b}{4\pi} K \frac{m_b}{m_{ab}} \frac{u}{1-u} \right) - 2\gamma \right] \tag{1.3}$$

$$C_{ab}^{\ell\ell\,\Delta Q} = -\frac{e_a^2 \kappa_b^2}{4\pi} \left(\frac{\beta_b m_b}{2\pi} \right)^{1/2} \int_0^{\infty} dv_{ab} \left\{ \text{Re}\, \psi\,(1 + i\eta_{ab}) - \ln \eta_{ab} \right\}$$
$$\left[\exp\left\{ -\frac{1}{2} \beta_b m_b \,(v_a - v_{ab})^2 \right\} - \exp\left\{ -\frac{1}{2} \beta_b m_b \,(v_a + v_{ab})^2 \right\} \right], \tag{1.4}$$

This regular form is BPS (9.7); the singular form is BPS (9.5); the quantum form is BPS (10.27), where

$$\eta_{ab} = \frac{e_a e_b}{4\pi \hbar v_{ab}} \,. \tag{1.5}$$

The Debye wavenumber K is arbitrary and will typically be chosen as $K = \kappa_e$. The function $F(v)$ takes the form

$$F(v) = -\int_{-\infty}^{\infty} du\, \frac{\rho_{\text{tot}}(u)}{v - u + i\eta} \quad \text{with} \quad \rho_{\text{tot}}(u) = \sum_b \rho_b(u) \tag{1.6}$$

$$\rho_b(v) = \kappa_b^2 \sqrt{\frac{\beta_b m_b}{2\pi}}\, v\, \exp\left\{ -\frac{1}{2} \beta_b m_b\, v^2 \right\}, \tag{1.7}$$

and its relation to the dielectric function is

$$k^2\, \epsilon(\mathbf{k}, \mathbf{k} \cdot \mathbf{v}) = k^2 + F(\hat{\mathbf{k}} \cdot \mathbf{v})\,. \tag{1.8}$$

---

[1] By convention $b = 1$ will be the electron component.

The first term $C_{ab,\mathrm{R}}^{\ell\ell\,<}$ arises from long-distance collective effects from the dielectric function, and it involves *all* plasma species (even species c different from a and b). This is the term I call non-separable, meaning that it cannot be written as a sum of individual plasma components involving only a single species. The second term $C_{ab,\mathrm{S}}^{\ell\ell\,\mathrm{C}}$ arises from short-distance two-body classical scattering, and the third term $C_{ab}^{\ell\ell\,\mathrm{QM}}$ is the two-body quantum scattering correction to all orders in the quantum parameters $\bar{\eta}_{ab}$. Three body and higher effects are contained in our systematic error term, the next-to-next-to-leading order term proportional to g^3. In a strongly coupled plasma these higher order effects dominate, but in a weakly coupled plasma they are negligible.

II. THE MAIN DRIVER

I will return the C-coefficients in three forms:

i. `bps_ccoeff_ab_mass`: For a given pair of indices p and b (the projectile p will often be denoted by species index a), this routine returns the individual component $C_{ab}^{\ell\ell}(E)$ for a given energy E. The quantum parameter η can be arbitrary. This routine is used to construct the entries in the next two subroutines.

ii. `bps_ccoeff_ab_matrix`: Returns the complete matrix of coefficients $C_{ab}^{\ell\ell}(E)$.

iii. `bps_ccoeff_ei_mass`: This routine returns the sum over the ions $C_{p\mathrm{I}}^{\ell\ell} = \sum_i C_{pi}^{\ell\ell}$ for a given projectile p. It also returns the coulomb logarithm.

A. The Driver Routine: bps_ccoeff_ab_mass

This subroutine returns the matrix of values $C_{ab}^{\ell\ell}(E)$ for a given energy E. The driving routine that calls and assembles the singular, regular, and quantum pieces.

