Clog Doc: dedxFortran.bps1.0.tex

Stopping Power in Clog

Robert Singleton

Research Notes

Project:
Clog Doc
Path of TeX Source:
Clog/doc/dedx/dedxFortran.bps1.0.tex
Last Modified By:
Robert Singleton
3/4/2020
Date Started:
3/4/2020
Date:
v1.0 Wednesday 4th March, 2020 14:52

Stopping Power in Clog

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

(Dated: 3/4/2020)

Abstract

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

Contents

I. The Stopping Power Expression of BPS 3
A. Description of the Plasma 3
B. The Representation of BPS in Ref. [1] 4
1. The Classical Result 4
2. Quantum Corrections 5
C. A More Compact Representation of BPS 6

II. Units 9
A. Thermal Velocity 9
B. Atomic Units 12
C. Break into Smaller Pieces for Coding 14

III. Asymptotic Limits 16
A. Classical 16
1. Small Velocity Limit 16

IV. Fitting 19
A. The General Method 19
B. An Example: the Dawson Integral 19

V. The Main Subroutine 23
A. Defining Parameters and the Main Call 25
B. Diagnostics 27
C. Call Classical and Quantum Stopping Power 28

VI. The Quantum Correction 29
A. The Quantum Integral by Gaussian Quadrature 31
B. The Large-s Limit 33
1. Fits to the Digamma Function and its Derivatives 36
2. The Digamma Function: repsi 38
3. The Derivative: repsi1 39
4. The Second Derivative: repsi2 41
C. The Small-s Limit 44
1. Fitting the Functions K_1 and K_3 46

VII. The Classical Contribution 53
A. Contribution $\boldsymbol{H_b}$ 55
B. Contribution $\boldsymbol{I_2}$ 57
C. Contribution $\boldsymbol{I_1}$ 58
1. $J_1(x)$ 60
2. $J_2(a)$ 63
3. $J_3(x)$ 66
4. $J_4(x)$ 69

References 98

I. THE STOPPING POWER EXPRESSION OF BPS

A. Description of the Plasma

In what follows, I will use the stopping power calculations of Ref. [1] (BPS). We consider a projectile moving through a hot fully ionized plasma. The velocity of the projectile is v_p, its charge e_p, and its mass m_p. The plasma will consist of species b of charge e_b and mass m_b, with number density n_b and inverse temperature $\beta_b = 1/T_b$, where the temperature is measured in energy units (by convention $b = 1$ represents the electron species and $b \geq 2$ the ion species). Rationalized units are used for the charge so that, for example, the Coulomb potential energy in three dimensions reads $e_p^2/(4\pi r)$. In summary, the properties characterizing the projectile and the plasma are

$$\text{projectile}: \quad e_p \;\; m_p \;\; v_p \tag{1.1}$$

$$\text{plasma}: \quad e_b \;\; m_b \;\; n_b \;\; \beta_b \; . \tag{1.2}$$

For species b the corresponding Debye wave number is defined by

$$\kappa_b^2 = \beta_b \, n_b \, e_b^2 \; . \tag{1.3}$$

and the total Debye wave number κ_{D} is defined by the sum over all the species

$$\kappa_{\text{D}}^2 = \sum\nolimits_b \kappa_b^2 \; . \tag{1.4}$$

The relative and total masses of the projectile and plasma particles are

$$\frac{1}{m_{pb}} \;=\; \frac{1}{m_p} + \frac{1}{m_b} \tag{1.5}$$

$$M_{pb} \;=\; m_p + m_b \; . \tag{1.6}$$

For a dilute plasma the dielectric function is given by[2]

$$\epsilon(\mathbf{k}, \omega) = 1 + \sum\nolimits_c \frac{e_c^2}{k^2} \int \frac{d^\nu \mathbf{p}_c}{(2\pi\hbar)^\nu} \frac{1}{\omega - \mathbf{k} \cdot \mathbf{v}_c + i\eta} \, \mathbf{k} \cdot \frac{\partial}{\partial \mathbf{p}_c} f_c(\mathbf{p}_c) \, , \tag{1.7}$$

where the prescription $\eta \to 0^+$ is implicit and defines the correct retarded response, and f_c is the Maxwell distribution function

$$f_c(\mathbf{p}_c) = n_c \left(\frac{2\pi\hbar^2 \beta_b}{m_c} \right)^{\nu/2} \exp\left\{ -\frac{\beta_c}{2} \, m_c v_c^2 \right\} \; . \tag{1.8}$$

Computing the derivative in Eq. (1.7) and then integrating out the momentum components of $\mathbf{p}_c$ perpendicular to $\mathbf{k}$ gives the structure

$$k^2 \, \epsilon(k, k v_p \cos\theta) = k^2 + F(v_p \cos\theta) \, . \tag{1.9}$$

The F function appears in the form of a dispersion relation

$$F(u) = -\int_{-\infty}^{+\infty} dv\, \frac{\rho_{\rm total}(v)}{u - v + i\eta}\,, \tag{1.10}$$

with the spectral weight

$$\rho_{\rm total}(v) = \sum\nolimits_c \rho_c(v)\,, \tag{1.11}$$

where

$$\rho_c(v) \;=\; \kappa_c^2 \sqrt{\frac{\beta_c m_c}{2\pi}}\; v\; \exp\left\{-\frac{1}{2}\beta_c m_c v^2\right\}\;. \tag{1.12}$$

For future use, we note that F satisfies the relations

$$F(-v) \;=\; F^*(v)\;. \tag{1.13}$$

$$\mathrm{Im}\, F(v) \;=\; \frac{1}{2i}\left[F(v) - F^*(v)\right] = \pi\rho_{\rm total}(v) \tag{1.14}$$

$$\rho_c(-v) \;=\; -\rho_c(v)\;. \tag{1.15}$$

The argument v of F and ρ_c has units of velocity. The real and imaginary parts of F can be written

$$F_{\rm R}(v) \;=\; \sum_b \kappa_b^2 \left[1 - 2\sqrt{\frac{\beta_b m_b}{2}}\; v\; \mathrm{daw}\left(\sqrt{\frac{\beta_b m_b}{2}}\, v\right)\right] \tag{1.16}$$

$$F_{\rm I}(v) \;=\; \sqrt{\pi}\sum_b \kappa_b^2 \sqrt{\frac{\beta_b m_b}{2}}\;\; v\, \exp\left[-\frac{\beta_b m_b}{2}\, v^2\right] = \pi\rho_{\rm total}(v)\;, \tag{1.17}$$

and the Dawson integral is defined by

$$\mathrm{daw}(x) \equiv \int_0^x dy\, e^{y^2 - x^2} = \frac{\sqrt{\pi}}{2}\, e^{-x^2}\mathrm{erfi}(x)\;. \tag{1.18}$$

B. The Representation of BPS in Ref. [1]

1. The Classical Result

The complete energy loss to the plasma species b in the classical case can be written[1]

$$\frac{dE_b^{\rm C}}{dx} = \frac{dE_{b,{\rm S}}^{\rm C}}{dx} + \frac{dE_{b,{\rm R}}^{<}}{dx}\,, \tag{1.19}$$

in which the two contribution are given by[1] :

$$\frac{dE^{\rm C}_{b,{\rm S}}}{dx} = \frac{e_p^2}{4\pi}\,\frac{\kappa_b^2}{m_p\,v_p}\,\left(\frac{m_b}{2\pi\beta_b}\right)^{1/2}\int_0^1 du\,u^{1/2}\,\exp\left\{-\frac{1}{2}\,\beta_b m_b v_p^2\,u\right\}$$
$$\left\{\left[-\ln\left(\beta_b\,\frac{e_p e_b\,K}{4\pi}\,\frac{m_b}{m_{pb}}\,\frac{u}{1-u}\right)+2-2\gamma\right]\left[\beta_b\,M_{pb}\,v_p^2-\frac{1}{u}\right]+\frac{2}{u}\right\}, \tag{1.20}$$

and

$$\frac{dE^{<}_{b,{\rm R}}}{dx} = \frac{e_p^2}{4\pi}\,\frac{i}{2\pi}\int_{-1}^{+1} d\cos\theta\,\cos\theta\,\frac{\rho_b(v_p\cos\theta)}{\rho_{\rm total}(v_p\cos\theta)}\,F(v_p\cos\theta)\ln\left(\frac{F(v_p\cos\theta)}{K^2}\right)$$
$$-\frac{e_p^2}{4\pi}\,\frac{i}{2\pi}\,\frac{1}{\beta_b m_p v_p^2}\,\frac{\rho_b(v_p)}{\rho_{\rm total}(v_p)}\left[F(v_p)\ln\left(\frac{F(v_p)}{K^2}\right)-F^*(v_p)\ln\left(\frac{F^*(v_p)}{K^2}\right)\right]. \tag{1.21}$$

The total result does not depend upon the arbitrary wave number K, and choosing K to be a suitable multiple of the Debye wave number of the plasma often simplifies the final results.

2. Quantum Corrections

In the previous section, our discussion was only for those cases in which classical physics applies. In these cases, the quantum parameters

$$\eta_{pb} = \frac{e_p e_b}{4\pi\hbar\,v_{pb}}\,, \tag{1.22}$$

are large. In the energy loss problem, these are the only independent dimensionless parameters that entail the quantum unit, Planck's constant $\hbar$. The parameters are large when the average relative velocity v_{pb} is small, and as far as an η_{pb} parameter is concerned this corresponds to the formal limit $\hbar \to 0$. We now treat the general case where the size of the quantum parameters η_{pb} has no restriction. The energy loss to the plasma species b in the general case appears as

$$\frac{dE_b}{dx} = \frac{dE_b^{\rm C}}{dx} + \frac{dE_b^{\rm Q}}{dx}\,, \tag{1.23}$$

[1] To save writing, we use e to denote the absolute value of the charge of a particle. Thus $e_p e_b$ is always positive even if projectile (p) and plasma (b) particles have charges of opposite sign.

where, the quantum correction is given by[1]

$$\frac{dE_b^{\rm Q}}{dx} = \frac{e_p^2}{4\pi}\frac{\kappa_b^2}{2\beta_b m_p v_p^2}\left(\frac{\beta_b m_b}{2\pi}\right)^{1/2}\int_0^\infty dv_{pb}\ \left\{2\,{\rm Re}\,\psi\left(1+i\eta_{pb}\right)-\ln\eta_{pb}^2\right\}$$
$$\left\{\left[1+\frac{M_{pb}}{m_b}\frac{v_p}{v_{pb}}\left(\frac{1}{\beta_b m_b v_p v_{pb}}-1\right)\right]\exp\left\{-\frac{1}{2}\beta_b m_b\left(v_p-v_{pb}\right)^2\right\}\right.$$
$$\left.-\left[1+\frac{M_{pb}}{m_b}\frac{v_p}{v_{pb}}\left(\frac{1}{\beta_b m_b v_p v_{pb}}+1\right)\right]\exp\left\{-\frac{1}{2}\beta_b m_b\left(v_p+v_{pb}\right)^2\right\}\right\}. \quad (1.24)$$

Here

$$v_{pb} = |\mathbf{v}_p-\mathbf{v}_b| \quad (1.25)$$
$$\eta_{pb} = \frac{e_p e_b}{4\pi\hbar v_{pb}}, \quad (1.26)$$

with $\psi(z)$ being the logarithmic derivative of the gamma function, and

$${\rm Re}\,\psi(1+i\eta)=\sum_{k=1}^{\infty}\frac{1}{k}\,\frac{\eta^2}{k^2+\eta^2}-\gamma\,. \quad (1.27)$$

C. A More Compact Representation of BPS

For coding purposes I will write the classical contribution as

$$\frac{dE_b^{\rm C}}{dx}(v_p) = \frac{e_p^2\,\kappa_b^2}{4\pi}\frac{m_b}{m_p}\frac{1}{\sqrt{2\pi\,\beta_b m_b v_p^2}}\int_0^1 du\,u^{1/2}\,e^{-\beta_b m_b v_p^2\,u/2}\left\{\frac{2}{u}\ +\right.$$
$$\left.\left[-\ln\left(\beta_b\,\frac{e_p e_b\,K}{4\pi}\,\frac{m_b}{m_{pb}}\,\frac{u}{1-u}\right)+2-2\gamma\right]\left[\beta_b\,M_{pb}\,v_p^2-\frac{1}{u}\right]\right\}$$
$$+\ \frac{e_p^2}{4\pi}\frac{1}{4\pi}\int_{-1}^{+1}du\,u\,H_b(v_p u)\ -\ \frac{e_p^2}{4\pi}\,\frac{1}{2\pi}\,\frac{1}{\beta_b m_p v_p^2}\,H_b(v_p)\,. \quad (1.28)$$

where the K-dependent function is defined by

$$H_b(v)\equiv i\,\frac{\rho_b(v)}{\rho_{\rm total}(v)}\left[F(v)\ln\left(\frac{F(v)}{K^2}\right)-F^*(v)\ln\left(\frac{F^*(v)}{K^2}\right)\right]. \quad (1.29)$$

The quantum correction can be expressed as

$$\frac{dE_b^{\mathrm{Q}}}{dx}(v_p) = \frac{e_p^2\kappa_b^2}{4\pi}\frac{2}{\sqrt{2\pi\beta_b m_p v_p^2}}\, e^{-\beta_b m_b v_p^2/2}\int_0^\infty du e^{-\beta_b m_b v_p^2\, u^2/2}\Big\{\ln(\eta_b/u) - \mathrm{Re}\,\psi(1+i\eta_b/u)\Big\}$$
$$\left\{\frac{M_{pb}}{m_p}\frac{1}{u}\left(\cosh(\beta_b m_b v_p^2\, u) - \frac{\sinh(\beta_b m_b v_p^2\, u)}{\beta_b m_b v_p^2\, u}\right) - \frac{m_b}{m_p}\,\sinh(\beta_b m_b v_p^2 u)\right\}\,.$$

$$\eta_b = \frac{e_p e_b}{4\pi\hbar v_p} \tag{1.30}$$

To express the classical stopping power in the form (1.28), note that the first term is same as (1.20), while the next two terms of (1.28) derive from (1.21) in the following way. The last term of (1.28) follows trivially from (1.29), while the second term follows from the first line of (1.21) with the variable replacement $u = \cos\theta$ and the reflection property (1.13) of F. In particular, we use

$$-\int_{-1}^1 du\, u\, \frac{\rho_b(v_p u)}{\rho_{\mathrm{total}}(v_p u)} F^*(v_p u)\ln\left(\frac{F^*(v_p u)}{K^2}\right) = \int_{-1}^1 du\, u\, \frac{\rho_b(-v_p u)}{\rho_{\mathrm{total}}(-v_p u)} F^*(-v_p u)\ln\left(\frac{F^*(-v_p u)}{K^2}\right)$$
$$= \int_{-1}^1 du\, u\, \frac{\rho_b(v_p u)}{\rho_{\mathrm{total}}(v_p u)} F(v_p u)\ln\left(\frac{F(v_p u)}{K^2}\right)\,, \tag{1.31}$$

from which it follows that

$$\int_{-1}^1 du\, u\, H_b(v_p u) = i\int_{-1}^1 du\, u\, \frac{\rho_b(v_p u)}{\rho_{\mathrm{total}}(v_p u)}\left[F(v_p u)\ln\left(\frac{F(v_p u)}{K^2}\right) - F^*(v_p u)\ln\left(\frac{F^*(v_p u)}{K^2}\right)\right]$$
$$= 2\,i\int_{-1}^1 du\, u\, \frac{\rho_b(v_p u)}{\rho_{\mathrm{total}}(v_p u)}\; F(v_p u)\ln\left(\frac{F(v_p u)}{K^2}\right)\,. \tag{1.32}$$

The representation (1.30) of the quantum correction (1.24) follows from

$$\begin{aligned}
\frac{dE_b^{\mathrm{Q}}}{dx} &= \frac{e_p^2}{4\pi}\,\frac{\kappa_b^2}{2\beta_b m_p v_p^2}\left(\frac{\beta_b m_b}{2\pi}\right)^{1/2}\int_0^\infty dv_{pb}\ \left\{2\,\mathrm{Re}\,\psi\left(1+i\eta_{pb}\right)-\ln\eta_{pb}^2\right\} \\
&\quad\left\{\left[1+\frac{M_{pb}}{m_b}\frac{v_p}{v_{pb}}\left(\frac{1}{\beta_b m_b v_p v_{pb}}-1\right)\right]\exp\left\{-\frac{1}{2}\beta_b m_b\left(v_p-v_{pb}\right)^2\right\}\right. \\
&\quad\left.-\left[1+\frac{M_{pb}}{m_b}\frac{v_p}{v_{pb}}\left(\frac{1}{\beta_b m_b v_p v_{pb}}+1\right)\right]\exp\left\{-\frac{1}{2}\beta_b m_b\left(v_p+v_{pb}\right)^2\right\}\right\} \qquad (1.33)
\end{aligned}$$

$$\begin{aligned}
&= \frac{e_p^2}{4\pi}\,\frac{\kappa_b^2}{\beta_b m_p v_p}\left(\frac{\beta_b m_b}{2\pi}\right)^{1/2}\int_0^\infty du\ \left\{\mathrm{Re}\,\psi\left(1+i\eta_b/u\right)-\ln(\eta_b/u)\right\} \qquad : v_{pb}=v_p u \\
&\quad\left\{\left[1+\frac{M_{pb}}{m_b}\frac{1}{u}\left(\frac{1}{\beta_b m_b v_p^2\,u}-1\right)\right]\exp\left\{-\frac{1}{2}\beta_b m_b v_p^2\left(u-1\right)^2\right\}\right. \\
&\quad\left.-\left[1+\frac{M_{pb}}{m_b}\frac{1}{u}\left(\frac{1}{\beta_b m_b v_p^2\,u}+1\right)\right]\exp\left\{-\frac{1}{2}\beta_b m_b v_p^2\left(u+1\right)^2\right\}\right\} \qquad (1.34)
\end{aligned}$$

$$\begin{aligned}
&= \frac{e_p^2}{4\pi}\,\kappa_b^2\frac{e^{-\beta_b m_b v_p^2/2}}{\sqrt{2\pi\beta_b m_p v_p^2}}\int_0^\infty du\ e^{-\beta_b m_b v_p^2\,u^2/2}\left\{\mathrm{Re}\,\psi\left(1+i\eta_b/u\right)-\ln(\eta_b/u)\right\} \\
&\left\{\left[\frac{m_b}{m_p}+\frac{M_{pb}}{m_p u}\left(\frac{1}{\beta_b m_b v_p^2\,u}-1\right)\right]e^{\beta_b m_b v_p^2\,u}-\left[\frac{m_b}{m_p}+\frac{M_{pb}}{m_p u}\left(\frac{1}{\beta_b m_b v_p^2\,u}+1\right)\right]e^{-\beta_b m_b v_p^2\,u}\right\}
\end{aligned}$$

$$\begin{aligned}
&= \frac{e_p^2}{4\pi}\,\kappa_b^2\frac{2\,e^{-\beta_b m_b v_p^2/2}}{\sqrt{2\pi\beta_b m_p v_p^2}}\int_0^\infty du\ e^{-\beta_b m_b v_p^2\,u^2/2}\left\{\mathrm{Re}\,\psi\left(1+i\eta_b/u\right)-\ln(\eta_b/u)\right\} \\
&\quad\left\{\frac{m_b}{m_p}\,\sinh(\beta_b m_b v_p^2 u)+\frac{M_{pb}}{m_p}\,\frac{1}{u}\left(\frac{\sinh(\beta_b m_b v_p^2\,u)}{\beta_b m_b v_p^2\,u}-\cosh(\beta_b m_b v_p^2\,u)\right)\right\}.
\end{aligned}$$