ccoeff.f90:bps_ccoeff_ab_mass

```
!
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! main driver for C-coefficient for general quantum and electron-mass regimes
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!
      SUBROUTINE bps_ccoeff_ab_mass(nni, ep, mp, zp, ia, ib, betab, zb, mb, nb, &
          a_ab, a_ab_sing, a_ab_reg, a_ab_qm)
      USE physvars
      USE mathvars
        IMPLICIT NONE                                           ! Plasma:
        INTEGER,                            INTENT(IN)  :: nni   !  number of ions
        REAL,                               INTENT(IN)  :: ep    !  energy input [ke
        REAL,                               INTENT(IN)  :: mp    !  mass [keV]
        REAL,                               INTENT(IN)  :: zp    !  charge
        INTEGER,                            INTENT(IN)  :: ia    !
        INTEGER,                            INTENT(IN)  :: ib    !
        REAL,    DIMENSION(1:nni+1),        INTENT(IN)  :: betab !  temp array [1/ke
        REAL,    DIMENSION(1:nni+1),        INTENT(IN)  :: mb    !  mass array [keV]
        REAL,    DIMENSION(1:nni+1),        INTENT(IN)  :: nb    !  density [1/cc]
        REAL,    DIMENSION(1:nni+1),        INTENT(IN)  :: zb    !  charge array
                                                               !
                                                               ! C-coeffs [MeV/mic
        REAL,                               INTENT(OUT) :: a_ab
        REAL,                               INTENT(OUT) :: a_ab_sing
        REAL,                               INTENT(OUT) :: a_ab_reg
        REAL,                               INTENT(OUT) :: a_ab_qm

        REAL,    DIMENSION(1:nni+1)  :: mpb, mbpb, kb2, ab
        REAL                         :: vp, zp2, k, k2, kd, kd2, a, b, eta
        REAL                         :: ac_r, ac_s, aq, c1, c2

        REAL, PARAMETER              :: EPS_SMALL_E=2.E-4
        REAL, PARAMETER              :: EPS_SMALL_E_SING=2.E-4
        REAL, PARAMETER              :: EPS_SMALL_E_REG=2.E-4
!
! initialize components of C-coefficients
!
        kb2=8*PI*A0CM*BEKEV*zb*zb*nb*betab
        kd2 = SUM(kb2)                 ! [1/cm^2]
```

```
        kd   = SQRT(kd2)                ! [1/cm]
        k2   = kb2(1)                   ! [1/cm^2]
        k    = SQRT(k2)                 ! [1/cm]    k = k_e
!
! Loop over charged plasma species
!
        mpb = mp*mb/(mp+mb)             ! [keV]
        mbpb= mb/mpb                    ! [dimensionless]
        vp =CC*SQRT(2*ep/mp)            ! [cm/s]
        zp2=zp**2                       ! [dimensionless]
                                        ! ab=(1/2) betab(ib)*mbc2(ib)*vp2/CC2
        ab  =0.5*betab*mb*vp*vp/CC2     ! [dimensionless]
        IF (zb(ib) .NE. 0.) THEN
        a  =ab(ib)
        b  =-Log(2*betab(ib)*BEKEV*ABS(zp*zb(ib))*k*A0CM*mbpb(ib) )-2*GAMMA+2
        eta=ABS(zp*zb(ib))*2.1870E8/vp ! defined with projectile velocity vp
        c1=2*zp2*BEKEV*kb2(ib)*A0CM     ! [keV/cm] c1 = e_p^2 kappa_b^2/(4 Pi)
        c1=c1*1.E-7                     ! [MeV/micron]
        c2=SQRT(a/PI)                   ! [dimensionless]
                                        ! c2=SQRT(betab(ib)*mb(ib)/TWOPI)*vp/CC
!
! C_{ab}-classical-singular
!
        CALL c_sing_mass(a,b,ac_s)
        a_ab_sing=c1*c2*ac_s
!
! C_{ab}-classical-regular
!
        CALL c_reg_mass(nni,ia,ib,vp,k2,kb2,betab,mb,ac_r)
        a_ab_reg=c1*ac_r
!
! C_{ab}-quantum
!
        CALL c_quantum_mass(ia,ib,a,eta,aq) ! eta = dimensionless quantum param.
        a_ab_qm=c1*c2*aq
!
! C_{ab}-total
!
        a_ab=a_ab_sing + a_ab_reg + a_ab_qm
        ENDIF
      END SUBROUTINE bps_ccoeff_ab_mass
```

ccoeff.f90:bps_ccoeff_ab_matrix

```
!
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! Assembles the matrix C_{ab} of the C-coefficients.
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!
      SUBROUTINE bps_ccoeff_ab_matrix(nni, ep, betab, zb, mb, nb,     &
        c_ab, c_ab_sing, c_ab_reg, c_ab_qm, c_tot, c_i, c_e, cc_tot, &
        cc_i, cc_e, cq_tot, cq_i, cq_e, cc_s_i, cc_s_e, cc_r_i, cc_r_e)
      USE physvars
      USE mathvars
        IMPLICIT NONE                                      ! Plasma:
        INTEGER,                          INTENT(IN)  :: nni    !  number of ions
        REAL,                             INTENT(IN)  :: ep     !  energy
        REAL,    DIMENSION(1:nni+1),      INTENT(IN)  :: betab  !  temp array [1/ke
        REAL,    DIMENSION(1:nni+1),      INTENT(IN)  :: zb     !  charge array
        REAL,    DIMENSION(1:nni+1),      INTENT(IN)  :: mb     !  mass array [keV]
        REAL,    DIMENSION(1:nni+1),      INTENT(IN)  :: nb     !  density [1/cc]
                                                           !
                                                           ! C-coeffs [MeV/mic
        REAL,    DIMENSION(1:nni+1,1:nni+1),INTENT(OUT) :: c_ab
        REAL,    DIMENSION(1:nni+1,1:nni+1),INTENT(OUT) :: c_ab_sing
```

```fortran
      REAL,     DIMENSION(1:nni+1,1:nni+1),INTENT(OUT) :: c_ab_reg
      REAL,     DIMENSION(1:nni+1,1:nni+1),INTENT(OUT) :: c_ab_qm
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: c_tot
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: c_i
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: c_e
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cc_tot
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cc_i
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cc_e
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cq_tot
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cq_i
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cq_e
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cc_s_i
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cc_s_e
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cc_r_i
      REAL,     DIMENSION(1:nni+1),        INTENT(OUT) :: cc_r_e

      REAL    :: cab, cab_sing, cab_reg, cab_qm
      REAL    :: mp, zp
      INTEGER :: ia, ib

      c_i   = 0
      cc_s_i= 0
      cc_r_i= 0
      cc_i  = 0
      cq_i  = 0
      DO ia=1,nni+1
        mp=mb(ia)
        zp=zb(ia)
        DO ib=1,nni+1
          CALL bps_ccoeff_ab_mass(nni, ep, mp, zp, ia, ib, betab, zb, mb, nb, &
          cab, cab_sing, cab_reg, cab_qm) !*! change to bps_acoeff_ab_mass
          c_ab(ia,ib)      =cab
          c_ab_sing(ia,ib) =cab_sing
          c_ab_reg(ia,ib)  =cab_reg
          c_ab_qm(ia,ib)   =cab_qm
          IF (ib == 1) THEN
             c_e(ia)    = cab
             cc_s_e(ia) = cab_sing
             cc_r_e(ia) = cab_reg
             cc_e(ia)   = cab_sing + cab_reg
             cq_e(ia)   = cab_qm
          ELSE
             c_i(ia)    = c_i(ia)    + cab
             cc_s_i(ia) = cc_s_i(ia) + cab_sing
             cc_r_i(ia) = cc_r_i(ia) + cab_reg
             cc_i(ia)   = cc_i(ia)   + cab_sing + cab_reg
             cq_i(ia)   = cq_i(ia)   + cab_qm
          ENDIF
        ENDDO
        c_tot(ia) = c_e(ia)  + c_i(ia)
        cc_tot(ia)= cc_e(ia) + cc_i(ia)
        cq_tot(ia)= cq_e(ia) + cq_i(ia)
      ENDDO
    END SUBROUTINE bps_ccoeff_ab_matrix
```