II. UNITS

A. Thermal Velocity

It will be convenient to express the projectile velocity v_p in units of the thermal velocity of a reference plasma species. For particles of mass m, we define the thermal velocity $v_{\rm th}$ from the equipartition theorem,

$$\frac{1}{2}\,m\,v_{\rm th}^2 = \frac{3}{2}\,T\ , \tag{2.1}$$

and therefore $v_{\rm th} = \sqrt{3/\beta m}$. Since there are variations on this definition in the literature, we will define the the *dimensionless* thermal velocity $\bar{v}_p$ of the projectile as

$$v_p \ = \ \bar{v}_p \cdot v_{\rm th} \tag{2.2}$$

$$v_{\rm th} \ \equiv \ \sqrt{\frac{r}{\beta_m\,m}}\ , \tag{2.3}$$

where m is the mass of a reference plasma species, β_m is the corresponding temperature, and and r is an arbitrary real number. We will usually take m to be the electron mass, and $r = 3$ (but some authors take $r = 2$ or $r = 1$, and for this reason we keep r arbitrary for now). From here on we will also express mass in units of m, *i.e.* in terms of the *dimensionless* mass ratio $m_b^0 = m_b/m$, or $m_{pb}^0 = m_{pb}/m$, or $M_{pb}^0 = M_{pb}/m$. Dimensionless combinations such as $\beta_b m_b\,v_p^2$ will always appear together, and using (2.2) we can express these as $\beta_b m_b\,v_p^2 = r_b\,m_b^0\,\bar{v}_p^2$ where we have defined the quantity

$$r_b \equiv r\,\frac{\beta_b}{\beta_m}\ . \tag{2.4}$$

The classical contribution takes the form

$$\begin{aligned}\frac{dE_b^{\rm C}}{dx}(\bar{v}_p) \ = \ & \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{m_b^0}{m_p^0}\,\frac{1}{\sqrt{2\pi\,r_b m_b^0 \bar{v}_p^2}}\int_0^1 du\,u^{1/2}\,e^{-r_b\,m_b^0\,\bar{v}_p^2\,u/2}\Bigg\{\frac{2}{u}\ + \\ & \left[-\ln\left(\beta_b\,\frac{e_p e_b\,K}{4\pi}\,\frac{m_b^0}{m_{pb}^0}\,\frac{u}{1-u}\right) + 2 - 2\gamma\right]\left[r_b\,M_{pb}^0\,\bar{v}_p^2 - \frac{1}{u}\right]\Bigg\} \\ & + \ \frac{e_p^2}{4\pi}\frac{1}{4\pi}\int_{-1}^{+1} du\,u\,H_b(v_p u) \ - \ \frac{e_p^2}{4\pi}\,\frac{1}{2\pi}\,\frac{1}{r_b m_p^0 v_p^2}\,H_b(v_p)\ ,\end{aligned} \tag{2.5}$$

while the quantum correction is

$$\begin{aligned}\frac{dE_b^{\rm Q}}{dx}(\bar{v}_p) \ = \ & \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{2}{\sqrt{2\pi\,r_b m_p^0 \bar{v}_p^2}}\,e^{-r_b m_b^0 \bar{v}_p^2/2}\int_0^\infty du e^{-r_b m_b^0 \bar{v}_p^2\,u^2/2}\Big\{\ln(\eta_b/u) - {\rm Re}\,\psi(1+i\eta_b/u)\Big\} \\ & \left\{\frac{M_{pb}^0}{m_p^0}\,\frac{1}{u}\left(\cosh(r_b m_b^0 \bar{v}_p^2\,u) - \frac{\sinh(r_b m_b^0 \bar{v}_p^2\,u)}{r_b m_b^0 \bar{v}_p^2\,u}\right) - \frac{m_b^0}{m_p^0}\,\sinh(r_b m_b^0 \bar{v}_p^2\,u)\right\}\ .\end{aligned}$$

The quantum correction is complete as it stands, but the classical contribution requires more work to re-express $H_b(v_p)$ as a function of the dimensionless velocity $\bar{v}_p$.

We continue by expressing $\rho_c(v)$ of (1.12) in terms of the dimensionless variable $\bar{v}$ defined by $v = v_{\rm th}\,\bar{v}$:

$$\rho_c(v) \;=\; \kappa_c^2 \sqrt{\frac{\beta_c m_c v_{\rm th}^2}{2\pi}}\;\bar{v}\;e^{-\beta_c m_c v_{\rm th}^2\,\bar{v}^2/2} = \kappa_c^2 \sqrt{\frac{r_c m^0}{2\pi}}\;\bar{v}\;e^{-r_c m^0\,\bar{v}^2/2}\;, \tag{2.6}$$

which allows us to define the spectral weight as

$$\bar{\rho}_c(x) \;=\; \frac{\kappa_c^2}{\sqrt{\pi}}\;x\;e^{-x^2} \tag{2.7}$$

$$\bar{\rho}_{\rm total}(\bar{v}) \;=\; \sum\nolimits_b \bar{\rho}_b(x_b) \qquad x_b = \bar{v}\,\sqrt{r_b m^0/2}\;. \tag{2.8}$$

Note that $\bar{\rho}$ itself still has dimensions of κ^2. We then define

$$\bar{F}(\bar{v}) = -\int_{-\infty}^{+\infty} d\bar{u}\,\frac{\bar{\rho}_{\rm total}(\bar{u})}{\bar{v}-\bar{u}+i\eta}\,, \tag{2.9}$$

where now $\bar{v}$, $\bar{u}$, and η are dimensionless, and the real and imaginary parts become

$$\bar{F}_{\rm R}(\bar{v}) \;=\; \sum_b \kappa_b^2\Big[1 - 2x_b\,{\rm daw}\,(x_b)\Big] \qquad x_b = \bar{v}\,\sqrt{r_b m^0/2} \tag{2.10}$$

$$\bar{F}_{\rm I}(\bar{v}) \;=\; \sqrt{\pi}\sum_b \kappa_b^2\,x_b\,e^{-x_b^2} = \pi\bar{\rho}_{\rm total}(\bar{v})\;. \tag{2.11}$$

Note that $F(v) = F(v_{\rm th}\bar{v}) = \bar{F}(\bar{v})$ and that $\rho_c(v) = \rho_c(v_{\rm th}\bar{v}) = \bar{\rho}_c(\bar{v})$. This prompts us to define

$$\bar{H}_b(\bar{v}) \equiv i\,\frac{\bar{\rho}_b(\bar{v})}{\bar{\rho}_{\rm total}(\bar{v})}\left[\bar{F}(\bar{v})\ln\left(\frac{\bar{F}(\bar{v})}{K^2}\right) - \bar{F}^*(\bar{v})\ln\left(\frac{\bar{F}^*(\bar{v})}{K^2}\right)\right]\,, \tag{2.12}$$

so that $H_b(v) = H_b(v_{\rm th}\bar{v}) = \bar{H}_b(\bar{v})$. In particular the last two term of the classical piece contain

$$H_b(v_p u) = H_b(v_{\rm th}\bar{v}_p\,u) = \bar{H}_b(\bar{v}_p u)\;\;, \tag{2.13}$$

which allows us to write the expression completely in terms of $\bar{H}_b(\bar{v})$.

$$\frac{dE_b^{\rm C}}{dx}(\bar{v}_p) \;=\; \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{m_b^0}{m_p^0}\,\frac{1}{\sqrt{2\pi\,r_b m_b^0 \bar{v}_p^2}}\int_0^1 du\,u^{1/2}\,e^{-r_b\,m_b^0\,\bar{v}_p^2\,u/2}\Bigg\{\frac{2}{u}\;+ \\ \left[-\ln\left(\beta_b\,\frac{e_p e_b\,K}{4\pi}\,\frac{m_b^0}{m_{pb}^0}\,\frac{u}{1-u}\right)+2-2\gamma\right]\left[r_b\,M_{pb}^0\,\bar{v}_p^2-\frac{1}{u}\right]\Bigg\} \\ +\;\frac{e_p^2}{4\pi}\frac{1}{4\pi}\int_{-1}^{+1}du\,u\,\bar{H}_b\left(\bar{v}_p u\right)\;\;-\;\;\frac{e_p^2}{4\pi}\,\frac{1}{2\pi}\,\frac{1}{r_b m_p^0\bar{v}_p^2}\,\bar{H}_b\left(\bar{v}_p\right)\;. \tag{2.14}$$

In summary, the classical contribution and the quantum correction are

$$\frac{dE_b^{\rm C}}{dx}(\bar{v}_p) \;=\; \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{m_b^0}{m_p^0}\,\frac{1}{\sqrt{2\pi\,r_b m_b^0 \bar{v}_p^2}}\int_0^1 du\,u^{1/2}\,e^{-r_b\,m_b^0\,\bar{v}_p^2\,u/2}\Bigg\{\frac{2}{u}\;+ \\ \left[-\ln\left(\beta_b\,\frac{e_p e_b\,K}{4\pi}\,\frac{m_b^0}{m_{pb}^0}\,\frac{u}{1-u}\right)+2-2\gamma\right]\left[r_b\,M_{pb}^0\,\bar{v}_p^2-\frac{1}{u}\right]\Bigg\} \\ +\;\frac{e_p^2}{4\pi}\frac{1}{4\pi}\int_{-1}^{+1}du\,u\,\bar{H}_b\left(\bar{v}_p u\right)\;\;-\;\;\frac{e_p^2}{4\pi}\,\frac{1}{2\pi}\,\frac{1}{r_b m_p^0\bar{v}_p^2}\,\bar{H}_b\left(\bar{v}_p\right) \tag{2.15}$$

$$\frac{dE_b^{\rm Q}}{dx}(\bar{v}_p) \;=\; \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{2}{\sqrt{2\pi\,r_b m_p^0\bar{v}_p^2}}\,e^{-r_b m_b^0\bar{v}_p^2/2}\int_0^\infty du e^{-r_b m_b^0\bar{v}_p^2\,u^2/2}\left\{\ln\left(\frac{\bar{\eta}_b}{\bar{v}_p u}\right)-\mathrm{Re}\,\psi\left(1+i\frac{\bar{\eta}_b}{\bar{v}_p u}\right)\right\} \\ \left\{\frac{M_{pb}^0}{m_p^0}\,\frac{1}{u}\left(\cosh(r_b m_b^0\bar{v}_p^2\,u)-\frac{\sinh(r_b m_b^0\bar{v}_p^2\,u)}{r_b m_b^0\bar{v}_p^2\,u}\right)-\frac{m_b^0}{m_p^0}\,\sinh(r_b m_b^0\bar{v}_p^2\,u)\right\}. \\ \bar{\eta}_b = \frac{e_p e_b}{4\pi\hbar\,v_{\rm th}} \tag{2.16}$$

It is convenient to change u-integration variables in the quantum term to $\bar{u} = \bar{v}_p u$:

$$\frac{dE_b^{\mathrm{Q}}}{dx}(\bar{v}_p) = \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{2}{\sqrt{2\pi\,r_b m_p^0 \bar{v}_p^2}}\,e^{-r_b m_b^0 \bar{v}_p^2/2}\int_0^\infty du e^{-r_b m_b^0 \bar{v}_p^2\,u^2/2}\left\{\ln\left(\frac{\bar{\eta}_b}{\bar{v}_p u}\right) - \mathrm{Re}\,\psi\left(1+i\frac{\bar{\eta}_b}{\bar{v}_p u}\right)\right\}$$
$$\left\{\frac{M_{pb}^0}{m_p^0}\,\frac{1}{u}\left(\cosh(r_b m_b^0 \bar{v}_p^2\,u) - \frac{\sinh(r_b m_b^0 \bar{v}_p^2\,u)}{r_b m_b^0 \bar{v}_p^2\,u}\right) - \frac{m_b^0}{m_p^0}\,\sinh(r_b m_b^0 \bar{v}_p^2\,u)\right\}$$

$$= \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{2}{\sqrt{2\pi\,r_b m_p^0 \bar{v}_p^2}}\,e^{-r_b m_b^0 \bar{v}_p^2/2}\int_0^\infty \frac{d\bar{u}}{\bar{v}_p}\,e^{-r_b m_b^0 \bar{u}^2/2}\left\{\ln\left(\frac{\bar{\eta}_b}{\bar{u}}\right) - \mathrm{Re}\,\psi\left(1+i\frac{\bar{\eta}_b}{\bar{u}}\right)\right\}$$
$$\left\{\frac{M_{pb}^0}{m_p^0}\,\frac{\bar{v}_p}{\bar{u}}\left(\cosh(r_b m_b^0 \bar{v}_p\,\bar{u}) - \frac{\sinh(r_b m_b^0 \bar{v}_p\,\bar{u})}{r_b m_b^0 \bar{v}_p\,\bar{u}}\right) - \frac{m_b^0}{m_p^0}\,\sinh(r_b m_b^0 \bar{v}_p\,\bar{u})\right\}$$

$$= \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{2}{\sqrt{2\pi\,r_b m_p^0 \bar{v}_p^2}}\,e^{-r_b m_b^0 \bar{v}_p^2/2}\int_0^\infty du\,e^{-r_b m_b^0 u^2/2}\left\{\ln\left(\frac{\bar{\eta}_b}{u}\right) - \mathrm{Re}\,\psi\left(1+i\frac{\bar{\eta}_b}{u}\right)\right\}$$
$$\left\{\frac{M_{pb}^0}{m_p^0}\,\frac{1}{u}\left(\cosh(r_b m_b^0 \bar{v}_p\,u) - \frac{\sinh(r_b m_b^0 \bar{v}_p\,u)}{r_b m_b^0 \bar{v}_p\,u}\right) - \frac{m_b^0}{m_p^0}\,\frac{\sinh(r_b m_b^0 \bar{v}_p\,u)}{\bar{v}_p}\right\}.$$

B. Atomic Units

Distances will be measured in units of the Bohr Radius,

$$a_0 = \frac{4\pi\hbar^2}{m_e e^2} = 5.3\times 10^{-9}\,\mathrm{cm}\ , \tag{2.17}$$

and temperature in units of eV:

$$K = K^0\,a_o^{-1} \tag{2.18}$$
$$n_b = n_b^0\,a_0^{-3} \tag{2.19}$$
$$\beta_b = \beta_b^0\,\mathrm{eV}^{-1} \tag{2.20}$$
$$mc^2 = m^0\,\mathrm{eV} \tag{2.21}$$
$$m_b^0 = m_b/m\ ,\quad m_p^0 = m_p/m\ ,\quad \text{and}\quad M_{pb}^0 = M_{pb}/m \tag{2.22}$$
$$e_b = Z_b\,e\quad \text{and}\quad e_p = Z_p\,e\ . \tag{2.23}$$

Any quantity with a superscript zero is dimensionless. The binding energy of the electron in a Hydrogen atom is $B_e\,\mathrm{eV} = 13.6\,\mathrm{eV}$, and upon using the expression

$$\frac{e^2}{4\pi a_0} = 2B_e\,\mathrm{eV} = 27\,\mathrm{eV}\ , \tag{2.24}$$

or

$$e^2 = 8\pi B_e \cdot \mathrm{eV} a_0 \ . \tag{2.25}$$

The inverse-squared Debye length takes the form

$$\begin{aligned} \kappa_b^2 \ &= \ \beta_b n_b \, e_b^2 = \left(\beta_b^0 \, \mathrm{eV}^{-1}\right)\left(n_b^0 \, a_0^{-3}\right)\left(Z_b^2 \, 8\pi B_e \cdot \mathrm{eV} a_0\right) \\ &= \ 8\pi B_e \, \beta_b^0 Z_b^2 \, n_b^0 \cdot a_0^{-2} \ , \end{aligned} \tag{2.26}$$

and although we will not use it as such, the combination $e^2\kappa_b^2$ can be written

$$e_p^2 \, \kappa_b^2 = 64\pi^2 B_e^2 \, \beta_b^0 Z_p^2 Z_b^2 \, n_b^0 \cdot \mathrm{eV}/a_0 \ . \tag{2.27}$$

The dimensionless, quantity under the log takes the form

$$\beta_b e_p e_b K/4\pi \ = \ (\beta_b^0 \, \mathrm{eV}^{-1}) \cdot (Z_p Z_b 8\pi B_e \, \mathrm{eV} a_0) \cdot (K^0 a_0^{-1})/4\pi \tag{2.28}$$

$$= \ 2B_e \, \beta_b^0 Z_p Z_b \, K^0 \ . \tag{2.29}$$

We define the *dimensionless* thermal velocity $\bar{v}_{\rm th}$ as the thermal velocity in units of the speed of light, *i.e.* by $v_{\rm th} = c\,\bar{v}_{\rm th}$. Since we have defined $mc^2 = m^0\,\mathrm{eV}$ and $\beta_m = \beta_m^0\,\mathrm{eV}^{-1}$, the thermal velocity can be expressed as $v_{\rm th} = \ c\,\sqrt{r/\beta_m^0\, m^0}$:

$$v_{\rm th} \ = \ c\,\bar{v}_{\rm th} \quad \text{where} \ \ c = 2.998\times 10^{10}\,\mathrm{cm/s} \ , \quad \text{and} \tag{2.30}$$

$$\bar{v}_{\rm th} \ = \ \sqrt{\frac{r}{\beta_m^0\, m^0}} \ . \tag{2.31}$$

It will be convenient to define the quantum parameter $\bar{\eta}_b \equiv e_p e_b/4\pi\hbar v_{\rm th}$ set by the thermal speed, so that

$$\bar{\eta}_b = \left(\frac{Z_p Z_b}{4\pi\hbar c\,\bar{v}_{\rm th}}\right)\left(8\pi\, B_e \cdot \mathrm{eV} a_0\right) = \frac{2B_e\, Z_p Z_b}{\bar{v}_{\rm th}}\ \frac{\mathrm{eV} a_0}{\hbar c} \quad ; \tag{2.32}$$

however, $\hbar c = 197.3\,\mathrm{eVnm} = 3.723\times 10^3\,\mathrm{eV}a_0$ and therefore

$$\bar{\eta}_b = \frac{e_p e_b}{4\pi\hbar v_{\rm th}} = \frac{2B_e\, Z_p Z_b}{\bar{v}_{\rm th}} \times 2.686\times 10^{-4} \ . \tag{2.33}$$

With the following dimensionless parameters,

$$A_b \ \equiv \ \sqrt{\frac{r_b m_b^0}{2}} \qquad E_b \equiv \frac{m_b^0}{m_p^0}\ \frac{1}{\sqrt{2\pi r_b m_b^0}} \qquad F_b \equiv \frac{2}{\sqrt{2\pi r_b m_b^0}} \qquad B_b \equiv r_b M_{pb}^0 \tag{2.34}$$

$$C_b \ \equiv \ 2 - 2\gamma - \ln\!\left(2B_e\, \beta_b^0 Z_p Z_b K^0 m_b^0/m_{pb}^0\right) \quad \text{with} \ \ r_b \equiv r\,\beta_b^0/\beta_m^0 \ , \tag{2.35}$$

the stopping power takes the form

$$\frac{dE_b^{\rm C}}{dx} = \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{E_b}{\bar{v}_p}\int_0^1 du\,u^{1/2}\,e^{-(A_b\,\bar{v}_p)^2\,u}\left\{\frac{2}{u}+\left[C_b-\ln\left(\frac{u}{1-u}\right)\right]\left[B_b\,\bar{v}_p^2-\frac{1}{u}\right]\right\}$$
$$+\frac{e_p^2}{16\pi^2}\int_{-1}^{+1} du\,u\,\bar{H}_b\,(\bar{v}_p\,u) \;-\; \frac{e_p^2}{8\pi^2 m_p^0}\,\frac{1}{r_b\bar{v}_p^2}\,\bar{H}_b\,(\bar{v}_p) \tag{2.36}$$

$$\frac{dE_b^{\rm Q}}{dx} = \frac{e_p^2\,\kappa_b^2}{4\pi}\,\frac{F_b}{\bar{v}_p}\,e^{-(A_b\,\bar{v}_p)^2}\int_0^\infty du e^{-A_b^2\,u^2}\left\{\ln\left(\frac{\bar{\eta}_b}{u}\right)-\mathrm{Re}\,\psi\left(1+i\frac{\bar{\eta}_b}{u}\right)\right\}$$
$$\left\{\frac{M_{pb}^0}{m_p^0}\,\frac{1}{u}\left(\cosh(2A_b^2\bar{v}_p u)-\frac{\sinh(2A_b^2\bar{v}_p u)}{2A_b^2\bar{v}_p u}\right)-\frac{m_b^0}{m_p^0}\,\frac{\sinh(2A_b^2\,\bar{v}_p u)}{\bar{v}_p}\right\}. \tag{2.37}$$