ccoeff.f90:bps_ccoeff_ei_mass

```fortran
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! Returns C_{p I} = \sum_i C_{p i} for backward compatibility
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!
      SUBROUTINE bps_ccoeff_ei_mass(nni, ep, zp, mp, betab, zb, mb, nb, &
          c_tot, c_i, c_e, cc_tot, cc_i, cc_e, cq_tot, cq_i, cq_e, &
          cc_s_i, cc_s_e, cc_r_i, cc_r_e)
      USE physvars
      USE mathvars
      USE controlvars
        IMPLICIT NONE                                ! Plasma:
        INTEGER,                       INTENT(IN)  :: nni    !  number of ions
        REAL,    DIMENSION(1:nni+1),   INTENT(IN)  :: betab  !  temp array [1/keV]
        REAL,    DIMENSION(1:nni+1),   INTENT(IN)  :: mb     !  mass array [keV]
        REAL,    DIMENSION(1:nni+1),   INTENT(IN)  :: nb     !  density [1/cc]
        REAL,    DIMENSION(1:nni+1),   INTENT(IN)  :: zb     !  charge array
                                                     !
                                                     ! Projectile
        REAL,                          INTENT(IN)  :: ep     !  projectile energy [keV]
        REAL,                          INTENT(IN)  :: mp     !  projectile mass   [keV]
        REAL,                          INTENT(IN)  :: zp     !  projectile charge
                                                     !
                                                     ! C-coeffs [MeV/micron]
        REAL,                          INTENT(OUT) :: c_tot  !  electron + ion
        REAL,                          INTENT(OUT) :: c_i    !  ion contribution
        REAL,                          INTENT(OUT) :: c_e    !  electron contribution
        REAL,                          INTENT(OUT) :: cc_tot !  classical
        REAL,                          INTENT(OUT) :: cc_i   !  classical
        REAL,                          INTENT(OUT) :: cc_e   !  classical
        REAL,                          INTENT(OUT) :: cq_tot !  quantum
        REAL,                          INTENT(OUT) :: cq_i   !  quantum
        REAL,                          INTENT(OUT) :: cq_e   !  quantum
        REAL,                          INTENT(OUT) :: cc_s_i
        REAL,                          INTENT(OUT) :: cc_s_e
        REAL,                          INTENT(OUT) :: cc_r_i
        REAL,                          INTENT(OUT) :: cc_r_e

        REAL     :: cdum, cc_s, cc_r, cq
        INTEGER  :: ia, ib, nnb
!
! initialize components of C-coefficients
!
      c_tot =0  ! electron + ion
      c_i   =0  ! ion contribution
      c_e   =0  ! electron contribution
      cc_tot=0  ! classical total
      cc_e  =0  ! classical electron
      cc_i  =0  ! classical ion
      cq_tot=0  ! quantum total
      cq_e  =0  ! quantum electron
      cq_i  =0  ! quantum ion
      cc_s_i=0
      cc_s_e=0
      cc_r_i=0
      cc_r_e=0

      NNB = nni+1                 ! number of ions + electrons
      ia=1
      DO ib=1,nni+1
      IF (zb(ib) .NE. 0.) THEN
          CALL bps_ccoeff_ab_mass(nni, ep, mp, zp, ia, ib, betab, zb, mb, nb, &
          cdum, cc_s, cc_r, cq)
          CALL x_collect(ib, NNB, cc_s, cc_r, cq,         &
          c_tot, c_i, c_e, cc_tot, cc_i, cc_e, cq_tot,  &
```

```
        cq_i, cq_e, cc_s_i, cc_s_e, cc_r_i, cc_r_e)
      ENDIF
      ENDDO
    END SUBROUTINE bps_ccoeff_ei_mass
```

B.   The Regular Contribution: c_reg_mass

The long-distance regular contribution can be expressed as

$$
\begin{aligned}
C_{ab,\text{R}}^{\ell\ell<} &= \frac{e_a^2}{4\pi}\frac{i}{2\pi}\int_{-1}^{1}\frac{du}{u}\frac{\rho_b(v_a u)}{\rho_{\text{total}}(v_a u)}\,F(v_a u)\ln\left\{\frac{F(v_a u)}{K^2}\right\} \tag{2.1}
\end{aligned}
$$

$$
= \frac{e_a^2}{4\pi}\frac{i}{2\pi}\int_{0}^{1}\frac{du}{u}\frac{\rho_b(v_a u)}{\rho_{\text{total}}(v_a u)}\left[F(v_a u)\ln\left\{\frac{F(v_a u)}{K^2}\right\} - F^*(v_a u)\ln\left\{\frac{F^*(v_a u)}{K^2}\right\}\right] \tag{2.2}
$$

$$
= -\frac{e_a^2}{4\pi}\frac{1}{2\pi}\int_{0}^{1}\frac{du}{u}\frac{\rho_b(v_a u)}{\rho_{\text{total}}(v_a u)}\,H(v_a u)\;, \tag{2.3}
$$

where we have defined

$$
H(v) \equiv -i\left[F(v)\ln\left\{\frac{F(v)}{K^2}\right\} - F^*(v)\ln\left\{\frac{F^*(v)}{K^2}\right\}\right] = 2\left[F_{\text{Re}}\arg\{F\} + F_{\text{Im}}\ln\left\{\frac{|F|}{K^2}\right\}\right]\;. \tag{2.4}
$$

We shall factor out a dimensionfull wavenumber K and define dimensionless quantities $\mathbb{F}(v)$ and $\mathbb{H}(v)$ through

$$
F(v) = K^2\,\mathbb{F}(v) \quad\text{and}\quad H(v) = K^2\,\mathbb{H}(v)\;. \tag{2.5}
$$

Defining the parameters

$$
a_c \equiv \left(\frac{\beta_c m_c}{2}\right)^{1/2} \tag{2.6}
$$

$$
\bar{\kappa}_c^2 \equiv \frac{\kappa_c^2}{K^2} \tag{2.7}
$$

gives the real and imaginary parts of $\mathbb{F}$,

$$
\mathbb{F}_{\text{Re}}(\{a_c v\}, \{\bar{\kappa}_c\}) = \sum_c \bar{\kappa}_c^2\left(1 - 2a_c\,v\,\text{daw}\{a_c\,v\}\right) \tag{2.8}
$$

$$
\mathbb{F}_{\text{Im}}(\{a_c v\}, \{\bar{\kappa}_c\}) = \sqrt{\pi}\sum_c \bar{\kappa}_c^2\,a_c v\,e^{-a_c^2 v^2}\;. \tag{2.9}
$$

The ratio of weighting factors can be written in terms of a function $\mathbb{R}_{ab}$ defined by

$$\frac{\rho_b(v_a u)}{\rho_{\text{total}}(v_a u)} \cdot H(v_a u) \;=\; K^2 \, \frac{\rho_b(v_a u)}{\rho_{\text{total}}(v_a u)} \cdot \mathbb{H}(v_a u) \tag{2.10}$$

$$=\; K^2 \, \frac{\kappa_b^2 \, (\beta_b m_b/2\pi)^{1/2} \, v_a u \, e^{-\frac{1}{2}\,\beta_b m_b v_a^2 \, u^2}}{\sum_c \kappa_c^2 \, (\beta_c m_c/2\pi)^{1/2} \, v_a u \, e^{-\frac{1}{2}\,\beta_c m_c v_a^2 \, u^2}} \cdot \mathbb{H}(v_a u) \tag{2.11}$$

$$=\; \kappa_b^2 \cdot \underbrace{\left[\sum_c \frac{\kappa_c^2}{K^2}\left(\frac{\beta_c m_c}{\beta_b m_b}\right)^{1/2} e^{\frac{1}{2}\,(\beta_b m_b - \beta_c m_c) v_a^2 \, u^2}\right]^{-1}}_{\mathbb{R}_{ab}(v_a u)} \cdot \mathbb{H}(v_a u) \tag{2.12}$$

$$=\; \kappa_b^2 \, \mathbb{R}_{ab}(v_a u) \, \mathbb{H}(v_a u) \;. \tag{2.13}$$

We can now express the regular piece as

$$C_{ab,\text{R}}^{\ell\ell\,\text{C}} \;=\; \underbrace{\left[\frac{e_a^2 \, \kappa_b^2}{4\pi}\right]}_{c_{ab,1}} \cdot \mathsf{C}_{ab\,\text{R}}(v_a, \{a_c\}, \{\bar{\kappa}_c\}) \tag{2.14}$$

$$\mathsf{C}_{ab\,\text{R}}(v_a, \{a_c\}, \{\bar{\kappa}_c\}) \;=\; -\int_0^1 \frac{du}{u} \underbrace{\mathbb{R}_{ab}(\{a_c v_a u\}) \, \mathbb{H}(\{a_c \, v_a u\}, \{\bar{\kappa}_c\})}_{\text{d\_cab\_reg}} \;. \tag{2.15}$$