C. Break into Smaller Pieces for Coding

We will break the stopping power into small component functions and handle each separately in our coding.

$$\frac{dE_b^{\rm C}}{dx} = \frac{c_1}{\bar{v}_p}\,\kappa_b^2 E_b\;I_1(A_b\bar{v}_p,\;B_b\bar{v}_p^2,\;C_b)+c_2\,I_2(\bar{v}_p)-\frac{c_3}{r_b\bar{v}_p^2}\,\bar{H}_b(\bar{v}_p) \tag{2.38}$$

$$\frac{dE_b^{\rm Q}}{dx} = \frac{c_1}{\bar{v}_p}\,\kappa_b^2\,F_b\,e^{-(A_b\,\bar{v}_p)^2}\;I_{\rm QM}(A_b^2,\;\bar{\eta}_b,\;\bar{v}_p) \tag{2.39}$$

where we define the functions of dimensionless arguments

$$
\begin{aligned}
I_1(a,b,c) &= \int_0^1 du\, e^{-a^2 u}\left\{\frac{2}{\sqrt{u}} + \left[c - \ln\left(\frac{u}{1-u}\right)\right]\left[b\sqrt{u} - \frac{1}{\sqrt{u}}\right]\right\} \\
I_2(a) &= \int_{-1}^{+1} du\, u\, \bar{H}_b(a\,u) \qquad (2.40) \\
I_{\rm QM}(a,e,v) &= \int_0^\infty du e^{-au^2}\left\{\ln\left(\frac{e}{u}\right) - \mathrm{Re}\,\psi\left(1 + i\frac{e}{u}\right)\right\} \\
&\quad \left\{\frac{M^0_{pb}}{m^0_p}\frac{1}{u}\left(\cosh(2av\,u) - \frac{\sinh(2av\,u)}{2av\,u}\right) - \frac{m^0_b}{m^0_p}\frac{\sinh(2av\,u)}{v}\right\}. \qquad (2.41)
\end{aligned}
$$

and the constants

$$
\begin{aligned}
c_1 &= \frac{e_p^2}{4\pi} = 2B_e Z_p^2 \cdot \mathrm{eV} a_0 \qquad (2.42) \\
c_2 &= \frac{e_p^2}{16\pi^2} = \frac{B_e Z_p^2}{2\pi} \cdot \mathrm{eV}\, a_0 \qquad (2.43) \\
c_3 &= \frac{e_p^2}{8\pi^2}\frac{1}{m^0_p} = \frac{B_e Z_p^2}{\pi m^0_p} \cdot \mathrm{eV}\, a_0\ . \qquad (2.44)
\end{aligned}
$$

III. ASYMPTOTIC LIMITS

The only limit employed in the code so far is the small velocity limit for the classical term. See dedxAsymptotic1.x.tex for the other limits.

A. Classical

1. *Small Velocity Limit*

In this section we seek the small velocity limit $v_p \to 0$ of the classical contribution[2] $dE_b^{\rm C}/dx$. As illustrated in Eq. (1.19), the classical contribution $dE_b^{\rm C}/dx$ is given by the sum of two terms, Eqs. (1.20) and (1.21). The term (1.20) becomes

$$\begin{aligned} & v_p \to 0 \ \text{ or } \ \beta_b m_b v_p^2/2 \ll 1 \ : \\ \frac{dE_{b,\rm S}^{\rm C}}{dx} \ &= \ \frac{e_p^2}{4\pi}\,\frac{\kappa_b^2}{m_p\,v_p}\,\left(\frac{m_b}{2\pi\beta_b}\right)^{1/2}\left\{\left[\ln\left(\beta_b\frac{e_p e_b}{16\pi}K\frac{m_b}{m_{pb}}\right)+2\gamma\right]\left[2-\beta_b v_p^2\left(\frac{2}{3}m_p+m_b\right)\right]\right. \\ & \qquad\qquad \left. -\frac{2}{3}\,\beta_b m_b v_p^2\right\} \ + \ \mathcal{O}(v_p^2)\ , \end{aligned} \tag{3.1}$$

where we have used the following elementary u integrals

$$\int_0^1 du\,u^{1/2} = \frac{2}{3}\,, \qquad \int_0^1 du\,u^{1/2}\ln\left(\frac{1-u}{u}\right) = \frac{4}{3}\,(\ln 2 - 1)\ ,$$

$$\int_0^1 du\,u^{-1/2} = 2\,, \qquad \int_0^1 du\,u^{-1/2}\ln\left(\frac{1-u}{u}\right) = 4\ln 2\,.$$

To obtain the small velocity behavior of Eq. (1.21), we first add and subtract $v_p\cos\theta/v$ in the numerator of the integrand of (1.10) to get

$$F(v_p\cos\theta) = \kappa_{\rm D}^2 - \sum\nolimits_c \kappa_c^2 \int_{-\infty}^{+\infty} dv\,\frac{v_p\cos\theta}{v_p\cos\theta + i\eta - v}\sqrt{\frac{\beta_c m_c}{2\pi}}\exp\left\{-\frac{1}{2}\beta_c m_c v^2\right\}\ , \tag{3.2}$$

where

$$\kappa_{\rm D}^2 = \sum\nolimits_c \kappa_c^2 \tag{3.3}$$

is the total squared Debye wave number of the plasma. We now make use of the relation

$$\frac{1}{v_p\cos\theta - v + i\eta} = -i\pi\delta(v_p\cos\theta - v) + \mathcal{P}\frac{1}{v_p\cos\theta - v}\,, \tag{3.4}$$

[2] Incidentally, large velocities are inconsistent with the purely classical limit, so we can consider $v_p \to \infty$ only when we take quantum corrections into account. That is to say, while we may formally take the large velocity limit of the function $dE_b^{\rm C}/dx$, it alone has no physical validity unless we also add to this term to the large velocity limit of the quantum piece $dE_b^{\rm Q}/dx$.

in which $\mathcal{P}$ denotes the principal part prescription. Since $\mathcal{P}(1/x)$ defines an odd function, the translation $u = v - v_p \cos\theta$ of the integration variable gives

$$\begin{aligned} F(v_p\cos\theta) \;&=\; \kappa_{\rm D}^2 + \pi\, i \sum\nolimits_c \rho_c(v_p\cos\theta) \\ &\quad -2\, v_p\cos\theta \sum\nolimits_c \kappa_c^2 \sqrt{\frac{\beta_c m_c}{2\pi}}\, \exp\left\{-\frac{1}{2}\beta_c m_c\, (v_p\cos\theta)^2\right\} \\ &\quad \times \int_0^\infty \frac{du}{u}\, \sinh\left(\beta_c m_c\, u\, v_p\cos\theta\right)\, \exp\left\{-\frac{1}{2}\beta_c m_c\, u^2\right\}. \end{aligned} \tag{3.5}$$

In this form the small v_p limit is reduced to the evaluation of elementary Gaussian integrals and we have

$$\begin{aligned} &v_p \to 0 : \\ &F(v_p\cos\theta) \;=\; \kappa_{\rm D}^2 - \sum\nolimits_c \kappa_c^2\, \beta_c\, m_c\, v_p^2 \cos^2\theta + O(v_p^4) + \pi i\, \rho_{\rm total}(v_p\cos\theta)\,, \end{aligned} \tag{3.6}$$

where we note that $\rho_{\rm total}(u)$ starts out at order u. Placing this result in Eq. (1.21) produces

$$\begin{aligned} v_p \;&\to\; 0 : \\ \frac{dE^<_{b,{\rm R}}}{dx} \;&=\; -\frac{e_p^2}{4\pi}\,\kappa_b^2 \left(\frac{\beta_b m_b}{2\pi}\right)^{1/2} v_p\, \frac{2}{3}\left[\ln\left(\frac{\kappa_{\rm D}}{K}\right) + \frac{1}{2}\right] \\ &\quad + \frac{e_p^2}{4\pi}\,\frac{\kappa_b^2}{m_p v_p}\left(\frac{m_b}{2\pi\beta_b}\right)^{1/2} \left\{ \left[1 - \frac{1}{2}\beta_b m_b v_p^2\right]\left[2\ln\left(\frac{\kappa_{\rm D}}{K}\right) + 1\right] \right. \\ &\quad \left. - \sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\,\beta_c\, m_c\, v_p^2 + \frac{\pi}{12}\, v_p^2 \left[\sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\,(\beta_c m_c)^{1/2}\right]^2 \right\}. \end{aligned} \tag{3.7}$$

The result (3.1) added to the small velocity limit (3.7) produces

$$\begin{aligned} v_p \;&\to\; 0 \quad \text{or} \quad \beta_b m_b v_p^2/2 \ll 1 : \\ \frac{dE_b^{\rm C}}{dx} \;&=\; \frac{e_p^2}{4\pi}\,\frac{\kappa_b^2}{m_p v_p}\left(\frac{m_b}{2\pi\beta_b}\right)^{1/2} \left\{ \left[\ln\left(\beta_b \frac{e_p e_b}{16\pi}\,\kappa_{\rm D}\frac{m_b}{m_{pb}}\right) + \frac{1}{2} + 2\gamma\right]\left[2 - \beta_b v_p^2\left(\frac{2}{3}\,m_p + m_b\right)\right] \right. \\ &\quad \left. - \frac{2}{3}\,\beta_b m_b v_p^2 - \sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\,\beta_c m_c\, v_p^2 + \frac{\pi}{12}\, v_p^2 \left[\sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\,(\beta_c m_c)^{1/2}\right]^2 \right\} \;+\; \mathcal{O}(v_p^2)\,. \end{aligned} \tag{3.8}$$

Let us now express this result in code variables.

$$
\begin{aligned}
& v_p \;\to\; 0 \quad \text{or} \quad \beta_b m_b v_p^2/2 \ll 1 \quad \text{or} \quad r_b m_b^0 \bar{v}_p^2/2 \ll 1 : \\
\frac{dE_b^{\rm C}}{dx} \;&=\; \frac{e_p^2\kappa_b^2}{4\pi}\,\frac{m_b}{m_p}\,\frac{1}{\sqrt{2\pi\beta_b m_b v_p^2}}\Bigg\{\left[\ln\left(\beta_b\frac{e_p e_b}{16\pi}\,\kappa_{\rm D}\frac{m_b}{m_{pb}}\right)+\frac{1}{2}+2\gamma\right]\left[2-\beta_b m_b v_p^2\left(1+\frac{2}{3}\,\frac{m_p}{m_b}\right)\right] \\
&\qquad -\frac{2}{3}\,\beta_b m_b v_p^2 - \sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\,\beta_c m_c\, v_p^2 + \frac{\pi}{12}\,\left[\sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\left(\beta_c m_c v_p^2\right)^{1/2}\right]^2\Bigg\} \qquad (3.9) \\
&=\; \frac{e_p^2\kappa_b^2}{4\pi}\,\frac{m_b^0}{m_p^0}\,\frac{1}{\sqrt{2\pi r_b m_b^0 \bar{v}_p^2}}\Bigg\{\left[\ln\left(\beta_b\frac{e_p e_b}{16\pi}\,\kappa_{\rm D}\frac{m_b}{m_{pb}}\right)+\frac{1}{2}+2\gamma\right]\left[2-r_b m_b^0\bar{v}_p^2\left(1+\frac{2}{3}\,\frac{m_p^0}{m_b^0}\right)\right] \\
&\qquad -\frac{2}{3}\,r_b m_b^0\bar{v}_p^2 - \sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\,r_c m_c^0\,\bar{v}_p^2 + \frac{\pi}{12}\,\left[\sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\left(r_c m_c^0\bar{v}_p^2\right)^{1/2}\right]^2\Bigg\}\,, \qquad (3.10)
\end{aligned}
$$

where we have written the expression (i) in dimensionless units $\bar{v}_p$ of the thermal velocity $v_{\rm th}$ defined in (2.2), (ii) in dimensionless inverse temperature units defined in (2.4), and (iii) in terms of dimensionless masses m_b^0 defined in (II B). We can write this expression in terms of the variables A_b and E_b defined in (II B), which we repeat here for convenience, and a new variable G_b:

$$
A_b \;\equiv\; \sqrt{\frac{r_b m_b^0}{2}} \qquad E_b \equiv \frac{m_b^0}{m_p^0}\,\frac{1}{\sqrt{2\pi r_b m_b^0}} \qquad (3.11)
$$

$$
G_b \;\equiv\; \frac{1}{2}+2\gamma+\ln\!\left(2B_e\,\beta_b^0 Z_p Z_b \kappa_{\rm D}^0 m_b^0/m_{pb}^0\right) = \frac{1}{2}+2\gamma+\ln\!\left(2B_e\,\beta_b^0 Z_p Z_b \kappa_{\rm D}^0 m_b^0/m_{pb}^0\right) \qquad (3.12)
$$

In expressing the final form of G_b, we have used Eqs. (2.17), (II B), (2.24), and (2.25). We also define $c_1 = e_p^2/4\pi$, as in (2.42), and write (3.10) as

$$
\begin{aligned}
& v_p \;\to\; 0 \quad \text{or} \quad (A_b\bar{v}_p)^2 \ll 1 : \\
\frac{dE_b^{\rm C}}{dx} \;&=\; \frac{c_1}{\bar{v}_p}\,\kappa_b^2\,E_b\Bigg\{2G_b\left[1-A_b^2\bar{v}_p^2\left(1+\frac{2}{3}\,\frac{m_p^0}{m_b^0}\right)\right] \\
&\qquad -\frac{4}{3}\,A_b^2\bar{v}_p^2 - 2\sum_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\,A_c^2\,\bar{v}_p^2 + \frac{\pi}{6}\,\left[\sum\nolimits_c \frac{\kappa_c^2}{\kappa_{\rm D}^2}\,A_c\bar{v}_p\right]^2\Bigg\} + \mathcal{O}\Big((A_b\bar{v}_p)^2\Big) \qquad (3.13)
\end{aligned}
$$

IV. FITTING

A. The General Method

Fits to functions $f(x)$ will be performed with rational polynomials over the positive real numbers. In particular, we will find rational functions $R(x)$ and $Q(x)$ such that

$$f(x) \approx R(x)\, Q(x) \ ; \tag{4.1}$$

we require that $Q(x)$ asymptotes to one for $x \to 0$ and $x \to \infty$, while $R(x)$ is designed to capture the asymptotic behavior of $f(x)$. For example, suppose $f(x) \sim x$ for $x \sim 0$, and $f(x) \sim 1/x$ for large x, then we may take $R = x/(x^2+1)$. We will take Q to be of the form

$$Q_n(x) = \frac{\sum_{\ell=0}^{n} b_\ell x^\ell}{\sum_{\ell=0}^{n} a_\ell x^\ell} \ , \tag{4.2}$$

with $b_n = a_n = 1$ and $a_0 = b_0$, i.e.

$$Q_n(x) = \frac{x^n \ + \ b_{n-1}x^{n-1} \ + \ b_{n-2}x^{n-2} \ + \ \cdots \ + \ b_2x^2 \ + \ b_1x \ + \ b_0}{x^n \ + \ a_{n-1}x^{n-1} \ + \ a_{n-2}x^{n-2} \ + \ \cdots \ + \ a_2x^2 \ + \ a_1x \ + \ b_0} \ . \tag{4.3}$$

Note that $Q_n(x) \to 1$ for $x \to \infty$ and $x \to 0$. We will determine the $2n-1$ variables b_ℓ (with $\ell = 0, 1, \cdots, n-1$) and a_ℓ (with $\ell = 1, 2, \cdots, n-1$) by requiring that the spline-fit agrees at points x_m with $m = 0, 2, \cdots 2n-2$, i.e.

$$R_m Q_n(x_m) = f_m \qquad m = 0, 2, \cdots, 2n-2 \ . \tag{4.4}$$

where $f_m = f(x_m)$ and $R_m = R(x_m)$. Writing $F_m = f_m/R_m$, we find $2n-1$ linear equations

$$\sum_{\ell=1}^{n-1} \left(x_m^\ell \cdot b_\ell - x_m^\ell F_m \cdot a_\ell \right) \ + \ (1 - F_m) \cdot b_0 = (F_m - 1) x_m^n \qquad m = 0, 1, \cdots, 2n-2 \ , \tag{4.5}$$

which may be solved for b_ℓ and a_ℓ.

B. An Example: the Dawson Integral

We need to fit the Dawson integral

$$\mathrm{daw}(x) = \int_0^x dy\, e^{y^2 - x^2} \ , \tag{4.6}$$

and to evaluate the this function analytically at a few points we use

$$\mathrm{daw}(x) = \frac{\sqrt{\pi}}{2}\, e^{-x^2}\, \mathrm{erfi}(x)\ . \tag{4.7}$$

For a detailed evaluation of the integrals and limits that follow, see daw.nb. The asymptotic forms for large and small x are

$x \ll 1$:

$$\mathrm{daw}(x) \;=\; x - \frac{2x^3}{3} + \frac{4x^5}{15} + \mathcal{O}(x^7) \qquad 0.1\%\text{ error for } x < x_{\min} = 0.5 \tag{4.8}$$

$x \gg 1$:

$$\mathrm{daw}(x) \;=\; \frac{1}{2x} + \frac{1}{4x^3} + \frac{3}{8x^5} + \mathcal{O}(x^{-7}) \qquad 0.1\%\text{ error for } x > x_{\max} = 5\ . \tag{4.9}$$

We can use these analytic expressions for $x < x_{\min}$ and $x > x_{\max}$. For values in between we will fit to rational functions with $n = 6$, and approximate

$$\mathrm{daw}(x) = R(x)\, Q_6(x)\ . \tag{4.10}$$

The rational function $Q_6(x)$ goes to unity for large and small x, so we must choose $R(x)$ to yield the correct asymptotic behavior. We have seen that $\mathrm{daw}(x) \sim x$ for $x << 1$ and $\mathrm{daw}(x) \sim 1/2x$ for $x \gg 1$, so we take

$$R(x) = \frac{x}{2x^2+1}\ . \tag{4.11}$$

For $n = 6$, we need $2n - 1 = 11$ data points: we will take m to start at zero and end at ten, and we choose

m	x_m	$\mathrm{daw}(x_m)$
0	0.92413	0.541044
1	0.2	0.194751
2	0.5	0.424436
3	0.7	0.510504
4	1.2	0.507273
5	1.4	0.456507
6	1.6	0.399940
7	2.0	0.301340
8	3.0	0.178271
9	4.0	0.129348
10	8.0	0.0630002

, (4.12)

and solving the linear equations gives

ℓ	b_ℓ	a_ℓ
0	5.735938802443	5.7359388024432
1	−6.736660071378	−6.8237204895090
2	19.979442278715	13.3804115903096
3	−18.550635026076	−14.2130723670491
4	12.265136090570	11.1714434417979
5	−4.672858126848	−4.6630338746894

(4.13)

In the source code, the parameters are set in globalvars.f90 and the the polynomial fits take place in the subroutine daw() of dedx.f90.

globalvars.f90:

```
MODULE globalvars
!
! mathematical constants
!
  REAL,    PARAMETER :: PI   =3.141592654   ! pi
  REAL,    PARAMETER :: SQPI =1.772453851   ! sqrt(pi)
  REAL,    PARAMETER :: GAMMA=0.577215665   ! Euler Gamma
  REAL,    PARAMETER :: LOG2 =0.6931471806  ! ln(2)
  REAL,    PARAMETER :: LOG4 =1.386294361   ! ln(4)
  REAL,    PARAMETER :: LOG8 =2.079441542   ! ln(8)
  REAL,    PARAMETER :: LOG16=2.772588722   ! ln(16)
  REAL,    PARAMETER :: ZETA3=1.202056903   ! zeta(3)
  REAL,    PARAMETER :: EXP2E=3.172218958   ! exp(2*GAMMA)

....
....

! daw() approximates Dawson's integral by rational
! functions with coefficients:
!
  INTEGER, PARAMETER          :: NNDAW=3, NMDAW=2*NNDAW-1 ! NMDAW=5
  REAL,    DIMENSION(0:NMDAW) :: DWB, DWA
  PARAMETER (             &
  DWB=(/                  &
    5.73593880244318E0,  & !b0
   -6.73666007137766E0,  & !b1
    1.99794422787154E1,  & !b2
   -1.85506350260761E1,  & !b3
    1.22651360905700E1,  & !b4
   -4.67285812684807E0/),& !b5
  DWA=(/                  &
    DWB(0),               & !a0
   -6.82372048950896E0,  & !a1
    1.33804115903096E1,  & !a2
   -1.42130723670491E1,  & !a3
    1.11714434417979E1,  & !a4
   -4.66303387468937E0/) ) !a5

....
....

END MODULE globalvars
```

dedx.f90:

```
!
! See daw.nb for details.
!
 FUNCTION daw(x)
```

```
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL  daw

   REAL,    PARAMETER :: XMIN=0.4D0, XMAX=5.D0
   REAL    :: x3, x5, xx, ra, rc
   INTEGER :: n
   IF (x .LE. XMIN) THEN
      x3=x*x*x
      x5=x3*x*x
      daw=x - 2.D0*x3/3.D0 + 4.D0*x5/15.D0
   ELSEIF (x .GE. XMAX) THEN
      x3=x*x*x
      x5=x3*x*x
      daw=1.D0/(2.D0*x)+1.D0/(4.D0*x3)+3.D0/(8.D0*x5)
   ELSE
      ra=0.E0
      rc=0.E0
      xx=1.E0
      DO n=0,NMDAW
         ra=ra+DWA(n)*xx
         rc=rc+DWB(n)*xx
         xx=x*xx
      ENDDO
      ra=ra+xx
      rc=rc+xx
      daw=x/(1.E0+2.E0*x*x)
      daw=daw*rc/ra
   ENDIF
 END FUNCTION daw
```

V. THE MAIN SUBROUTINE

The main driving subroutine that returns the stopping power is called dedx_bps and takes the form

dedx.f90:

```
SUBROUTINE dedx_bps(nni, vp, zp, mp, betab, zb, mb, nb, &
    dedxtot, dedxsumi, dedxctot, dedxcsumi, dedxqtot, dedxqsumi)

  A. define useful global variables
  B. print diagnostics, such as plasma coupling (if desired)
  C. call classical and quantum stopping power

END SUBROUTINE dedx_bps
```

where the input and output variables are described by the following table.

variable name	input or output	description
nni	input	number of ion species
vp	input	projectile velocity $\bar{v}_p$ in thermal units $v_{\rm th}$
zp	input	charge of projectile
mp	input	mass of projectile in eV
betab	input	inverse temperature array of plasma in eV^{-1}
zb	input	charge array of plasma
mb	input	mass array of plasma in eV
nb	input	number density array of plasma in cm^{-3}
dedxtot	output	total stopping power in MeV/μm
dedxsumi	output	ionic stopping power in MeV/μm
dedxctot	output	classical contribution to the total stopping power
dedxcsumi	output	classical contribution to the ionic stopping power
dedxqtot	output	quantum contribution to the total stopping power
dedxqsumi	output	quantum contribution to the ionic stopping power

The subroutine (a) defines some useful global variables, (b) prints diagnostics, and (b) evaluates the stopping power. Global variables are declared as such, but not necessarily set, in the module globalvars.f90. This module also defines some parameters, such as π and the Euler constant γ.

globalvars.f90:

```
MODULE globalvars
!
! mathematical constants
!
  REAL,    PARAMETER :: PI   =3.141592654   ! pi
  REAL,    PARAMETER :: SQPI =1.772453851   ! sqrt(pi)
  REAL,    PARAMETER :: GAMMA=0.577215665   ! Euler Gamma
  REAL,    PARAMETER :: LOG2 =0.6931471806  ! ln(2)
  REAL,    PARAMETER :: LOG4 =1.386294361   ! ln(4)
  REAL,    PARAMETER :: LOG8 =2.079441542   ! ln(8)
  REAL,    PARAMETER :: LOG16=2.772588722   ! ln(16)
  REAL,    PARAMETER :: ZETA3=1.202056903   ! zeta(3)
  REAL,    PARAMETER :: EXP2E=3.172218958   ! exp(2*GAMMA)
!
! physical parameters and conversion factors
!
  REAL,    PARAMETER :: BE=13.6             ! binding energy of Hydrogen
  REAL,    PARAMETER :: CC=2.998E10         ! speed of light
  REAL,    PARAMETER :: MPEV =0.938271998E9 ! proton mass in eV
  REAL,    PARAMETER :: MEEV =0.510998902E6 ! electron mass in eV
  REAL,    PARAMETER :: AMUEV=0.931494012E9 ! AMU in eV
  REAL,    PARAMETER :: KTOEV=8.61772E-5    ! conversion factor
  REAL,    PARAMETER :: CMTOA0=1.8867925E8  ! conversion factor
  REAL,    PARAMETER :: MTR=1.E-6           ! length unit
  REAL,    PARAMETER :: EV=1.E6             ! energy unit
  REAL,    PARAMETER :: CONVFACT=CMTOA0*(MTR*100.)/EV
!
! misc parameters
!
  INTEGER, PARAMETER :: R=3        ! thermal velocity parameter
  INTEGER, PARAMETER :: I=1        ! plasma species index
  REAL               :: K          ! arbitrary wave number units a0^-1

! plasma parameters: values set in dedx_bps
!
! REAL,    DIMENSION(1:NNB)          :: kb2, ab, bb, cb, eb, fb, rb, gb
! REAL,    DIMENSION(1:NNB)          :: ab2, etb, rmb0, rrb0, mb0
  REAL,    DIMENSION(:), ALLOCATABLE :: kb2, ab, bb, cb, eb, fb, rb, gb
  REAL,    DIMENSION(:), ALLOCATABLE :: ab2, etb, rmb0, rrb0, mb0
  LOGICAL, DIMENSION(:), ALLOCATABLE :: lzb
  REAL    :: cp1, cp2, cp3, vth, vthc, mp0, kd
  INTEGER :: NNB  ! number of plasma species = ni+1
....
....
END MODULE globalvars
```

The last part of globalvars.f90 not shown here involves parameters for the functions daw(x), $J_1(x)$, $J_2(x)$, $J_3(x)$, and $J_4(x)$, which are used to calculate the classical contribution to the stopping power and will be described later.

A. Defining Parameters and the Main Call

Some general parameters and mass parameters:

$$\begin{aligned}
\text{mb0} &= m_b^0 = m_b/m && (5.1)\\
\text{rb} &= r_b = r\beta_b/\beta_m && (5.2)\\
\text{rmb0} &= M_{pb}^0/m_p^0 && (5.3)\\
\text{rrb0} &= m_b^0/m_p^0 && (5.4)\\
\text{kb2} &= \kappa_b^2 = 8\pi B_e\,\beta_b^0 Z_b^2\,n_b^0 \quad \text{kb} = \sqrt{\text{kb2}} && (5.5)\\
\text{cp1} &= c_1 = e_p^2/4\pi = 2B_e Z_p^2 && (5.6)\\
\text{cp2} &= c_2 = e_p^2/16\pi^2 = B_e Z_p^2/2\pi && (5.7)\\
\text{cp3} &= c_3 = e_p^2/8\pi^2 m_p^0 = B_e Z_p^2/\pi m_p^0 && (5.8)
\end{aligned}$$

Classical parameters:

$$\begin{aligned}
\text{ab} &= A_b = \sqrt{r_b m_b^0/2} \qquad \text{ab2} \equiv \text{ab}\cdot\text{ab} && (5.9)\\
\text{bb} &= B_b = r_b M_{pb}^0 = r_b(m_p^0 + m_b^0) && (5.10)\\
\text{cb} &= C_b = 2 - 2\gamma - \ln\!\left(2B_e\,\beta_b^0 Z_p Z_b K^0 m_b^0/m_{pb}^0\right) && (5.11)\\
\text{gb} &= G_b = \frac{1}{2} + 2\gamma + \ln\!\left(2B_e\,\beta_b^0 Z_p Z_b \kappa_{\text{D}}^0 m_b^0/m_{pb}^0\right) \ : \text{for the small } \bar{v}_p \text{ limit} && (5.12)\\
\text{eb} &= E_b = \frac{m_b^0}{m_p^0}\,\frac{1}{\sqrt{2\pi r_b m_b^0}} && (5.13)
\end{aligned}$$

Quantum parameters:

$$\begin{aligned}
\text{fb} &= F_b = \frac{2}{\sqrt{2\pi r_b m_b^0}} && (5.14)\\
\text{etb} &= \bar{\eta}_b = \frac{e_p e_b}{4\pi\hbar\, v_{\text{th}}} = \frac{2B_e\,Z_p Z_b}{\bar{v}_{\text{th}}} \times 2.686\times 10^{-4} && (5.15)
\end{aligned}$$

dedx.f90:

```
SUBROUTINE dedx_bps(nni, vp, zp, mp, betab, zb, mb, nb, &
    dedxtot, dedxsumi, dedxctot, dedxcsumi, dedxqtot, dedxqsumi)
  USE globalvars

  IMPLICIT NONE
  INTEGER,                      INTENT(IN)  :: nni    ! number of ions
  REAL,    DIMENSION(1:nni+1), INTENT(IN)  :: betab  ! plasma temp array
  REAL,    DIMENSION(1:nni+1), INTENT(IN)  :: mb, nb ! mass and density
  INTEGER, DIMENSION(1:nni+1), INTENT(IN)  :: zb     ! charge array
  REAL,                        INTENT(IN)  :: vp     ! projectile velocity
  REAL,                        INTENT(IN)  :: zp     ! projectile charge
  REAL,                        INTENT(IN)  :: mp     ! projectile mass
  REAL,                        INTENT(OUT) :: dedxtot,  dedxsumi
  REAL,                        INTENT(OUT) :: dedxctot, dedxcsumi
  REAL,                        INTENT(OUT) :: dedxqtot, dedxqsumi
  REAL, DIMENSION(1:nni+1) :: mpb0, rpb0
  REAL :: betam, kd, mm
```

```
   REAL                        :: e, gd
   REAL, DIMENSION(1:nni+1) :: gpb
   REAL, DIMENSION(1:nni+1) :: ub2, mpb, loglamb

   NNB=nni+1
   ALLOCATE(kb2(1:NNB),ab(1:NNB),bb(1:NNB),cb(1:NNB))
   ALLOCATE(eb(1:NNB),fb(1:NNB),rb(1:NNB),gb(1:NNB))
   ALLOCATE(ab2(1:NNB),etb(1:NNB),rmb0(1:NNB),rrb0(1:NNB))
   ALLOCATE(mb0(1:NNB))
!
! plasma parameters
!
   betam=betab(I)                ! inv temp of index plasma species
   rb=R*betab/betam              ! r_b array
   kb2=8*PI*BE*betab*zb*zb*nb    ! inv Debye length squared
   kd=SUM(ABS(kb2))              ! total inv Debye length
   kd=SQRT(kd)                   ! units aa0^-1
   K =kd                         ! set K to Debye

   mm=mb(I)                      ! mass of index plasma species
   mp0=mp/mm                     ! rescaled proj mass
   cp1=2*BE*zp**2                ! units of eV-a0
   cp2=(BE*zp**2)/(2*PI)         ! units of eV-a0
   cp3=(BE*zp**2)/(PI*mp0)       ! dimensionless parameter
   vthc=SQRT(R/(betam*mm))       ! thermal velocity of mm: units of c
   vth =CC*vthc                  ! thermal velocity of mm: units cm/s

   mb0 =mb/mm                    ! rescaled plasma masses
   mpb0=mp0 + mb0                ! Mpb0
   rpb0=mp0*mb0/mpb0             ! mpb0
   rmb0=mpb0/mp0                 ! rm0=rMb0=(mp0+mb0)/mp0
   rrb0=mb0/mp0                  ! rr0=rmb0=mb0/mp0

   ab  =SQRT(rb*mb0/2)
   ab2 =ab*ab
   bb  =rb*mpb0
   eb  =(mb0/mp0)/SQRT(2*PI*rb*mb0)
   etb =2*BE*ABS(zp*zb)*(2.686E-4)/vthc
   fb  =2/SQRT(2*PI*rb*mb0)
   WHERE ( zb /= 0 )             ! do not take log of zero
      lzb=.TRUE.                 ! flag for future use
      cb  =2 - 2*GAMMA - LOG(ABS((2*BE)*betab*zp*zb*K*mb0/rpb0))
      gb  =0.5 + 2*GAMMA + Log(ABS(0.5*BE*&    ! for small vp limit
           betab*zp*zb*kd*mb0/rpb0))           !
   ELSEWHERE
      cb=0
      gb=0
      lzb=.FALSE.
   ENDWHERE
...
...
```

B. Diagnostics

We now display several diagnostic parameters (commented out for a production run): (i) check for charge neutrality, (ii) display the arbitrary wavenumber K in comparison to the Debye wave number $\kappa_{\rm D}$, (iii) calculation the plasma coupling constants g_{pb}, (iv) display the thermal velocity in cm/s, (v) and print the Coulomb logarithm as defined by Li and Petrasso.

```
...
...
! check for charge neutrality
!
!   e=SUM(zb*nb)
!   PRINT *, 'charge  = ', e

! print K and kd
!
!   PRINT *, 'K(a0^-1)= ', K
!   PRINT *, 'kd      = ', kd

! g-factors
!
!   gpb=2*BE*betam*SQRT(kb2)       ! g_pb
!   gd =2*BE*betam*kd              ! g_d
!   PRINT *,'gD      = ', gd

! thermal velocity (cm/s)
!
!   PRINT *, 'vth     = ', vth
!

! Coulomb log
!
!   ub2=(vp*vthc)**2 + 2/(betab*mb)**2     ! velocity (units of c)
!   mpb=rpb0*mm                            ! reduced mass array (eV)
!   loglamb=(8*PI*zb*zb)**2/(mpb*ub2)**2   ! a0 = 5.29*10^-11 m
!   loglamb=loglamb + (2.69E-4)**2/(2*mpb*mpb*ub2) ! = 2.69*10^-4 eV
!   loglamb=-0.5*LOG(loglamb)
!   PRINT *, 'log(Lam)=',loglamb
...
...
```

C. Call Classical and Quantum Stopping Power

The subroutine dedxc calculates the classical contribution while dedxq finds the quantum correction:

```
...
...

   CALL dedxc(vp,dedxctot,dedxcsumi)   ! returned in MeV/mu-m
   CALL dedxq(vp,dedxqtot,dedxqsumi)   !
   dedxtot =dedxctot  + dedxqtot
   dedxsumi=dedxcsumi + dedxqsumi

   ! PRINT *,"       MeV/mu-m    MeV/mu-m"
   ! PRINT *,"tot:",dedxtot,  dedxsumi
   ! PRINT *,"cl :",dedxctot, dedxcsumi
   ! PRINT *,"qm :",dedxqtot, dedxqsumi

   DEALLOCATE(kb2,ab,bb,cb,eb,fb,rb,gb)
   DEALLOCATE(ab2,etb,rmb0,rrb0,mb0,lzb)
 END SUBROUTINE dedx_bps
```

VI. THE QUANTUM CORRECTION

Since the quantum correction is easier to calculate than the classical contribution, we start here first. The subroutine dedxq evaluates

$$\frac{dE_b^{\rm Q}}{dx} = \underbrace{\frac{c_1}{\bar{v}_p}}_{\rm cp1/vp} \cdot \underbrace{\kappa_b^2\, F_b}_{\rm kb2*fb} \cdot \underbrace{I_{\rm QM}(A_b^2,\, \bar{\eta}_b,\, \bar{v}_p)}_{\rm dedxqi} \tag{6.1}$$

where we define

$$I_{\rm QM}(a,e,v) = e^{-av^2}\int_0^\infty du e^{-au^2}\left\{\ln\left(\frac{e}{u}\right) - \mathrm{Re}\,\psi\left(1+i\frac{e}{u}\right)\right\} \left\{\frac{M_{pb}^0}{m_p^0}\frac{1}{u}\left(\cosh(2av\,u) - \frac{\sinh(2av\,u)}{2av\,u}\right) - \frac{m_b^0}{m_p^0}\frac{\sinh(2av\,u)}{v}\right\}. \tag{6.2}$$

The first two terms of (6.1) are multiplicative factors, and all the work is performed by a call to a subroutine dedxqi that evaluates the integral (6.2). The structure of the quantum driving routine is of the form

```
SUBROUTINE dedxq(vp, dedxqtot, dedxqsumi)
 ...
 ...
 A. loop over species ib=1,NNB (ib=1 is the electron)
 B. calculate IQM for each ib, i.e. call dedxqi
 C. multiply by factors kb2(ib)*fb(ib) and cp1/vp
 ...
 ...
END SUBROUTINE dedxq
```

The full subroutine is listed here:

```
SUBROUTINE dedxq(vp, dedxqtot, dedxqsumi)
  USE globalvars

  IMPLICIT NONE
  REAL, INTENT(IN) :: vp
  REAL, INTENT(OUT):: dedxqtot, dedxqsumi

  REAL, DIMENSION(1:NNB) :: qmb
  REAL                   :: dedxqi, a1, a2, e, rm, rr
  INTEGER                :: ib

  qmb=0
  DO ib=1,NNB            ! sum over plasma species
     IF ( lzb(ib) ) THEN ! compute only if zb(ib) /= 0
        a1=ab(ib)
        a2=a1*a1
        e=etb(ib)
        rm=rmb0(ib)      ! rmb0=rMb0=(mp0+mb0)/mp0
        rr=rrb0(ib)      ! rrb0=rmb0=mb0/mp0
```

```
         qmb(ib)=qmb(ib)+dedxqi(vp,a2,e,rm,rr)
         qmb(ib)=qmb(ib)*kb2(ib)*fb(ib)
      ELSE
         qmb(ib)=0        ! don't compute if zb(ib) = 0
      ENDIF
   ENDDO
   dedxqsumi=SUM(qmb(2:NNB))
   dedxqtot =qmb(1)
   dedxqtot =dedxqtot+dedxqsumi

   dedxqtot=CONVFACT*(cp1/vp)*dedxqtot
   dedxqsumi=CONVFACT*(cp1/vp)*dedxqsumi
END SUBROUTINE dedxq
```

The subroutine dedxq is basically a wrapper for the function dedxqi (to be described momentarily) that sums the quantum components over all species $b =$ ib from 1 to NNB (the total number of plasma species; charged zero species are not counted). Our convention is that $b = 1$ corresponds to electrons, so that the ionic contribution dedxqsumi does not include the $b = 1$ term. During the loop over species: (i) the function dedxqi is called, (ii) the result is multiplied by $\kappa_b^2 F_b =$ kb2(ib) $*$ fb(ib), and (iii) the final result is multiplies by $c_1/\bar{v}_p =$ cp1/vp. The factor CONVFACT converts from eV/a_0 to MeV/μm. The heart of the subroutine is the function dedxqi, which performs a Gaussian quadrature over the integrand d_dedxq, or returns its value based on analytic limits if they apply (the large mass limit for ions and the large thermal velocity limit for electrons).