ccoeff.f90: d_cab_reg

```
!
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! regular contribution for non-zero electron mass
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!
      SUBROUTINE c_reg_mass(nni, ia, ib, vp, k2, kb2, betab, mb, cc_r)
      USE physvars
       IMPLICIT NONE
       INTEGER,                           INTENT(IN)  :: nni
       INTEGER,                           INTENT(IN)  :: ia
       INTEGER,                           INTENT(IN)  :: ib
       REAL,                              INTENT(IN)  :: vp
       REAL,                              INTENT(IN)  :: k2
       REAL,                              INTENT(IN)  :: kb2
       REAL,    DIMENSION(1:nni+1),  INTENT(IN)  :: betab
       REAL,    DIMENSION(1:nni+1),  INTENT(IN)  :: mb
       REAL,                              INTENT(OUT) :: cc_r
       REAL,    DIMENSION(1:nni+1)   :: ab
!      INTEGER, PARAMETER :: NR=10 ! integration regions: must be even
       INTEGER, PARAMETER :: NR=100 ! integration regions: must be even
       REAL,    PARAMETER :: UPM=0.7745966692E0 ! parameters for Gaussian Quad
       REAL,    PARAMETER :: W13=0.5555555556E0, W2=0.8888888889E0
       REAL               :: u0, u1, du, u, um, d_cab_reg
       INTEGER            :: iu
       ab=SQRT(0.5*betab*mb)*vp/CC
       cc_r=0
       u0=0.0
       u1=1.
!      u1=MIN(1.,5/(ab(ib)**2)) ! support can lie << 1
       du=(u1-u0)/NR
       u=u0-du
       DO iu=1,NR,2 ! Gaussian quadrature
          u=u+2.*du
```

```fortran
               cc_r=cc_r+W2*d_cab_reg(u,vp,ia,ib,nni,k2,kb2,betab,mb)
               um=u-du*UPM
               cc_r=cc_r+W13*d_cab_reg(um,vp,ia,ib,nni,k2,kb2,betab,mb)
               um=u+du*UPM
               cc_r=cc_r+W13*d_cab_reg(um,vp,ia,ib,nni,k2,kb2,betab,mb)
            ENDDO
            cc_r=cc_r*du
         END SUBROUTINE c_reg_mass

         FUNCTION d_cab_reg(u, vp, ia, ib, nni, k2, kb2, betab, mb)
         USE mathvars
         USE physvars
           IMPLICIT NONE
           REAL,                            INTENT(IN)  :: u      ! [dimensionless]
           REAL,                            INTENT(IN)  :: vp     ! Projectile velocity [cm/
           INTEGER,                         INTENT(IN)  :: ia     ! Species number
           INTEGER,                         INTENT(IN)  :: ib     ! Species number
           INTEGER,                         INTENT(IN)  :: nni    ! Number of ion species
           REAL,                            INTENT(IN)  :: k2     ! Wave-number squared [1/c
           REAL,    DIMENSION(1:nni+1),     INTENT(IN)  :: kb2    ! Debye wavenumber squared
           REAL,    DIMENSION(1:nni+1),     INTENT(IN)  :: betab  ! Temperature array [1/keV]
           REAL,    DIMENSION(1:nni+1),     INTENT(IN)  :: mb     ! Mass array [keV]
           REAL                                         :: d_cab_reg! [dimensionless]
           REAL,    DIMENSION(1:nni+1) :: kbar2b, ab, ab2
           REAL                       :: fr, fi, fabs, farg, h, r_ib
           REAL                       :: kcb, bm_ic, bm_ib, a_ic, a_ib, ex, au
           INTEGER                    :: ic
           ab=SQRT(0.5*betab*mb)*vp/CC
           ab2=ab*ab
           kbar2b=kb2/k2
           CALL frfi(u,nni,kbar2b,ab,fr,fi,fabs,farg)
           h=2*(fr*farg + fi*LOG(fabs))*u
           r_ib=0
           bm_ib=betab(ib)*mb(ib)
           a_ib =ab(ib)*ab(ib)
           DO ic=1,nni+1
              kcb=kb2(ic)/k2
              bm_ic=betab(ic)*mb(ic)
              a_ic =ab(ic)*ab(ic)
              IF (ic == ib) THEN
                 ex=1.
              ELSE
                 au=(a_ic-a_ib)*u
                 ex=EXP(-au)
              ENDIF
              r_ib=r_ib + kcb*SQRT(bm_ic/bm_ib)*ex
           ENDDO
           r_ib=1./r_ib
!
           d_cab_reg=-r_ib*h/TWOPI
         END FUNCTION d_cab_reg
```

C. The Singular Contribution: c_sing

The singular contribution,

$$
\mathcal{A}_{b,s}^{\mathrm{C}} = \left[\frac{e_p^2\,\kappa_b^2}{4\pi} \left(\frac{\beta_b m_b}{2\pi} \right)^{1/2} v_p \right] \int_0^1 du\, u^{-1/2} e^{-\frac{1}{2}\,\beta_b m_b v_p^2\,u} \left[-\ln\left\{ \frac{\beta_b e_b e_p}{4\pi}\,K\,\frac{m_b}{m_{pb}}\,\frac{u}{1-u} \right\} - 2\gamma \right],
$$

(2.16)

is quite easy to code. The integral can be broke into the pieces

$$\int_0^1 du\, u^{-1/2} e^{-\frac{1}{2}\beta_b m_b v_p^2 u} \left[\ln\left\{ \frac{u}{1-u} \right\} - \ln\left\{ \frac{\beta_b e_b e_p}{4\pi} K \frac{m_b}{m_{pb}} \right\} - 2\gamma \right], \qquad (2.17)$$

which motivates the definition

$$C_{b,\text{S}}^{\ell\ell\,\text{C}} = c_{b,1}\, c_{b,2} \cdot \mathsf{C}_\text{S}(a_{pb}, b_{pb}) \qquad (2.18)$$

$$\mathsf{C}_\text{S}(a, b) = \int_0^1 du\, u^{-1/2} e^{-a\,u} \left[-\ln\left\{ \frac{u}{1-u} \right\} + b \right] \qquad (2.19)$$

$$a_{pb} = \frac{1}{2}\beta_b m_b v_p^2 \qquad \text{and} \qquad b_{pb} = -\ln\left\{ \frac{\beta_b e_b e_p}{4\pi} K \frac{m_b}{m_{pb}} \right\} - 2\gamma \qquad (2.20)$$

$$c_{b,1} = \frac{e_p^2\, \kappa_b^2}{4\pi} \qquad c_{b,2} = \left(\frac{\beta_b m_b}{2\pi} \right)^{1/2} v_p\,. \qquad (2.21)$$