During the loop over species ib, the arguments of the function dedxqi(vp,a,e,rm,rr) have the following meaning:

variable name	description
vp	projectile velocity $\bar{v}_p$ in thermal units $v_{\rm th}$
a	$A_b^2 =$ ab(ib) $*$ ab(ib)
e	$\bar{\eta}_b =$ etb(ib)
rm	$M_{pb}^0/m_p^0 =$ rmb0(ib)
rr	$m_b^0/m_p^0 =$ rrb0(ib)

while the function itself is defined to be

$$\begin{aligned}\text{dedxqi(a, e, v, rm, rr)} \;=\; e^{-\text{av}^2}\int_0^\infty du\, e^{-\text{a}u^2}\left\{\ln\left(\text{e}/u\right) - \text{repsi}\left(\text{e}/u\right)\right\} \\ \left\{\text{rm}\,\frac{1}{u}\left(\cosh(2\text{av}\,u) - \frac{\sinh(2\text{av u})}{2\text{av}\,u}\right) - \text{rr}\,\frac{\sinh(2\text{av}\,u)}{\text{v}}\right\}\,, \quad (6.3)\end{aligned}$$

with repsi(x) $\equiv \text{Re}\,\psi(1 + i\,x)$.

A. The Quantum Integral by Gaussian Quadrature

We now look at dedxqi in more detail. For the moment we will revert to the notation of (6.2), and write dedxqi as

$$I_{\rm QM}(a,e,v) \;=\; \frac{M^0_{pb}}{m^0_p}\,Q_2(a,e,v) - \frac{m^0_b}{m^0_p}\,Q_1(a,e,v)\;, \tag{6.4}$$

where

$$Q_1(a,e,v) \;=\; \frac{1}{v}\,e^{-av^2}\int_0^\infty du\,e^{-au^2}G(e/u)\,\sinh(2av\,u) \tag{6.5}$$

$$Q_2(a,e,v) \;=\; e^{-av^2}\int_0^\infty du\,e^{-au^2}G(e/u)\,\frac{1}{u}\left(\cosh(2av\,u) - \frac{\sinh(2av\,u)}{2av\,u}\right)\;, \tag{6.6}$$

with

$$G(y) \;=\; \ln(y) - \mathrm{Re}\,\psi(1+iy)\;. \tag{6.7}$$

For Gaussian quadrature, and for deriving a useful analytic result based on a saddle point approximation (to be described more fully in a moment), it is convenient to change variables to $x = u/v$. We can then can write the integrals in terms of two parameters,

$$r \;=\; e/v \tag{6.8}$$

$$s \;=\; av^2\;, \tag{6.9}$$

as follows:

$$Q_1(r,s) \;=\; e^{-s}\int_0^\infty dx\,e^{-sx^2}G(r/x)\,\sinh(2sx) \tag{6.10}$$

$$Q_2(r,s) \;=\; e^{-s}\int_0^\infty dx\,e^{-sx^2}G(r/x)\,\frac{1}{x}\left(\cosh(2sx) - \frac{\sinh(2sx)}{2sx}\right)\;. \tag{6.11}$$

We can also expand the hyperbolic functions and write

$$Q_1(r,s) \;=\; \frac{1}{2}\int_0^\infty dx\,G(r/x)\,\left[e^{-s(x-1)^2} - e^{-s(x+1)^2}\right] \tag{6.12}$$

$$Q_2(r,s) \;=\; \frac{1}{2}\int_0^\infty dx\,G(r/x)\,\frac{1}{x}\left[e^{-s(x-1)^2} + e^{-s(x+1)^2} - \frac{e^{-s(x-1)^2} - e^{-s(x+1)^2}}{2s\,x}\right]\;, \tag{6.13}$$

which illustrates that the integrand peaks as $x = 1$, or at $u = v$ in terms of the old variable, with a width of order $1/\sqrt{s}$.

For large and small values of s we can perform analytic approximations; however, in the most general case we must resort to a numerical integration scheme. As previously noted, the integrand d_dedxq(r,s,rm,rr,x) is peaked about $x = 1$ with a width of order $1/\sqrt{\rm s}$, and we can therefore restrict our numerical integration between the limits $1 \pm N/\sqrt{\rm s}$, taking a large value such as $N = 30$ for safety. If the lower integration limit becomes negative with this prescription we set it to zero. Omitting most of the variable declarations, the Gaussian quadrature routine is:

```
 FUNCTION dedxqi(v, a, e, rm, rr)
!
! This function performs the integration numerically by
! Gaussian Quadrature. The polynomial P3(x)=(5*x^3-3*x)/2
! is employed, and I have defined the appropriate weights
! W13, W2 and relative position UPM in parameter statements.
!
   IMPLICIT NONE

   REAL, INTENT(IN) :: v, a, e, rm, rr
   REAL :: dedxqi

   REAL,    PARAMETER :: UPM=0.7745966692E0
   REAL,    PARAMETER :: W13=0.5555555556E0, W2=0.8888888889E0
   INTEGER, PARAMETER :: NG=10000  ! must be even
   REAL,    PARAMETER :: NN=30.E0
...
   Special analytic limits will be placed here
...
   r=e/v
   s=a*v*v

   x0=1.E0 - NN/SQRT(s)
   x0=MAX(0.,x0)
   x1=1.E0 + NN/SQRT(s)
   dx=(x1-x0)/NG
   dedxqi=0.E0
   x=x0-dx
   DO ix=1,NG,2
!
      x=x+2.E0*dx
      dedxqi=dedxqi+W2*d_dedxq(r,s,rm,rr,x)
!
      xm=x-dx*UPM
      dedxqi=dedxqi+W13*d_dedxq(r,s,rm,rr,xm)
!
      xm=x+dx*UPM
      dedxqi=dedxqi+W13*d_dedxq(r,s,rm,rr,xm)
   ENDDO
   dedxqi=dedxqi*dx
 END FUNCTION dedxqi
```

The integrand d_dedxq is

```
 FUNCTION d_dedxq(r, s, rm, rr, x)
   IMPLICIT NONE
   REAL, INTENT(IN) :: r, s, rm, rr, x
   REAL, PARAMETER  :: SXMAX=0.05
   REAL :: d_dedxq
   REAL :: repsi, rx, sx, sh, ch
   REAL :: ep, em, xm1, xp1
   rx=r/x
   sx=2*s*x
   xm1=x-1
   xp1=x+1
   ep=EXP(-s*xp1*xp1)
   em=EXP(-s*xm1*xm1)
```

```
   sh=0.5E0*(em-ep)          ! sh and ch are
   IF (sx .GT. SXMAX) THEN   ! not sinh or cosh
      ch=0.5E0*(em+ep)       !
      ch=(ch - sh/sx)/x
   ELSE
      ch=2.E0*sx/3 + (1.E0/15.E0 - 1.E0/(6.E0*s))*sx*sx*sx
      ch=s*ch*EXP(-s)
   ENDIF
   d_dedxq=LOG(ABS(rx)) - repsi(rx)
   d_dedxq=d_dedxq*(rm*ch - rr*sh)
 END FUNCTION d_dedxq
```

The function repsi(x) will be coded in a moment. Note that if the argument sx $\equiv 2sx$ is smaller than SXMAX $= 0.05$ then we expand

$$\frac{1}{2x}\left[e^{-s(x-1)^2}+e^{-s(x+1)^2}-\frac{e^{-s(x-1)^2}-e^{-s(x+1)^2}}{2s\,x}\right]=s\,e^{-s}\left[\frac{2}{3}\,\mathrm{sx}+\left(\frac{1}{15}-\frac{1}{6s}\right)\mathrm{sx}^3\right] \quad (6.14)$$

B. The Large-s Limit

We can approximate the $s \gg 1$ limit analytically by a saddle-point evaluation of the x-integral. For the value of the parameter a passed to the subroutine dedxqi, this limit corresponds to

$$s=\frac{1}{2}\,\beta_b m_b v_p^2=\frac{r}{2}\,\frac{\beta_b}{\beta_m}\,\frac{m_b}{m}\,\bar{v}_p^2\gg 1\ . \quad (6.15)$$

The dimensionless projectile velocity $\bar{v}_p$ is in units of the thermal velocity $v_{\rm th}$ defined in (2.2), for some reference species of mass m and inverse temperature β_m. We will always take the reference species to be the electron. The value $r = 3$ is the default, in which case $v_{\rm th}$ correspond to the thermal root-mean-squared velocity of the electron. Each plasma component will usually have a different value for s, with the difference being especially acute between electrons and ions because of the mass ratio m_b/m (which is one for electrons and several thousand for ions). We see that the large-s limit encompasses several physical regimes, and is typically realized when: (i) the projectile velocity v_p is much greater than the thermal velocity of the electron $v_{\rm th}$, so that $\bar{v}_p = v_p/v_{\rm th}$ is large, (ii) the temperature T_b of species b is much less than the electron temperature T_m, so that β_b/β_m is large, and (iii) when the species b is an ion, so that m_b/m is large. For the case of ions, the value of s can be of order 10^3 or more. However, as the ion velocity decreases, the value of s decreases as the square of the velocity, so s need not always be large for ions.

Let us now look at the saddle-point approximation to the integrals Q_1 and Q_2 in the case

of large s. We will start with Q_1 and evaluate the integral for $s \gg 1$:

$$Q_1(r,s) \;=\; \frac{1}{2}\int_0^\infty dx\, G(r/x)\,\left[e^{-s(x-1)^2} - e^{-s(x+1)^2}\right] \tag{6.16}$$

$$\;=\; \frac{1}{2}\int_{-1}^\infty dx\, G\left(\frac{r}{x+1}\right)\,\left[e^{-sx^2} - e^{-s(x+2)^2}\right] \qquad :\; x \to x+1 \tag{6.17}$$

$$\;\approx\; \frac{1}{2}\int_{-1}^\infty dx\, G\left(\frac{r}{x+1}\right)\, e^{-sx^2} \qquad : \text{dropping the second exponential} \tag{6.18}$$

$$\;=\; \frac{1}{2}\int_{-1}^\infty dx\,\left[G_0(r) + G_1(r)\,x + \frac{1}{2}G_2(r)\,x^2 + \cdots\right] e^{-sx^2} \;: \text{expanding } G \text{ in } \; x \tag{6.19}$$

$$\;\approx\; \frac{1}{2}\int_{-\infty}^\infty dx\,\left[G_0(r) + G_1(r)\,x + \frac{1}{2}G_2(r)\,x^2 + \cdots\right] e^{-sx^2} \;: \text{extending lower int limit.}$$

The expansion coefficients are x-independent and given by

$$G_n(r) \;=\; \frac{d^n}{dx^n}\, G\left(\frac{r}{x+1}\right)\Bigg|_{x=0} \qquad \Rightarrow \tag{6.20}$$

$$G_0(r) \;=\; G(r) \tag{6.21}$$

$$G_1(r) \;=\; -rG'(r) \tag{6.22}$$

$$G_2(r) \;=\; 2rG'(r) + r^2G''(r)\;, \tag{6.23}$$

with

$$G(y) \;=\; \ln(y) - \mathrm{Re}\,\psi(1+iy) \equiv \ln(y) - \mathrm{repsi}(y) \tag{6.24}$$

$$G'(r) \;=\; \frac{1}{y} + \mathrm{Im}\,\psi'(1+iy) \equiv \frac{1}{y} - \mathrm{repsi1}(y) \tag{6.25}$$

$$G''(r) \;=\; -\frac{1}{y^2} + \mathrm{Re}\,\psi''(1+iy) \equiv -\frac{1}{y^2} - \mathrm{repsi2}(y)\;. \tag{6.26}$$

For numerical use we will construct fits to the functions G, G' and G'', or actually to repsi, repsi1 and repsi2, but first let's continue with the integral, which can now be performed exactly using,

$$\int_{-\infty}^\infty dx e^{-sx^2} = \sqrt{\frac{\pi}{s}} \qquad \int_{-\infty}^\infty dx\, x\, e^{-sx^2} = 0 \qquad \int_{-\infty}^\infty dx\, x^2\, e^{-sx^2} = \frac{\sqrt{\pi}}{2s^{3/2}}\;, \tag{6.27}$$

thereby giving

$$Q_1(r,s) \;\approx\; \frac{\sqrt{\pi}}{2}\left[\frac{G_0(r)}{s^{1/2}} + \frac{G_2(r)}{4s^{3/2}}\right]\;. \tag{6.28}$$

We can evaluate the integral Q_2 in a similar fashion,

$$Q_2(r,s) \approx \frac{1}{2}\int_{-\infty}^{\infty} dx\, G\left(\frac{r}{x+1}\right)\left[\frac{1}{x+1} - \frac{1}{2s\,(x+1)^2}\right] e^{-sx^2} \tag{6.29}$$

$$= \frac{1}{2}\int_{-\infty}^{\infty} dx \left[H_0(r,s) + H_1(r,s)\,x + \frac{1}{2}\,H_2(r,s)\,x^2 + \cdots\right] e^{-sx^2}\ , \tag{6.30}$$

where the expansion coefficients are

$$H_n(r,s) = \frac{d^n}{dx^n}\,G\!\left(\frac{r}{x+1}\right)\left[\frac{1}{x+1} - \frac{1}{2s\,(x+1)^2}\right]\Bigg|_{x=0} \quad \Rightarrow \tag{6.31}$$

$$H_0(r,s) = G_0(r)\left(1 - \frac{1}{2s}\right) \tag{6.32}$$

$$H_1(r,s) = G_1(r)\left(1 - \frac{1}{2s}\right) - G_0(r)\left(1 - \frac{1}{s}\right) \tag{6.33}$$

$$H_2(r,s) = G_2(r)\left(1 - \frac{1}{2s}\right) - 2\,G_1(r)\left(1 - \frac{1}{s}\right) + 2\,G_0(r)\left(1 - \frac{3}{2s}\right)\ , \tag{6.34}$$

and the integral therefore becomes

$$Q_2(r,s) \approx \frac{\sqrt{\pi}}{2}\left[\frac{H_0(r,s)}{s^{1/2}} + \frac{H_2(r,s)}{4s^{3/2}}\right]\ . \tag{6.35}$$

In summary, for large s the quantum integral can be written as

$$s \gg 1: \tag{6.36}$$

$$I_{\rm QM} \equiv \mathrm{dedxqi(a,e,v,rm,rr)} = \mathrm{rm}\,Q_2(\mathrm{r,s}) - \mathrm{rr}\,Q_1(\mathrm{r,s})$$

$$\mathrm{r = e/v \text{ and } s = av^2}\ ,$$

with an accuracy of about 0.2% for $s > s_{\max} = 10$. The corresponding subroutine is listed below, with most of the variable declarations omitted for brevity.

```
FUNCTION dedxqi(v, a, e, rm, rr)
  IMPLICIT NONE
  REAL, INTENT(IN) :: v, a, e, rm, rr
  REAL :: dedxqi
  ...
  REAL,    PARAMETER :: SQPI =1.772453851    ! sqrt(pi)
  REAL,    PARAMETER :: SMAX=10.             ! cut on s
  ...

  r=e/v
  s=a*v*v
  IF ( s .GT. SMAX) THEN          ! large s can be performed analytically
```

```
      g0 =LOG(ABS(r)) - repsi(r)   ! this case is usually realized for ions
      dg =1/r - repsi1(r)          !
      ddg=-1/(r*r) - repsi2(r)     !
      g1=-r*dg                     !
      g2=2*r*dg + r*r*ddg          !
      s1=1/s                       ! s1=1/s
      s2=s1/2                      ! s2=1/2*s
      s3=3*s2                      ! s3=3/2*s
      s05=SQRT(s)                  ! s05=s^(1/2)
      s15=s*s05                    ! s15=s^(3/2)
      h0=g0*(1-s2)
      h2=g2*(1-s2) - 2*g1*(1-s1) + 2*g0*(1-s3)
      q1=SQPI/2
      q2=SQPI/2
      q1=q1*(g0/s05 + 0.25E0*g2/s15)
      q2=q2*(h0/s05 + 0.25E0*h2/s15)
      dedxqi=rm*q2 - rr*q1
   ELSE                            ! otherwise do integral numerically
      ...
      Previously listed Gaussian quadrature method here
      ...
   ENDIF
 END FUNCTION dedxqi
```

This subroutine calls the three special functions repsi(x) = $\mathrm{Re}\,\psi(1+ix)$, repsi1(x) = $-\mathrm{Im}\,\psi'(1+ix)$, and repsi2($x$) = $-\mathrm{Re}\,\psi''(1+ix)$. The polygamma functions are usually not part of the repertoire of standard mathematical functions accompanying Fortran, so we will construct these functions by finding analytic expressions for large and small x and by using fits to polynomials and other functions at intermediate values. Rather than using canned functions from a mathematical library, constructing the functions we need ourselves will make the code more portable.

1. Fits to the Digamma Function and its Derivatives

We collect here a number of useful results for finding expansions of the polygamma functions in the limit of large and small arguments. These results are taken from Ref. [3], Chapter 6 on pp 259,260. In finding the small-x expansion of $\psi(1+ix)$ and its derivatives with respect to x we use:

$$\psi^{(n)}(1) \;=\; (-1)^{n+1}\, n!\, \zeta(n+1) \quad n = 1, 2, 3, \cdots \qquad : 6.4.2 \;, \tag{6.37}$$

and $\psi(1) = -\gamma$, where $\gamma = 0.57721\cdots$ is the Euler constant. For a general complex argument z we can Taylor expand

$$\psi^{(n)}(1+z) \;=\; \sum_{m=0}^{\infty} \frac{1}{m!}\, \psi^{(n+m)}(1)\, z^m = \sum_{m=0}^{\infty} \frac{1}{m!}\, (-1)^{n+m+1}\, (n+m)!\, \zeta(n+m+1)\, z^m \tag{6.38}$$

so that

$$\psi^{(n)}(1+ix) \;=\; \sum_{m=0}^{\infty} \frac{1}{m!}\,(-1)^{n+m+1}\,(n+m)!\,\zeta(n+m+1)\,(ix)^m \;. \tag{6.39}$$

For small x, the series can be truncated at the appropriate order to give the desired accuracy.

To find the large-argument expansion we first use

$$\psi^{(n)}(1+z) \;=\; \psi^{(n)}(z) + (-1)^n\,n!\,z^{-(1+n)} \quad n=0,1,2,\cdots \quad : 6.4.6 \;, \tag{6.40}$$

and then employ the asymptotically large limits

$$|z|\to\infty \text{ with } \arg(z)<\pi :$$

$$\psi(z) \;=\; \ln z - \frac{1}{2z} - \frac{1}{12z^2} + \frac{1}{120z^4} - \frac{1}{252z^6} + \mathcal{O}(z^{-8}) \qquad : 6.3.18 \tag{6.41}$$

$$\psi^{(1)}(z) \;=\; \frac{1}{z} + \frac{1}{2z^2} + \frac{1}{6z^3} - \frac{1}{30z^5} + \frac{1}{42z^7} - \frac{1}{30z^9} + \mathcal{O}(z^{-11}) \qquad : 6.4.12 \tag{6.42}$$

$$\psi^{(2)}(z) \;=\; -\frac{1}{z^2} - \frac{1}{z^3} - \frac{1}{2z^4} + \frac{1}{6z^6} - \frac{1}{6z^8} + \frac{3}{10z^{10}} - \frac{5}{6z^{12}} + \mathcal{O}(z^{-14}) \quad : 6.4.13 \;. \tag{6.43}$$

For the case of interest, when the argument becomes $z=ix$, this gives

$$\begin{aligned}\psi(1+ix) \;&=\; \psi(ix) + (ix)^{-1} \\ &=\; -\frac{i}{2x} + \frac{i\pi}{2} + \ln x + \frac{1}{12x^2} + \frac{1}{120x^4} + \frac{1}{252x^6} + \mathcal{O}(x^{-8})\end{aligned} \tag{6.44}$$

$$\begin{aligned}\psi'(1+ix) \;&=\; \psi'(ix) - (ix)^{-2} \\ &=\; \frac{1}{2x^2} - \frac{i}{x} + \frac{i}{6x^3} + \frac{i}{30x^5} + \frac{i}{42x^7} + \frac{i}{30x^9} + \mathcal{O}(x^{-11})\end{aligned} \tag{6.45}$$

$$\begin{aligned}\psi''(1+ix) \;&=\; \psi''(ix) + 2(ix)^{-3} \\ &=\; \frac{i}{x^3} + \frac{1}{x^2} - \frac{1}{2x^4} - \frac{1}{6x^6} - \frac{1}{6x^8} - \frac{3}{10x^{10}} - \frac{5}{6x^{12}} + \mathcal{O}(x^{-14}) \;.\end{aligned} \tag{6.46}$$

It is sufficient to keep only the first three terms in each expansion, and so for large x we have

$$\text{repsi}(x) = \quad \text{Re}\,\psi(1+ix) \;=\; \ln x + \frac{1}{12x^2} + \frac{1}{120x^4} + \mathcal{O}(x^{-6}) \tag{6.47}$$

$$\text{repsi1}(x) = -\text{Im}\,\psi'(1+ix) \;=\; \frac{1}{x} - \frac{1}{6x^3} - \frac{1}{30x^5} - \mathcal{O}(x^{-7}) \tag{6.48}$$

$$\text{repsi2}(x) = -\text{Re}\,\psi''(1+ix) \;=\; -\frac{1}{x^2} + \frac{1}{2x^4} + \frac{1}{6x^6} + \mathcal{O}(x^{-8}) \;. \tag{6.49}$$

2. The Digamma Function: repsi

The integrand calls the function repsi$(x) \equiv \mathrm{Re}\,\psi(1+ix)$. The asymptotic limits of repsi(x) for large and small x are

$$x < x_{\min} = 0.16: \qquad \mathrm{repsi}(x) \;=\; -\gamma + \zeta(3)x^2 + \mathcal{O}(x^4) \qquad \text{max error } 0.1\% \quad (6.50)$$

$$x > x_{\max} = 1.5: \qquad \mathrm{repsi}(x) \;=\; \ln x + \frac{1}{12x^2} + \frac{1}{120\,x^4} + \mathcal{O}(x^{-6}) \qquad \text{max error } 0.1\% \quad (6.51)$$

For intermediate values between $x_{\min}$ and $x_{\max}$ we use the fit

$$\mathrm{repsi}(x) = -\gamma + \frac{1}{2}\ln\left(1 + \frac{e^{2\gamma}\,x^4 + 2\zeta(3)x^2}{(1+x^2)}\right)\left[1 - \frac{1}{10}\exp\left(-\frac{4x}{3} - \frac{9}{8x}\right)\right]^{-1} . \quad (6.52)$$

which has an maximum error of 0.1% around $x \sim 0.5$ and $x \sim 2$. The numerical values of the constants are $\gamma = 0.5772156649$, $\zeta(3) = 1.202056903$, and $e^{2\gamma} = 3.172218958$. The function repsi$(x)$ is now listed:

```
FUNCTION repsi(x)
  USE globalvars
  IMPLICIT NONE
  REAL, INTENT(IN) :: x
  REAL, PARAMETER :: XMIN=0.16E0, XMAX=1.5E0
  REAL, PARAMETER :: TZETA3=2.404113806E0 ! 2*ZETA(3)
  REAL, PARAMETER :: A=0.1E0, B=1.33333E0, C=1.125E0
  REAL :: repsi
  REAL :: x2, x4
  IF (x .LE. XMIN) THEN
     x2=x**2
     repsi=-GAMMA + ZETA3*x2
  ELSEIF (x .GE. XMAX) THEN
     x2=x**2
     x4=x2*x2
     repsi=LOG(x)+1.D0/(12.D0*x2)+1.D0/(120.D0*x4)
  ELSE
     x2=x*x
     repsi=0.5E0*LOG(1 + (EXP2E*x2*x2 + TZETA3*x2)/(1+x2))
     repsi=repsi/(1 - A*EXP(-B*x - C/x)) - GAMMA
  ENDIF
END FUNCTION repsi
```

3. The Derivative: repsi1

The integrand also calls the function repsi1$(x) = \frac{d}{dx}\,\mathrm{Re}\,\psi(1+ix) = -\mathrm{Im}\,\psi'(1+ix)$, which has the asymptotic limits

$$x < x_{\min} = 0.14: \qquad \mathrm{repsi1}(x) \;=\; 2\zeta(3)\,x - 4\zeta(5)\,x^3 \qquad \text{max error } 0.1\% \tag{6.53}$$
$$2\zeta(3) \;=\; 2.404113806319188 \qquad 4\zeta(5) = 4.14771102057348$$

$$x > x_{\max} = 1.9: \qquad \mathrm{repsi1}(x) \;=\; \frac{1}{x} - \frac{1}{6x^3} - \frac{1}{30x^5} + \mathcal{O}(x^{-7}) \qquad \text{max error } 0.08\%\ . \tag{6.54}$$

In contrast to the previous case, the asymptotic limits are polynomials, and at intermediate values of x between $x_{\min}$ and $x_{\max}$ we can therefore perform polynomial fits. The function repsi1(x) is always positive except at at $x = 0$ where it vanishes; it then increases to a maximum of about repsi1 ~ 0.9 around $x \sim 0.7$; and then decreases asymptotically to zero as x increases further. We could easily fit the function between the entire region $x_{\min} < x < x_{\max}$ with a single polynomial; however, I found it more convenient to fit between $x_{\min} < x < x_1$ and $x_1 < x < x_{\max}$, with $x_1 = 0.7$, using two separate polynomials. In the first region $x_{\min} < x < x_1$ the function increases monotonically, and in the second region $x_1 < x < x_{\max}$ it decreases monotonically, and therefore upon using separate polynomials in each region we can obtain more accuracy with a lower order. Least-square fits to 5^{th} order give

$$x_{\min} < x < x_1 = 0.7: \quad \mathrm{repsi1}(x) \;=\; \sum_{\ell=0}^{5} a_\ell\, x^\ell \tag{6.55}$$

$$0.7 = x_1 < x < x_{\max}: \quad \mathrm{repsi1}(x) \;=\; \sum_{\ell=0}^{5} b_\ell\, x^\ell\ , \tag{6.56}$$

with the coefficients given by

ℓ	a_ℓ	b_ℓ
0	0.004211521868683916	−0.25386287337370820
1	2.314767988469241000	4.60092985583543200
2	0.761843932767193200	−6.76154044407838200
3	−7.498711815965575000	4.46723854889984100
4	7.940030433629257000	−1.44439009761387350
5	2.749533936429732000	0.18595402917922707

(6.57)

See the Mathematica notebook IQM.nb for details. The full subroutine for repsi1(x) becomes

```
!               d
! repsi1(x) = --- Re[ Psi(1 + I*x) = -Im Psi'(1 + I*x).
!             dx
 FUNCTION repsi1(x)
```

```
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: repsi1

   REAL, PARAMETER :: XMIN=0.14E0, X1=0.7E0 ,XMAX=1.9E0
   REAL, PARAMETER :: ZETA32=2.404113806319188 ! 2*ZETA(3)
   REAL, PARAMETER :: ZETA54=4.147711020573480 ! 4*ZETA(5)
   REAL, PARAMETER :: a0= 0.004211521868683916
   REAL, PARAMETER :: a1= 2.314767988469241000
   REAL, PARAMETER :: a2= 0.761843932767193200
   REAL, PARAMETER :: a3=-7.498711815965575000
   REAL, PARAMETER :: a4= 7.940030433629257000
   REAL, PARAMETER :: a5=-2.749533936429732000
   REAL, PARAMETER :: b0=-0.253862873373708200
   REAL, PARAMETER :: b1= 4.600929855835432000
   REAL, PARAMETER :: b2=-6.761540444078382000
   REAL, PARAMETER :: b3= 4.467238548899841000
   REAL, PARAMETER :: b4=-1.444390097613873500
   REAL, PARAMETER :: b5= 0.185954029179227070
   REAL :: xi
   IF ( x .LE. XMIN) THEN              ! x < xmin=0.14
      repsi1=ZETA32*x - ZETA54*x*x*x   ! accurate to 0.1%
   ELSEIF (x .LE. x1) THEN
      repsi1=a5                        ! xmin < x < x1=0.7
      repsi1=a4 + repsi1*x             ! accurate to 0.002%
      repsi1=a3 + repsi1*x             ! a0 + a1*x + a2*x^2 +
      repsi1=a2 + repsi1*x             ! a3* x^3 + a4*x^4 + a5*x^5
      repsi1=a1 + repsi1*x
      repsi1=a0 + repsi1*x
   ELSEIF (x .LE. xmax) THEN           ! x1 < x < xmax=1.9
      repsi1=b5                        ! accurate to 0.1%
      repsi1=b4+repsi1*x               ! b0 + b1*x + b2*x^2 +
      repsi1=b3+repsi1*x               ! b3*x^3 + b4*x^4 +
      repsi1=b2+repsi1*x               ! b5*x^5
      repsi1=b1+repsi1*x
      repsi1=b0+repsi1*x
   ELSE
      xi=1/x                           ! x > xmax=1.9
      repsi1=-1.E0/30.E0               ! accurate to 0.08%
      repsi1=repsi1*xi                 ! 1/x - 1/6x^3 - 1/30x^5
      repsi1=-1.E0/6.D0 + repsi1*xi    !
      repsi1=repsi1*xi                 !
      repsi1=1.E0 + repsi1*xi          !
      repsi1=repsi1*xi                 !
   ENDIF
 END FUNCTION repsi1
```

Note the manner in which the polynomials have been calculated. One might be tempted to define the values of the a_ℓ in an array a(0:5) and to write

```
 repsi1=0
 DO i=0,5
   repsi1 = repsi1 + a(i)*(x**i)
 ENDDO                          ;
```

however, it takes far fewer operations to construct the polynomial backward in the fashion

```
repsi1=0
DO i=5,0,-1
  repsi1 = a(i) + repsi1*x
ENDDO                              ;
```

or, as I have chosen to implement this procedure,

```
repsi1=b5
repsi1=b4 + repsi*x
repsi1=b3 + repsi*x
repsi1=b2 + repsi*x
repsi1=b1 + repsi*x
repsi1=b0 + repsi*x    .
```

In the latter there are only ten operations, five additions and five multiplications. For an N^{th} order polynomial this method would only require $2N$ operations. In the former there are $\ell + 2$ operations for each loop $\ell = 0, \cdots, 5$ (there are ℓ multiplications for x^ℓ, another multiplication for $a_\ell \cdot x^\ell$, and a single addition), giving a total of 27 operations. Again, for an N^{th} order polynomial the former method would require $\sum_{\ell=0}^{N}(\ell+2) = (N+1)(N+4)/2$ operations. The more obvious former method is quadratic while the latter method is only linear.

4. The Second Derivative: repsi2

The final polygamma function is $\text{repsi2}(x) = \frac{d^2}{dx^2}\,\text{Re}\,\psi(1+ix) = -\text{Re}\,\psi''(1+ix)$, which has the asymptotic forms

$$x < x_{\min} = 0.18 : \tag{6.58}$$

$$\begin{aligned} \text{repsi2}(x) &= 2\zeta(3) - 12\zeta(5)\,x^2 + 30\zeta(7)\,x^4 \qquad \text{max error } 0.1\% \\ 2\zeta(3) &= 2.404113806319188 \qquad 12\zeta(5) = 12.44313306172044 \\ 30\zeta(7) &= 30.250478321457685 \end{aligned}$$

$$x > x_{\max} = 2.5 :$$

$$\text{repsi2}(x) = -\frac{1}{x^2} + \frac{1}{2x^4} + \frac{1}{6x^6} + \mathcal{O}(x^{-8}) \qquad \text{max error } 0.1\% \ . \tag{6.59}$$

As before, it is convenient to divide the interval $x_{\min} < x < x_{\max}$ into the two regions $x_{\min} < x < x_1$ and $x_1 < x < x_{\max}$, this time with $x_1 = 1.2$. In the first interval the function repsi2(x) drops monotonically from a maximum of repsi2 ~ 2.4 at $x = 0$ to a minimum of repsi2 ~ -0.4 at $x \sim 1.2$ (the function crosses zero at $x = 0.678157$). In the second interval beyond $x \sim 1.2$, the function remains negative but begins to increase and asymptotically approaches the x-axis from below as x gets large. This is the motivation for dividing the

interval around $x_1 = 1.2$ and performing separate least-square fits in the two subintervals. Working to 7^{th} and 6^{th} orders for the first and second intervals gives

$$x_{\min} < x < x_1 = 1.2: \quad \text{repsi1}(x) = \sum_{\ell=0}^{7} a_\ell \, x^\ell \tag{6.60}$$

$$1.2 = x_1 < x < x_{\max}: \quad \text{repsi1}(x) = \sum_{\ell=0}^{6} b_\ell \, x^\ell \ , \tag{6.61}$$

with the coefficients

ℓ	a_ℓ	b_ℓ
0	2.42013533575662130	4.98436272402513600
1	−0.41115258967949336	−16.65464665310595300
2	−8.09116694062588400	20.69413003620411000
3	−24.93648245588276400	−13.37268378509369200
4	114.81090561524718000	4.83094787289278800
5	−170.85454523278196000	−0.92976482601030100
6	128.84024667658247000	0.07456475055097825
7	−50.24590900103020600	

(6.62)

The full subroutine for repsi2(x) is listed below.

```
!              d^2
! repsi2(x) = ---- Re[ Psi(1 + I*x) = -Re Psi''(1 + I*x).
!             dx^2
 FUNCTION repsi2(x)  ! needs to be completed
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: repsi2

   REAL, PARAMETER :: XMIN=0.18E0, X1=1.2E0 ,XMAX=2.5E0
   REAL, PARAMETER :: ZETA32=2.4041138063191880 ! 2*ZETA(3)
   REAL, PARAMETER :: ZETA512=12.44313306172044 ! 12*ZETA(5)
   REAL, PARAMETER :: ZETA730=30.250478321457685! 30*ZETA(7)
   REAL, PARAMETER :: a0= 2.42013533575662130
   REAL, PARAMETER :: a1=-0.41115258967949336
   REAL, PARAMETER :: a2=-8.09116694062588400
   REAL, PARAMETER :: a3=-24.9364824558827640
   REAL, PARAMETER :: a4=114.8109056152471800
   REAL, PARAMETER :: a5=-170.854545232781960
   REAL, PARAMETER :: a6=128.8402466765824700
   REAL, PARAMETER :: a7=-50.2459090010302060
   REAL, PARAMETER :: a8= 8.09941032385266400
   REAL, PARAMETER :: b0= 4.98436272402513600
   REAL, PARAMETER :: b1=-16.6546466531059530
   REAL, PARAMETER :: b2= 20.6941300362041100
   REAL, PARAMETER :: b3=-13.3726837850936920
   REAL, PARAMETER :: b4= 4.83094787289278800
   REAL, PARAMETER :: b5=-0.92976482601030100
   REAL, PARAMETER :: b6= 0.07456475055097825
   REAL :: xi, xx
   IF ( x .LE. XMIN) THEN
```

```
      xx=x*x
      repsi2=ZETA32 - ZETA512*xx + ZETA730*xx*xx ! x < xmin=0.18
   ELSEIF (x .LE. x1) THEN                       ! accurate to 0.1%
      repsi2=a8                                  !
      repsi2=a7 + repsi2*x                       ! xmin < x < x1=1.2
      repsi2=a6 + repsi2*x                       ! accurate to 0.01%
      repsi2=a5 + repsi2*x                       ! a0 + a1*x + a2*x^2 +
      repsi2=a4 + repsi2*x                       ! a3*x^3 + a4*x^4 +
      repsi2=a3 + repsi2*x                       ! a5*x^5 + a6*x^6 +
      repsi2=a2 + repsi2*x                       ! a7*x&7 + a8*x^8
      repsi2=a1 + repsi2*x
      repsi2=a0 + repsi2*x
   ELSEIF (x .LE. xmax) THEN                     ! x1 < x < xmax=2.5
      repsi2=b6                                  ! accurate to 0.2%
      repsi2=b5+repsi2*x                         ! b0 + b1*x + b2*x^2 +
      repsi2=b4+repsi2*x                         ! b3*x^3 + b4*x^4 +
      repsi2=b3+repsi2*x                         ! b5*x^5 + b6*x^6
      repsi2=b2+repsi2*x
      repsi2=b1+repsi2*x
      repsi2=b0+repsi2*x
   ELSE
      xi=1/x                                     ! x > xmax=2.5
      xi=xi*xi                                   ! accurate to 0.07%
      repsi2= 1.E0/6.E0                          ! -1/x^2 + 1/2x^4 +
      repsi2= 0.5E0 + repsi2*xi                  ! 1/6x^6
      repsi2=-1. + repsi2*xi                     !
      repsi2=repsi2*xi
   ENDIF
END FUNCTION repsi2
```

C. The Small-s Limit

We can also analytically approximate the integrals Q_1 and Q_2 defined in (VI A) in the $s \ll 1$ limit. In this case it is convenient to change the integration variable to $y = \sqrt{a}\,u$, and to define the parameters

$$r = \sqrt{a}\,e \tag{6.63}$$

$$s = a\,v^2 \quad \text{(as before)} , \tag{6.64}$$

which allows us to express the integrals as

$$Q_1(r,s) = \frac{1}{\sqrt{s}}\,e^{-s}\int_0^\infty dy\,e^{-y^2}G(r/y)\,\sinh\left(2\sqrt{s}\,y\right) \tag{6.65}$$

$$Q_2(r,s) = e^{-s}\int_0^\infty dy\,e^{-y^2}G(r/y)\,\frac{1}{y}\left(\cosh\left(2\sqrt{s}\,y\right) - \frac{\sinh\left(2\sqrt{s}\,y\right)}{2\sqrt{s}\,y}\right) . \tag{6.66}$$

We will keep the parameter r general, and expand the hyperbolic functions for small arguments,

$$\sinh x = x + \frac{x^3}{6} + \cdots \tag{6.67}$$

$$\frac{1}{x}\left(\cosh x - \frac{\sinh x}{x}\right) = \frac{x}{3} + \frac{x^3}{30} + \cdots . \tag{6.68}$$

This gives

$$Q_1(r,s) = \frac{1}{\sqrt{s}}\,e^{-s}\int_0^\infty dy\,e^{-y^2}G(r/y)\,\left[(2\sqrt{s}\,y) + \frac{1}{6}\,\left(2\sqrt{s}\,y\right)^3\right] \tag{6.69}$$

$$Q_2(r,s) = 2\sqrt{s}\,e^{-s}\int_0^\infty dy\,e^{-y^2}G(r/y)\,\left[\frac{1}{3}\,(2\sqrt{s}\,y) + \frac{1}{30}\,\left(2\sqrt{s}\,y\right)^3\right] , \tag{6.70}$$

or more succinctly

$$Q_1(r,s) = e^{-s}\left[2K_1(r) + \frac{4\,s}{3}\,K_3(r)\right] \tag{6.71}$$

$$Q_2(r,s) = e^{-s}\left[\frac{4\,s}{3}\,K_1(r) + \frac{8\,s^2}{15}\,K_3(r)\right] , \tag{6.72}$$

with

$$K_1(x) = \int_0^\infty dy\,y\,e^{-y^2}G(x/y) \tag{6.73}$$

$$K_3(x) = \int_0^\infty dy\,y^3\,e^{-y^2}G(x/y) . \tag{6.74}$$

In this way we have reduced the 2-parameter integrals $Q_1(r,s)$ and $Q_2(r,s)$ to 1-parameter integrals $K_1(x)$ and $K_3(x)$, which we can now fit numerically (and find their analytic asymptotic limits for large and small x). The quantum integral for small-s becomes,

$$s \ll 1 : \tag{6.75}$$

$$I_{\rm QM} \equiv \mathrm{dedxqi(a,e,v,rm,rr)} \;=\; e^{-s}\left[\left(\frac{4\,\mathrm{s}}{3}\,\mathrm{rm}-2\,\mathrm{rr}\right)K_1(\mathrm{r})+\left(\frac{8\,\mathrm{s}^2}{15}\,\mathrm{rm}-\frac{4\,\mathrm{s}}{3}\,\mathrm{rr}\right)K_3(\mathrm{r})\right]$$

$$\mathrm{r}=\sqrt{\mathrm{a}}\,\mathrm{v}\ \text{and}\ \mathrm{s}=\mathrm{av}^2\ ,$$

with an accuracy between 0.2% and 0.5% for $s < s_{\rm min} = 0.05$ (as r varies from 0.1 to 10 or so the accuracy decreases slightly). The relevant source code is listed below:

```
FUNCTION dedxqi(v, a, e, rm, rr)
  IMPLICIT NONE

  REAL, INTENT(IN) :: v, a, e, rm, rr
  REAL :: dedxqi
  ...
  REAL,    PARAMETER :: SMAX=10., SMIN=0.05  ! cuts on s
  ...
  s=a*v*v
  IF ( s .GT. SMAX) THEN
    ...
     Previously listed large-s method here
    ...
  ELSEIF ( s .LT. SMIN) THEN      ! small s can be performed analytically
     r=SQRT(a)*e                  ! this case is usually realized for elec.
     i1=kqm1(r)                   ! accuracy < 0.5%
     i2=kqm3(r)
     dedxqi=(4.*rm*s/3. - 2*rr)*i1
     dedxqi=dedxqi+ 4.*(4*rm*s*s/15.- rr*s/3.)*i2
     dedxqi=EXP(-s)*dedxqi
  ELSE
     ...
     Previously listed Gaussian quadrature method here
     ...
  ENDIF
END FUNCTION dedxqi
```

The functions $K_1(x)$ and $K_3(x)$ have been called kqm1 and kqm3 in the above subroutine.