The term involving b can be integrated exactly, but we will use Gaussian quadrature for both pieces.

acoeff.f90:

**
```
    FUNCTION dab_sing(u, a, b)
      IMPLICIT NONE   ! a=(1/2)*beta*mpc2*vp^2/C^2
      REAL,        INTENT(IN)   :: u ! [dimensionless]
      REAL,        INTENT(IN)   :: a ! [dimensionless]
      REAL,        INTENT(IN)   :: b ! [dimensionless]
      REAL                      :: dab_sing ! [dimensionless]
      dab_sing=SQRT(-u)*EXP(-a*u)*(-LOG(u/(1-u)) + b)
    END FUNCTION dab_sing
```

The numerical integration is performed by Gaussian quadrature:

acoeff.f90:

**
```
      SUBROUTINE a_sing(a, b, ac_s)
        REAL,    INTENT(IN)  :: a
        REAL,    INTENT(IN)  :: b
        REAL,    INTENT(OUT) :: ac_s
        REAL                 :: u0, u1, du, u, um
        INTEGER, PARAMETER :: NS=1000              ! integration regions singular: must
        REAL,    PARAMETER :: UPM=0.7745966692E0 ! parameters for Gaussian Quad
        REAL,    PARAMETER :: W13=0.5555555556E0, W2=0.8888888889E0
          ac_s=0
          u0=0
          u1=1
          du=(u1-u0)/NS
          u=u0-du
          DO iu=1,NS,2 ! Gaussian quadrature
             u=u+2.E0*du
             ac_s=ac_s+W2*dab_sing(u,a,b)
             um=u-du*UPM
             ac_s=ac_s+W13*dab_sing(um,a,b)
             um=u+du*UPM
             ac_s=ac_s+W13*dab_sing(um,a,b)
          ENDDO
          ac_s=ac_s*du
      END SUBROUTINE a_sing
```

D. The Quantum Correction: c_quantum

For the quantum term we make the change of variables $v_{pb} = v_p\, u$ so that

$$C_b^{\ell\ell\,\mathrm{QM}} = -\frac{e_p^2\,\kappa_b^2}{4\pi}\left(\frac{\beta_b m_b}{2\pi}\right)^{1/2} v_p \int_0^\infty du\left[\mathrm{Re}\,\psi\left\{1 + i\,\frac{\bar\eta_{pb}}{u}\right\} - \ln\left\{\frac{\bar\eta_{pb}}{u}\right\}\right]$$
$$\left[e^{-\frac{1}{2}\beta_b m_b v_p^2 (u-1)^2} - e^{-\frac{1}{2}\beta_b m_b v_p^2 (u+1)^2}\right]. \qquad (2.22)$$

The quantum function we need to code is therefore

$$C_b^{\ell\ell\,\mathrm{QM}} = \underbrace{\left[\frac{e_p^2\,\kappa_b^2}{4\pi}\left(\frac{\beta_b m_b}{2\pi}\right)^{1/2} v_p\right]}_{c_{b,1}\cdot c_{b,2}}\cdot \mathsf{C}_1^{\mathrm{QM}}(a_{pb}, \tilde\eta_{pb})\ , \qquad (2.23)$$

where the arguments of the function are defined by

$$a_{pb} = \frac{1}{2}\,\beta_b m_b\,v_p^2 \qquad (2.24)$$

$$\tilde\eta_{pb} = \frac{e_p e_b}{4\pi\hbar v_p} = |Z_p Z_b|\,\frac{e^2}{8\pi a_0}\frac{2a_0}{\hbar}\frac{1}{v_p} = |Z_p Z_b|\ \cdot 13.606\,\mathrm{eV}\ \cdot\frac{2\cdot 5.29\times 10^{-9}\,\mathrm{cm}}{6.5821\times 10^{-16}\,\mathrm{eV\,s}}\frac{1}{v_p}$$

$$= 2.1870\times 10^8\,\frac{|Z_p Z_b|}{v_p\cdot(\mathrm{cm/s})^{-1}}\ , \qquad (2.25)$$

and the function itself takes the form

$$\mathsf{C}_1^{\mathrm{QM}}(a, \eta) = -\int_0^\infty du\left[\mathrm{Re}\,\psi\left\{1 + i\,\frac{\eta}{u}\right\} - \ln\left\{\frac{\eta}{u}\right\}\right]\frac{1}{2a\,u}\left[e^{-a(u-1)^2} - e^{-a(u+1)^2}\right]. \quad (2.26)$$