1. Fitting the Functions K_1 and K_3

We now code the functions $K_1(x)$ and $K_2(x)$ by analytically calculating the large and small x-limits, *i.e.* for $x < x_{\rm min}$ and $x > x_{\rm max}$ where the values $x_{\rm min}$ and $x_{\rm max}$ will chosen to give an accuracy of order a tenth of a percent or so, and then by interpolating between these limits with least-squares fits.

We will first concentrate on the $x > x_{\rm max}$ regime, as this turns out to be the easier case, and we consider the functions

$$\begin{aligned} K_n(x) \;&=\; \int_0^\infty dy\, y^n\, e^{-y^2} G(x/y) \\ \text{with } G(x) \;&=\; \ln x - \mathrm{Re}\,\psi(1+ix) \end{aligned} \tag{6.76}$$

for positive integral values of n. In the end we are only interested in the cases $n = 1$ and 3; however, for clarity and to simplify the intermediate expressions we will work with a general K_n for now. The exponent in (6.76) ensures that the integrand is negligible unless $y \lesssim 1$, and so the argument of $G(x/y)$ remains large when x is large. We can expand $G(x/y)$ using (6.44), which for $x > x_{\rm max}$ (where $x_{\rm xmax}$ is to be determined) allows us to write

$$G(x/y) = -\frac{y^2}{12x^2} - \frac{y^4}{120\,x^4} - \frac{y^6}{252\,x^6} + \mathcal{O}(x^{-8})\ , \tag{6.77}$$

and therefore

$$K_n(x) \;=\; -\int_0^\infty dy\, e^{-y^2} \left[\frac{y^{2+n}}{12x^2} + \frac{y^{4+n}}{120\,x^4} + \frac{y^{6+n}}{252\,x^4} + \right] + \mathcal{O}(x^{-8})\ . \tag{6.78}$$

For $n = 1$ and 3 we need to evaluate the integrals,

$$\int_0^\infty dy\, y^3\, e^{-y^2} = \frac{1}{2} \qquad \int_0^\infty dy\, y^5\, e^{-y^2} = 1 \qquad \int_0^\infty dy\, y^7\, e^{-y^2} = 3 \qquad \int_0^\infty dy\, y^9\, e^{-y^2} = 12 \tag{6.79}$$

which gives

$$\begin{aligned} &x > x_{\rm max} = 3.2: \\ &K_1(x) \;=\; -\frac{1}{24\,x^2} - \frac{1}{120\,x^4} - \frac{1}{84\,x^6} \qquad \text{max error } 0.1\% \end{aligned} \tag{6.80}$$

$$\begin{aligned} &x > x_{\rm max} = 2.5: \\ &K_3(x) \;=\; -\frac{1}{12\,x^2} - \frac{1}{40\,x^4} - \frac{3}{64\,x^6} \qquad \text{max error } 0.1\% \ . \end{aligned} \tag{6.81}$$

By working to such a high order we are able to obtain an accuracy of less than 0.1% for reasonably small values of $x_{\rm max}$ of order one.

Let us now find an analytic approximation to $K_n(x)$ for $x < x_{\rm min}$, where we will choose $x_{\rm min}$ to give an accuracy of about 0.1%. This limit is more involved than the previous case. One might be tempted to use the small-x limit (6.50) for repsi(x) and to write

$$K_n(x) \;=\; \int_0^\infty dy\, y^n\, e^{-y^2} \left[\, \ln(x/y) + \gamma - \zeta(3)(x/y)^2 + \cdots \,\right]\ . \tag{6.82}$$

Indeed, for $n \geq 2$ this is a reasonably good approximation, and for the case of interest we find

$$K_3(x) = \int_0^\infty dy\, y^3\, e^{-y^2} \left[\ln(x/y) + \gamma - \zeta(3)(x/y)^2 + \cdots \right] \tag{6.83}$$

$$= (\ln x + \gamma) \underbrace{\int_0^\infty dy\, y^3\, e^{-y^2}}_{1/2} - \underbrace{\int_0^\infty dy\, y^3\, \ln y\, e^{-y^2}}_{(1-\gamma)/4} - \zeta(3)x^2 \underbrace{\int_0^\infty dy\, y\, e^{-y^2}}_{1/2} + \cdots \tag{6.84}$$

$$= \frac{1}{2} \ln x + \frac{3\gamma}{4} - \frac{1}{4} - \frac{1}{2}\zeta(3)x^2 + \cdots \ . \tag{6.85}$$

This expansion is accurate to 0.1% for $x < x_{\rm min} = 0.15$. The problem arises when one uses (6.50) to approximate K_1:

$$K_1(x) = \int_0^\infty dy\, y\, e^{-y^2} \left[\ln(x/y) + \gamma - \zeta(3)(x/y)^2 \cdots \right] \tag{6.86}$$

$$= (\ln x + \gamma) \underbrace{\int_0^\infty dy\, y\, e^{-y^2}}_{1/2} - \underbrace{\int_0^\infty dy\, y \ln y\, e^{-y^2}}_{-\gamma/4} - \zeta(3)x^2 \underbrace{\int_0^\infty \frac{dy}{y}\, e^{-y^2}}_{\text{log divergence at } y=0} + \cdots \tag{6.87}$$

$$= \frac{1}{2} \ln x + \frac{3\gamma}{4} + \cdots \ . \tag{6.88}$$

Ignoring the divergence and using only the first two terms to approximate K_1 gives an accuracy of 3.5% for $x \sim 0.1$, and the accuracy improves to only 2% for $x \sim 0.01$. For many applications, this might be good enough; however, we have consistently achieved an accuracy of 0.1%, so we must achieve that figure of merit here as well. The problem, of course, is that the quadratic term containing $\zeta(3)x^2$ is proportional to y^{-2}. Combined with the single factor of y in K_1, this term leads to the logarithmic singularity at $y = 0$. A singularity arises only for the $n = 1$ case, as the integral of y^{n-2} converges at $y = 0$ for $n \geq 2$. To find K_1, we must therefore be more careful in expanding $G(x/y)$. The full expansion of the digamma function gives

$$\mathrm{Re}\, \psi(1 + iz) = -\gamma + \sum_{k=1}^{\infty} \frac{1}{k} \frac{z^2}{k^2 + z^2} \ . \tag{6.89}$$

We have already calculated the contribution from the first term. We found that trouble arose when, for small arguments z, we approximated the sum by

$$\sum_{k=1}^{\infty} \frac{1}{k} \frac{z^2}{k^2 + z^2} \approx \sum_{k=1}^{\infty} \frac{z^2}{k^3} = \zeta(3)\, z^2 \ . \tag{6.90}$$

Inside the integral we must use this approximation with extreme care, since $z = x/y$ becomes large as $y \to 0$ regardless of how small we take x. We can make the sum look similar to (6.90) by writing

$$\sum_{k=1}^{\infty} \frac{1}{k} \frac{z^2}{k^2 + z^2} = \sum_{k=1}^{\infty} \frac{x^2}{k^3} \frac{d}{dy^2} \ln\left(k^2 y^2/x^2 + 1\right) \ . \tag{6.91}$$

The contribution from the sum, which we must add to (6.88), can therefore be written

$$\bar{K}_1(x) \equiv -\int_0^\infty dy\, y\, e^{-y^2} \sum_{k=1}^\infty \frac{1}{k}\,\frac{x^2}{k^2+x^2} = -\sum_{k=1}^\infty \frac{x^2}{k^3}\int_0^\infty dy\, y\, e^{-y^2}\frac{d}{dy^2}\ln\left(k^2y^2/x^2+1\right) \tag{6.92}$$

$$= -\frac{1}{2}\sum_{k=1}^\infty \frac{x^2}{k^3}\int_0^\infty du\, e^{-u}\frac{d}{du}\ln\left(k^2u/x^2+1\right) \tag{6.93}$$

$$= -\frac{1}{2}\sum_{k=1}^\infty \frac{x^2}{k^3}\int_0^\infty du\, e^{-u}\ln\left(k^2u/x^2+1\right)\ . \tag{6.94}$$

In the last equality we have integrated by parts,

$$\int_0^\infty du\, e^{-u}\frac{d}{du}\ln\left(k^2u/x^2+1\right) = e^{-u}\ln\left(k^2u/x^2+1\right)\Big|_{u=0}^\infty + \int_0^\infty du\, e^{-u}\ln\left(k^2u/x^2+1\right)$$

$$= \int_0^\infty du\, e^{-u}\ln\left(k^2u/x^2+1\right)\ . \tag{6.95}$$

The previous log-divergence at $u=0$ (or $y=0$) is now regulated by the 1 inside the logarithm, which consequently vanishes at $u=0$. Completing the integral is now straightforward:

$$\bar{K}_1(x) = -\frac{1}{2}\sum_{k=1}^\infty \frac{x^2}{k^3}\int_0^\infty du\, e^{-u}\left[\ln\left(\frac{u}{x^2}\right) + \underbrace{\ln\left(k^2+\frac{x^2}{u}\right)}_{k\text{ dep. isolated here}}\right] \tag{6.96}$$

$$= x^2\ln x \underbrace{\sum_{k=1}^\infty \frac{1}{k^3}}_{\zeta(3)}\underbrace{\int_0^\infty du\ e^{-u}}_{1} - \frac{x^2}{2}\underbrace{\sum_{k=1}^\infty \frac{1}{k^3}}_{\zeta(3)}\underbrace{\int_0^\infty du\ \ln u\, e^{-u}}_{-\gamma} \tag{6.97}$$

$$-x^2\underbrace{\sum_{k=1}^\infty \frac{\ln k}{k^3}}_{-\zeta'(3)}\underbrace{\int_0^\infty du\, e^{-u}}_{1} - \underbrace{\frac{x^2}{2}\sum_{k=1}^\infty \frac{1}{k^3}\int_0^\infty du\, e^{-u}\ln\left(1+\frac{x^2}{uk^2}\right)}_{\text{remainder}}$$

$$= \zeta(3)\,x^2\ln x + \left[\frac{1}{2}\zeta(3)\,\gamma + \zeta'(3)\right]x^2 + \text{remainder}\ . \tag{6.98}$$

We have now found both the leading-log term ($x^2\ln x$) and the sub-leading contribution (the x^2 piece). In other words, we have found the complete contribution of the form $Ax^2\ln x + Bx^2 = A\,x^2\ln(Cx)$, which amounts to finding the constant in front of the log ($A=\zeta(3)$ in this case) *and* the constant C inside the log (or $B = A\ln C = \zeta'(3) + \zeta(3)\gamma/2$). For small enough x, which will turn out to be of order a tenth, the remainder term can be neglected and still provide an accuracy of about 0.1%. Finally, one comment on $\zeta'(3) = -0.198126$ is in order. This is the derivative of the zeta-function $\zeta(s)$ evaluated at $s=3$. Normally one

thinks of the zeta-function as being defined at integer values of s, but one can analytically continue to the complex s-plane since the series

$$\zeta(s) = \sum_{k=1}^{\infty} \frac{1}{k^s} \tag{6.99}$$

converges for $\mathrm{Re}\, s > 1$. We can then take the derivative of the zeta-function:

$$\zeta'(s) = \sum_{k=1}^{\infty} \frac{d}{ds}\, k^{-s} = -\sum_{k=1}^{\infty} \frac{\ln k}{k^s}\ . \tag{6.100}$$

In summary, we have derived the small-x limits

$$x < x_{\rm min} = 0.15: \qquad K_1(x) \;=\; \frac{1}{2}\,\ln\!x + \frac{3\gamma}{4} + \zeta(3)\,x^2 \ln x \;+\; \left[\zeta'(3) + \frac{1}{2}\,\zeta(3)\gamma\right] x^2 \qquad \text{max error } 0.2\% \tag{6.101}$$

$$x < x_{\rm min} = 0.15: \qquad K_3(x) \;=\; \frac{1}{2}\,\ln\!x + \frac{3\gamma}{4} - \frac{1}{4} - \frac{1}{2}\,\zeta(3)\,x^2 \qquad \text{max error } 0.1\% \tag{6.102}$$

The numerical value of the quadratic term in K_1 is $\zeta'(3) + \zeta(3)\gamma/2 = 0.148797$.

At intermediate values we perform least-squares fits to the following functional forms,

$$0.15 = x_{\rm min} < x < x_{\rm max} = 3.2: \qquad K_1(x) \;=\; \sum_{\ell=0}^{4} c_\ell\, x^\ell + \sum_{\ell=1}^{4} \frac{d_\ell}{x^\ell} + d_0 \ln\!x \qquad \text{max error } 0.2\% \tag{6.103}$$

$$0.15 = x_{\rm min} < x < x_{\rm max} = 2.5: \qquad K_3(x) \;=\; \sum_{\ell=0}^{4} c_\ell\, x^\ell + \sum_{\ell=1}^{4} \frac{d_\ell}{x^\ell} + d_0 \ln\!x \qquad \text{max error } 0.04\%\ , \tag{6.104}$$

giving the coefficients

K_1	ℓ	c_ℓ	d_ℓ
	0	0.25109815055856566000	−0.18373957827052560000
	1	−0.02989283169766254700	−0.33121125339783110000
	2	0.03339880139150325000	0.04022076263527408400
	3	−0.00799128953390392700	−0.00331897950305779480
	4	0.00070251863606810650	0.00012313574797356784

(6.105)

and

K_1	ℓ	c_ℓ	d_ℓ
	0	0.691191700599840900000	0.835543536679762600000
	1	−1.094849572900974000000	0.047821976622976340000
	2	0.318264617154601400000	0.000053594881446931025
	3	−0.060275957444801354000	−0.000268040997573199600
	4	0.005112428730167831000	0.000015765134162582942

(6.106)

The source code for these functions, called kqm1 and kqm3 in Fortran, is now listed:

```
!
!             /Infinity
!             |
! kqm1(x)  =  | dy   y exp(-y^2) [ln(x/y) - repsi(x/y)]
!             |
!            /0
!
!
 FUNCTION kqm1(x)
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: kqm1

   REAL, PARAMETER :: XMIN=0.15E0, XMAX=3.2E0

   REAL, PARAMETER :: a0= 0.43291174867614964545494494 29 ! 3*GAMMA/4
   REAL, PARAMETER :: a1= 1.20205690315959428542504315 61 ! ZETA(3)
   REAL, PARAMETER :: a2= 0.14879679441773450264109933 31 ! ZETA'(3)+GAMMA*
   REAL, PARAMETER :: b2=-0.04166666666666666666666666 67 !-1/24  ZETA(3)/2
   REAL, PARAMETER :: b4=-0.00833333333333333333333333 33 !-1/120
   REAL, PARAMETER :: b6=-0.01190476190476190476190476 19 !-1/84
   REAL, PARAMETER :: c0= 0.25109815055856566000
   REAL, PARAMETER :: c1=-0.02989283169766254700
   REAL, PARAMETER :: c2= 0.03339880139150325000
   REAL, PARAMETER :: c3=-0.00799128953390392700
   REAL, PARAMETER :: c4= 0.00070251863606810650
   REAL, PARAMETER :: d0=-0.18373957827052560000
   REAL, PARAMETER :: d1=-0.33121125339783110000
   REAL, PARAMETER :: d2= 0.04022076263527408400
   REAL, PARAMETER :: d3=-0.00331897950305779480
   REAL, PARAMETER :: d4= 0.00012313574797356784
   REAL :: x2, lx, xi
   IF ( x .LE. XMIN) THEN                          ! x < xmin=0.15: to 0.06%
      x2=x*x                                       ! ln(x)/2 + 3*GAMMA/4 +
      lx=LOG(x)                                    ! ZETA(3)*X^2*ln(x) +
      kqm1=0.5E0*lx + a0 + a1*x2*lx + a2*x2        ! [ZETA'(3) + GAMMA*
                                                   ! ZETA(3)/2]*x^2
                                                   !
   ELSEIF ( x .GE. XMAX ) then                     ! x > xmax=3.2: to 0.12%
      xi=1/x                                       ! -1/24*x^2 - 1/120*x^4 -
      x2=xi*xi                                     ! 1/84*x^6
      kqm1=b6                                      !
      kqm1=b4 + kqm1*x2                            !
      kqm1=b2 + kqm1*x2                            !
      kqm1=kqm1*x2                                 !
   ELSE
      xi=1/x                                       ! xmin < x < xmax
      lx=LOG(x)                                    ! fit accurate to 0.2%
      kqm1=c4                                      ! c0 + c1*x + c2*x^2 +
      kqm1=c3+kqm1*x                               ! c3*x^3 + c4*x^4 +
      kqm1=c2+kqm1*x                               ! d0*ln(x) +
      kqm1=c1+kqm1*x                               ! d1/x + d2/x^2 +
      kqm1=c0+kqm1*x + d0*lx                       ! d3/x^3 + d4/x^4
      lx=d4                                        !
      lx=d3+lx*xi                                  !
      lx=d2+lx*xi                                  !
      lx=d1+lx*xi                                  !
```

```
      lx=lx*xi                                          !
      kqm1=kqm1+lx
   ENDIF
 END FUNCTION kqm1

!
!             /Infinity
!             |
! kqm3(x)  =  | dy   y^3 exp(-y^2) [ln(x/y) - repsi(x/y)]
!             |
!            /0
!
!
 FUNCTION kqm3(x)
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: kqm3

   REAL, PARAMETER :: XMIN=0.15E0, XMAX=2.5E0
   REAL, PARAMETER :: a0= 0.18291174867614964545494494 29 ! 3*GAMMA/4 - 1/4
   REAL, PARAMETER :: a2=-0.60102845157979714270733281 02 !-ZETA(3)/2
   REAL, PARAMETER :: b2=-0.08333333333333333333333333 33 !-1/12
   REAL, PARAMETER :: b4=-0.025                            !-1/40
   REAL, PARAMETER :: b6=-0.046875                         !-3/64
   REAL, PARAMETER :: c0= 0.691191700599840900000
   REAL, PARAMETER :: c1=-1.094849572900974000000
   REAL, PARAMETER :: c2= 0.318264617154601400000
   REAL, PARAMETER :: c3=-0.060275957444801354000
   REAL, PARAMETER :: c4= 0.005112428730167831000
   REAL, PARAMETER :: d0= 0.835543536679762600000
   REAL, PARAMETER :: d1= 0.047821976622976340000
   REAL, PARAMETER :: d2= 0.000053594881446931025
   REAL, PARAMETER :: d3=-0.000268040997573199600
   REAL, PARAMETER :: d4= 0.000015765134162582942
   REAL :: x2, lx, xi
   IF ( x .LE. XMIN) THEN                          ! x < xmin=0.15: to 0.1%
      x2=x*x                                       ! ln(x)/2 + 3*GAMMA/4 -1/4
      lx=LOG(x)                                    ! -[ZETA(3)/2]*x^2
      kqm3=0.5E0*lx + a0 + a2*x2                   !
                                                   !
                                                   !
   ELSEIF ( x .GE. XMAX ) then                     ! x > xmax=2.5: to 0.25%
      xi=1/x                                       ! -1/12*x^2 - 1/40*x^4 -
      x2=xi*xi                                     ! 3/64*x^6
      kqm3=b6                                      !
      kqm3=b4 + kqm3*x2                            !
      kqm3=b2 + kqm3*x2                            !
      kqm3=kqm3*x2                                 !
   ELSE
      xi=1/x                                       ! xmin < x , xmax
      lx=LOG(x)                                    ! fit accurate to 0.04%
      kqm3=c4                                      ! c0 + c1*x + c2*x^2 +
      kqm3=c3+kqm3*x                               ! c3*x^3 + c4*x^4 +
      kqm3=c2+kqm3*x                               ! d0*ln(x) +
      kqm3=c1+kqm3*x                               ! d1/x + d2/x^2 +
      kqm3=c0+kqm3*x + d0*lx                       ! d3/x^3 + d4/x^4
      lx=d4                                        !
```

```
      lx=d3+lx*xi                                    !
      lx=d2+lx*xi                                    !
      lx=d1+lx*xi                                    !
      lx=lx*xi                                       !
      kqm3=kqm3+lx
   ENDIF
 END FUNCTION kqm3
```