acoeff.f90:

```
**
      FUNCTION daq(u, a, eta)
      USE physvars
       IMPLICIT NONE
       REAL,                        INTENT(IN)  :: u      ! [dimensionless]
       REAL,                        INTENT(IN)  :: a      ! [dimensionless]
       REAL,                        INTENT(IN)  :: eta    ! [dimensionless]
       REAL                                     :: daq  ! [dimensionless]
       REAL, PARAMETER :: AMAX=25.
       REAL            :: repsi, au, eu, au2, ap, am, psilog, ch, sh
       eu=eta/u
       psilog=repsi(eu) - LOG(eu)
       au =2*a*u
       au2=a*u*u
       IF (a <= AMAX) THEN
          ch =EXP(-au2)*COSH(au)
          sh =EXP(-au2)*SINH(au)
       ELSE
          ap = au-au2-a
          am =-au-au2-a
          ch =0.5*(EXP(ap)+EXP(am))
          sh =0.5*(EXP(ap)-EXP(am))
       ENDIF
```

```fortran
      daq=-psilog*2*(ch - sh/au)/au
END FUNCTION daq


SUBROUTINE a_quantum(ib, a, eta, aq)
   IMPLICIT NONE
   INTEGER, INTENT(IN)  :: ib     ! species index
   REAL,    INTENT(IN)  :: a      ! [dimensionless] (1/2) betab mb vp^2
   REAL,    INTENT(IN)  :: eta    ! [dimensionless] ep eb/4pi hbar vp
   REAL,    INTENT(OUT) :: aq
   REAL              :: u0, u1, du, u, um
   INTEGER, PARAMETER :: NQ=1000             ! integration regions quantum : must
   REAL,    PARAMETER :: UPM=0.7745966692E0 ! parameters for Gaussian Quad
   REAL,    PARAMETER :: W13=0.5555555556E0, W2=0.8888888889E0
   REAL    :: daq
   INTEGER :: iu
   aq=0
   u0=0.
   aq=0
   IF (ib == 1) THEN
       u0=0
       u1=4./SQRT(a)
   ELSE
       u0=1-10./SQRT(a)
       u0=MAX(0.,u0)
       u1=1+10./SQRT(a)
   ENDIF
   du=(u1-u0)/NQ
   u=u0-du
   DO iu=1,NQ,2 ! Gaussian quadrature
       u=u+2.E0*du
       aq=aq+W2*daq(u,a,eta)
       um=u-du*UPM
       aq=aq+W13*daq(um,a,eta)
       um=u+du*UPM
       aq=aq+W13*daq(um,a,eta)
   ENDDO
   aq=aq*du
END SUBROUTINE a_quantum
```

Appendix A: Calculating the Real and Imaginary Parts of F

We can write the dielectric function (1.6) as a sum over plasma components,

$$F(v) = \sum_b F_b(v) \, , \tag{A1}$$

where we express the contribution from plasma species b as

$$F_b(v) = - \int_{-\infty}^{\infty} du \, \frac{\rho_b(v)}{v - u + i\eta} \tag{A2}$$

$$\rho_b(v) = \kappa_b^2 \sqrt{\frac{\beta_b m_b}{2\pi}} \, v \, \exp\left\{ -\frac{1}{2} \beta_b m_b \, v^2 \right\} \, . \tag{A3}$$

We will often decompose F into its contribution from electrons and ions and write

$$F(v) = F_e(v) + F_{\text{I}}(v) \, . \tag{A4}$$

Note the reflection property

$$F_b(-v) = F_b^*(v) \, , \tag{A5}$$

which means that the real part of $F_b(v)$ is even in v and the imaginary part is odd. For numerical work it is best to use the explicit real and imaginary parts of F, which can be written

$$F_{\text{Re}}(v) = \sum_b \kappa_b^2 \left[1 - 2\sqrt{\frac{\beta_b m_b}{2}} \, v \, \text{daw}\left\{ \sqrt{\frac{\beta_b m_b}{2}} \, v \right\} \right] \tag{A6}$$

$$F_{\text{Im}}(v) = \sqrt{\pi} \sum_b \kappa_b^2 \sqrt{\frac{\beta_b m_b}{2}} \, v \, \exp\left\{ -\frac{\beta_b m_b}{2} \, v^2 \right\} = \pi \rho_{\text{tot}}(v) \, , \tag{A7}$$

where the Dawson integral is defined by

$$\text{daw}(x) = \int_0^x dy \, e^{y^2 - x^2} = \frac{\sqrt{\pi}}{2} \, e^{-x^2} \text{erfi}(x) \, . \tag{A8}$$

The limits of small and large arguments of the Dawson function are

$$\text{daw}(x) = x + \frac{2x^3}{3} + \frac{4x^5}{15} + \mathcal{O}(x^7) \tag{A9}$$

$$\text{daw}(x) = \frac{1}{2x} + \frac{1}{4x^3} + \frac{3}{8x^5} + \mathcal{O}(x^{-7}) \, . \tag{A10}$$

The functions $F_b(v)$ have units of wave-number-squared $[1/L^2]$ and their argument v has units of velocity. We can express the functions $F_b(v)$ in terms of a single dimensionless function $\mathbb{F}(x)$ as follows:

$$F_b(v) = \kappa_b^2 \, \mathbb{F}\left(\sqrt{\frac{\beta_b m_b}{2}} \, v \right) \tag{A11}$$

with

$$\mathbb{F}(x) \;=\; \int_{-\infty}^{\infty} dy \, \frac{\bar{\rho}(y)}{y - x - i\eta} \tag{A12}$$

$$\bar{\rho}(y) \;=\; \frac{1}{\sqrt{\pi}} \, y \, e^{-y^2} \;. \tag{A13}$$

Relation (A11) holds because $\rho_b(u) = \kappa_b^2 \, \bar{\rho}(y)$ for $u = (2/\beta_b m_b)^{1/2} \, y$. The reflection property (A5) becomes

$$\mathbb{F}(-x) = \mathbb{F}^*(x) \;, \tag{A14}$$

which means that the real part is even in x and the imaginary part is odd,

$$\mathbb{F}_{\text{Re}}(-x) \;=\; \quad \mathbb{F}_{\text{Re}}(x) \tag{A15}$$

$$\mathbb{F}_{\text{Im}}(-x) \;=\; -\mathbb{F}_{\text{Im}}(x) \;. \tag{A16}$$

As with expressions (A6) and (A7), the real and imaginary parts can be written

$$\mathbb{F}_{\text{Re}}(x) \;=\; 1 - 2x \, \text{daw}\,(x) \tag{A17}$$

$$\mathbb{F}_{\text{Im}}(x) \;=\; \pi \, \bar{\rho}(x) = \sqrt{\pi} \, x \, e^{-x^2} \;. \tag{A18}$$

Let us now establish the forms (A6) and (A7) for the real and imaginary parts of $F(v)$. Staring with

$$\frac{1}{y - x - i\eta} = \mathsf{P}\frac{1}{y - x} + i\pi \, \delta(y - x) \;, \tag{A19}$$

the imaginary part becomes

$$\mathbb{F}_{\text{Im}}(x) \;=\; \int_{-\infty}^{\infty} dy \, \text{Im} \, \frac{\bar{\rho}(y)}{y - x - i\eta} = \int_{-\infty}^{\infty} dy \, \bar{\rho}(y) \, \pi \delta(y - x) = \pi \bar{\rho}(x) \;. \tag{A20}$$

The real part of the function must be evaluated by a principal part integral,

$$\mathbb{F}_{\text{Re}}(x) \;=\; \mathsf{P} \int_{-\infty}^{\infty} dy \, \frac{\bar{\rho}(y)}{y - x} \;. \tag{A21}$$

Let us add and subtract unity in the form

$$\int_{-\infty}^{\infty} \frac{dy}{y} \, \bar{\rho}(y) = 1 \;, \tag{A22}$$

so that

$$\mathbb{F}_{\text{Re}}(x) \;=\; 1 + \mathsf{P} \int_{-\infty}^{\infty} dy \left[\frac{\bar{\rho}(y)}{y - x} - \frac{\bar{\rho}(y)}{y} \right] \tag{A23}$$

$$=\; 1 + \mathsf{P} \int_{-\infty}^{\infty} dy \, \frac{x}{y(y - x)} \, \bar{\rho}(y) \tag{A24}$$

$$=\; 1 + \frac{x}{\sqrt{\pi}} \mathsf{P} \int_{-\infty}^{\infty} dy \, \frac{e^{-y^2}}{y - x} \;. \tag{A25}$$

16

Making the change of variables $y' = y - x$ (and dropping the prime) we can write

$$\mathbb{F}_{\text{Re}}(x) = 1 + \frac{x}{\sqrt{\pi}} \, \mathsf{P} \int_{-\infty}^{\infty} \frac{dy}{y} \, e^{-(y+x)^2} \tag{A26}$$

$$= 1 + \frac{x \, e^{-x^2}}{\sqrt{\pi}} \, \mathsf{P} \int_{-\infty}^{\infty} \frac{dy}{y} \, e^{-y^2 - 2xy} \tag{A27}$$

$$= 1 + \frac{x \, e^{-x^2}}{\sqrt{\pi}} \, \lim_{\epsilon \to 0^+} \left[\int_{-\infty}^{-\epsilon} \frac{dy}{y} \, e^{-y^2 - 2xy} + \int_{\epsilon}^{\infty} \frac{dy}{y} \, e^{-y^2 - 2xy} \right] . \tag{A28}$$

In the last expression we have used the definition of the principal part integration. Making a change of variables $y' = -y$ in the first integral in square brackets gives (and again dropping the prime)

$$\int_{-\infty}^{-\epsilon} \frac{dy}{y} \, e^{-y^2 - 2xy} = - \int_{\epsilon}^{\infty} \frac{dy}{y} \, e^{-y^2 + 2xy} \, , \tag{A29}$$

and this allows us to write

$$\mathbb{F}_{\text{Re}}(x) = 1 - \frac{x \, e^{-x^2}}{\sqrt{\pi}} \, \lim_{\epsilon \to 0^+} \int_{\epsilon}^{\infty} \frac{dy}{y} \, e^{-y^2} \left[e^{2xy} - e^{-2xy} \right] . \tag{A30}$$

The term in square braces is just $2\sinh(2xy)$, which renders the factor $1/y$ harmless when the limit $\epsilon \to 0^+$ is taken,

$$\mathbb{F}_{\text{Re}}(x) = 1 - \frac{2x \, e^{-x^2}}{\sqrt{\pi}} \int_{0}^{\infty} dy \, e^{-y^2} \, \frac{\sinh 2xy}{y} = 1 - 2x \, \text{daw}(x) \, . \tag{A31}$$

The latter form hold because this is just another integral representation of the Dawson function,

$$\text{daw}(x) = \frac{e^{-x^2}}{\sqrt{\pi}} \int_{0}^{\infty} dy \, e^{-y^2} \, \frac{\sinh 2xy}{y} \, . \tag{A32}$$

Compare this with

$$\text{daw}(x) = e^{-x^2} \int_{0}^{x} dy \, e^{y^2} \, . \tag{A33}$$