VII. THE CLASSICAL CONTRIBUTION

We now turn to the classical contribution

$$\frac{dE_b^{\rm C}}{dx} = \underbrace{\frac{c_1}{\bar{v}_p}}_{\rm cp1/vp} \cdot \underbrace{\kappa_b^2 E_b}_{\rm kb2*eb} \cdot \underbrace{I_1(A_b\bar{v}_p,\, B_b\bar{v}_p^2,\, C_b)}_{\rm intone} + \underbrace{c_2}_{\rm cp2} \cdot \underbrace{I_2(\bar{v}_p)}_{\rm inttwo} - \underbrace{\frac{c_3}{r_b\bar{v}_p^2}}_{\rm cp3/vp2*rb} \cdot \underbrace{\bar{H}_b(\bar{v}_p)}_{\rm hi} , \quad (7.1)$$

in which

$$I_1(a,b,c) = \int_0^1 du\, e^{-a^2u} \left\{ \frac{2}{\sqrt{u}} + \left[c - \ln\left(\frac{u}{1-u}\right)\right] \left[b\sqrt{u} - \frac{1}{\sqrt{u}}\right] \right\}$$

$$I_2(a) = \int_{-1}^{+1} du\, u\, \bar{H}_b(a\,u) \; . \quad (7.2)$$

The general structure of the classical routine dedxc is

dedx.f90:

```
SUBROUTINE dedxc(vp, dedxctot, dedxcsumi)
  USE globalvars
  ...
  ...
  A. loop over plasma species ib=1,NNB (ib=1 is electron)
  B. compute classical stopping power
     1. small velocity limit
     2. or numerical evalulation
        a. compute intone (I1)
        b. compute inttwo (I2)
        c. compute hi
  ...
  ...
END SUBROUTINE dedxc
```

or in complete form

dedx.f90:

```
SUBROUTINE dedxc(vp, dedxctot, dedxcsumi)
  USE globalvars

  IMPLICIT NONE
  REAL, INTENT(IN) :: vp
  REAL, INTENT(OUT):: dedxctot, dedxcsumi

  REAL, PARAMETER        :: ABV20=0.001
  REAL, DIMENSION(1:NNB) :: abv, abv2, clb
  REAL                   :: abv2ib, ss, vp2, kd2, kd4
  REAL                   :: hi, inttwo, intone, a, b, c, ke
  INTEGER                :: ib
!
! define input variables
!
```

```
   vp2=vp*vp
   abv=ab*vp      ! for intone, inttwo, hi
   clb=0          ! initialize classical to zero
   DO ib=1,NNB    ! loop over plasma components
      IF ( lzb(ib) ) THEN ! compute only if zb(ib) /= 0
         abv2=abv*abv                !
         abv2ib=abv2(ib)             ! cut on each component
         IF (abv2ib .LT. ABV20) THEN  ! small velocity limit is analytic
            kd2 =kd*kd
            kd4 =kd2*kd2
            clb(ib)=clb(ib)+2*gb(ib)*(1-abv2ib*(1 + &
                 (2./3.)*(mp0/mb0(ib))))
            clb(ib)=clb(ib)-(4./3.)*abv2ib -2*SUM(kb2*abv2)/kd2
            ss=(SUM(kb2*abv))
            ss=ss*ss
            clb(ib)=clb(ib)+(PI/6.)*ss/kd4
            clb(ib)=clb(ib)*cp1*kb2(ib)*eb(ib)/vp
         ELSE                  ! general velocities are numerical
!
! int1: dedxc=(cp1/vp)*intone(abv,bbv,kb2,eb,cb)
!
            a=abv(ib)
            b=bb(ib)*vp2
            c=cb(ib)
            ke=kb2(ib)*eb(ib)
            clb(ib)=clb(ib)+(cp1/vp)*ke*intone(a,b,c)
!
! int2: dedxc=dedxc + cp2*inttwo(abv,kb2)
!
            clb(ib)=clb(ib)+cp2*inttwo(ib,abv)
!
! H: dedxc=dedxc - (cp3/vp2)*h(abv,kb2)
!
            clb(ib)=clb(ib)-(cp3/vp2)*hi(ib,abv)/rb(ib)
         ENDIF
      ELSE
         clb(ib)=0  ! don't compute when zb(ib) = 0
      ENDIF
   ENDDO
   dedxcsumi=SUM(clb(2:NNB))
   dedxctot =clb(1)
   dedxctot =dedxctot+dedxcsumi

   dedxctot =CONVFACT*dedxctot   ! convert to MeV/mu-m
   dedxcsumi=CONVFACT*dedxcsumi  !
 END SUBROUTINE dedxc
```

We will look at each of these processes in reverse order, starting with the function hi, then inttwo, and finally intone.

A. Contribution H_b

The function hi(ib, abv) calculates $\bar{H}_b(\bar{v}_p)$ with $b =$ ib, where we have defined

$$\bar{H}_b(\bar{v}_p) = i\,\frac{\bar{\rho}_b(\bar{v}_p)}{\bar{\rho}_{\rm total}(\bar{v}_p)}\left[\bar{F}(\bar{v}_p)\ln\left(\frac{\bar{F}(\bar{v}_p)}{K^2}\right) - \bar{F}^*(\bar{v}_p)\ln\left(\frac{\bar{F}^*(\bar{v}_p)}{K^2}\right)\right], \tag{7.3}$$

with the real and imaginary parts of F being

$$\bar{F}_{\rm R}(\bar{v}_p) \;=\; \sum_b \kappa_b^2\Big[1 - 2x_b\,{\rm daw}\,(x_b)\Big] \qquad x_b = A_b\bar{v}_p \tag{7.4}$$

$$\bar{F}_{\rm I}(\bar{v}_p) \;=\; \sqrt{\pi}\sum_b \kappa_b^2\,x_b\,e^{-x_b^2} = \pi\bar{\rho}_{\rm total}(\bar{v}_p)\;. \tag{7.5}$$

Let us first look at the subroutine fri(xb, fr, fi, fabs, farg) that returns the necessary parts of $\bar{F}$. The arguments are described by

variable name	description
xb	$x_b = A_b\bar{v}_p$ array
fr	F_r array
fi	F_i array
fabs	$\lvert F\rvert = \sqrt{F_r^2 + F_i^2}$ array
farg	array of angles between real and imag

and the subroutine itself is

dedx.f90:

```
SUBROUTINE fri(xb, fr, fi, fabs, farg)
  USE globalvars
  IMPLICIT NONE
  REAL, DIMENSION(1:NNB), INTENT(IN)  :: xb
  REAL,                   INTENT(OUT) :: fr,  fi, fabs, farg
  REAL    :: x, daw, d
  INTEGER :: ib
  fr=0
  fi=0
  DO ib=1,NNB
     x=xb(ib)
     d=daw(x)
     fr=fr+(kb2(ib)*(1-2*x*d))
     fi=fi+kb2(ib)*x*EXP(-x*x)
  ENDDO
  fi=fi*SQPI
  fabs=SQRT(fr*fr + fi*fi)
  farg=ATAN2(fi,fr)
END SUBROUTINE fri
```

To calculate the function hi(ib, abv), we write (7.3) as

$$\bar{H}(\bar{v}_p) \equiv i\left[\bar{F}\ln(\bar{F}/K^2) - \bar{F}^*\ln(\bar{F}^*/K^2)\right] \tag{7.6}$$

$$\bar{H}_b(\bar{v}_p) = \pi\bar{H}(\bar{v}_p)\frac{\rho_b(x_b)}{\bar{F}_{\rm I}(\bar{v}_p)} = \pi\bar{H}(\bar{v}_p)\frac{\kappa_b^2 x_b e^{-x_b^2}/\sqrt{\pi}}{\sqrt{\pi}\sum_c \kappa_c^2\, x_c\, e^{-x_c^2}} \quad x_b = A_b\bar{v}_p\ . \tag{7.7}$$

The logarithm can be written

$$\begin{aligned}\ln\bar{F} &= \ln(\bar{F}_{\rm R} + i\bar{F}_{\rm I}) = \ln|\bar{F}|e^{i\arg\bar{F}} \\ &= \ln|\bar{F}| + i\arg\bar{F}\ .\end{aligned} \tag{7.8}$$

For the reflection properties to hold we must choose $-\pi < \arg\bar{F} \le \pi$. Upon expanding $\bar{H}$ we find

$$\begin{aligned}\bar{H} &= -2\,{\rm Im}\left\{\bar{F}\ln(\bar{F}/K^2)\right\} = -2\left(\bar{F}_{\rm R}\,{\rm Im}\ln(\bar{F}/K^2) + \bar{F}_{\rm I}\,{\rm Re}\ln(\bar{F}/K^2)\right) \\ &= -2\left[\,\bar{F}_{\rm R}\arg\bar{F} + \bar{F}_{\rm I}\,\ln(|\bar{F}|/K^2)\,\right]\end{aligned}$$

The subroutine hi now takes the form

dedx.f90:

```
 FUNCTION hi(ib, abv)
   USE globalvars
   IMPLICIT NONE
   REAL,    DIMENSION(1:NNB), INTENT(IN) :: abv
   INTEGER,                   INTENT(IN) :: ib
   REAL                                  :: hi
   REAL, PARAMETER :: EPS=1.E-6, XB2MAX=10.D0
   REAL    :: fr, fi, fabs, farg, h, ebc
   REAL    :: xb, xb2, xc, xc2, xbc
   INTEGER :: ic
   CALL fri(abv, fr, fi, fabs, farg)
   h=-2*(fi*LOG(fabs/K**2) + fr*farg)
   xb =abv(ib)
   xb2=xb*xb
!
! Express Fi in terms of its sum and write
!
!   hi = kb^2*xb*Exp[-xb^2]/Sum_c kc^2*xc*Exp[-xc^2]
!
! then calculate hi^-1 first, excluding kb^2*xb since
! some components of kb^2*xb might be zero.
   hi=0
   DO ic=1,NNB
      xc =abv(ic)
      xc2=xc*xc
      xbc=xb2-xc2
      IF (xbc .LT. XB2MAX) THEN
         ebc=EXP(xb2-xc2)
      ELSE
         hi=0    ! ebc is large and dominates the sum,
```

```
         RETURN  ! and its inverse is almost zero
      ENDIF
      hi=hi + kb2(ic)*xc*ebc
   ENDDO
   hi=kb2(ib)*xb/hi  ! invert and include kb^2*xb
   hi=h*hi
 END FUNCTION hi
```

In calculating (7.7) we must be careful about numerically multiplying powers of large and small exponents together. Numerically, we will find that

$$e^{-(\text{very large})} \times e^{+\text{large}} = e^{-(\text{very large})} \times \text{number} = 0 \quad \text{(to machine precision)} \ , \tag{7.9}$$

when in fact the product of the two exponentials should be a very small but nonzero number. To avoid this and obtain more accuracy, we shall first multiply the exponentials by hand,

$$e^{-(\text{very large})} \times e^{+\text{large}} = e^{-(\text{very large})+\text{large}} \ , \tag{7.10}$$

since the combined exponent will not be as large (in absolute value) as either one individually. In fact, we essentially calculate reciprocal

$$\bar{H}_b^{-1} \ = \ \sum_c \kappa_c^2 \, x_c \, e^{x_b^2 - x_c^2} \ \frac{1}{\kappa_b^2 x_b \, \bar{H}} \ , \tag{7.11}$$

and then take the reciprocal again.

B. Contribution I_2

$$I_2(a) \ = \ \int_{-1}^{+1} du \, u \, \bar{H}_b(a\,u) = 2 \int_0^1 du \, u \, \bar{H}_b(a\,u) \ . \tag{7.12}$$

dedx.f90:

```
 FUNCTION inttwo(ib, abv)
   USE globalvars

   IMPLICIT NONE
   REAL, DIMENSION(1:NNB), INTENT(IN) :: abv
   INTEGER,                INTENT(IN) :: ib
   REAL                               :: inttwo

   REAL, PARAMETER :: UPM=0.7745966692E0
   REAL, PARAMETER :: W13=0.5555555556E0, W2=0.8888888889E0

   INTEGER, PARAMETER :: NG=2000  ! NG must be even
   REAL,    PARAMETER :: U0=0.E0, U1=1.E0, DU=1.E0/NG

   REAL    :: u, um, hi, uu(NNB)
```

```
   INTEGER :: iu, ibb

   inttwo=0
   u=U0-DU
   DO iu=1,NG,2
!
      u=u+2.E0*DU
      DO ibb=1,NNB
         uu(ibb)=u*abv(ibb)
      ENDDO
      inttwo=inttwo+W2*u*hi(ib,uu)
!
      um=u-DU*UPM
      DO ibb=1,NNB
         uu(ibb)=um*abv(ibb)
      ENDDO
      inttwo=inttwo+W13*um*hi(ib,uu)
!
      um=u+DU*UPM
      DO ibb=1,NNB
         uu(ibb)=um*abv(ibb)
      ENDDO
      inttwo=inttwo+W13*um*hi(ib,uu)
   ENDDO
   inttwo=2*inttwo*DU
 END FUNCTION inttwo
```

C. Contribution I_1

Recall that the first classical contribution is

$$I_1(a,b,c) = \int_0^1 du\, e^{-a^2 u}\left\{\frac{2}{\sqrt{u}} + \left[c - \ln\left(\frac{u}{1-u}\right)\right]\left[b\sqrt{u} - \frac{1}{\sqrt{u}}\right]\right\} \tag{7.13}$$

$$= \sqrt{\pi}\,\mathrm{erf}(a)\left[\frac{2-c}{a} + \frac{bc}{2a^3}\right] - \frac{bc}{a^2}\,e^{-a^2} + J_3(a^2) - J_1(a^2) + b\Big[J_2(a^2) - J_4(a^2)\Big]\,, \tag{7.14}$$

where the functions $J_1 \cdots J_2$ are defined

$$J_1(x) \equiv \int_0^1 du\, e^{-xu}\, \frac{\ln(1-u)}{\sqrt{u}} \tag{7.15}$$

$$J_2(x) \equiv \int_0^1 du\, e^{-xu}\, \sqrt{u}\, \ln(1-u) \tag{7.16}$$

$$J_3(x) \equiv \int_0^1 du\, e^{-xu}\, \frac{\ln u}{\sqrt{u}} \tag{7.17}$$

$$J_4(x) \equiv \int_0^1 du\, e^{-xu}\, \sqrt{u}\, \ln u\ . \tag{7.18}$$

The source code is
dedx.f90:

```
FUNCTION intone(a, b, c)
  USE globalvars

  IMPLICIT NONE
  REAL, INTENT(IN) :: a, b, c
  REAL             :: intone

  REAL    :: bc, a2, a3, erfa, expa, ferf
  REAL    :: j1, j2, j3, j4
  bc=b*c
  a2=a*a
  a3=a2*a
  erfa=SQPI*ferf(a)  ! see ferf.f
  expa=EXP(-a2)
  intone=erfa*((2-c)/a + bc/(2*a3))-bc*expa/a2
  intone=intone + j3(a2) - j1(a2) + b*(j2(a2) -j4(a2))
END FUNCTION intone
```

For a detailed evaluation of the integrals and limits that follow, see intJ1234.nb. For the moment note that we have performed a few of the integrals exactly, namely

$$\int_0^1 du\, e^{-a^2u}\, \frac{1}{\sqrt{u}} \;=\; \frac{\sqrt{\pi}}{a}\, \mathrm{erf}(a) \tag{7.19}$$

$$\int_0^1 du\, e^{-a^2u}\, \sqrt{u} \;=\; \frac{\sqrt{\pi}}{2a^3}\, \mathrm{erf}(a) - \frac{e^{-a^2}}{a^2}\ , \tag{7.20}$$

Let us now look at the integrals $J_1 \cdots J_4$, which can be expressed in terms of hypergeometric functions

$${}_pF_q(\mathbf{a}, \mathbf{p}, z) \equiv \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_p)_k}{(b_1)_k \cdots (b_q)_k}\, \frac{z^k}{k!}\ , \tag{7.21}$$

where **a** and **b** are p and q dimensional vectors, respectively, and the Pochhammer symbol is

$$(s)_0 \equiv 1 \tag{7.22}$$

$$(s)_k \equiv s(s+1)\cdots(s+k-1)\ . \tag{7.23}$$

The integrals J_3 and J_4 are easily expressed as

$$J_3(a^2) = \int_0^1 du\, e^{-a^2 u}\, \frac{\ln u}{\sqrt{u}} = -4\ {}_2F_2(\mathbf{1}/2,\, \mathbf{3}/2, -a^2) \tag{7.24}$$

$$J_4(a^2) = \int_0^1 du\, e^{-a^2 u}\, \sqrt{u}\, \ln u = -\frac{4}{9}\ {}_2F_2(\mathbf{3}/2,\, \mathbf{5}/2, -a^2) \tag{7.25}$$

$$\mathbf{1} = (1,1) \quad \mathbf{3} = 3\cdot\mathbf{1} \quad \mathbf{5} = 5\cdot\mathbf{1}\ .$$

The integrals J_1 and J_2 are somewhat more complicated,

$$J_1(a^2) \equiv \int_0^1 du\, e^{-a^2 u}\, \frac{\ln(1-u)}{\sqrt{u}} = \sqrt{\pi}\left[-\gamma\, \frac{\mathrm{erf}(a)}{a} + {}_1\bar{F}_1(1/2,\, 3/2,\, -a^2)\right] \tag{7.26}$$

$$J_2(a^2) \equiv \int_0^1 du\, e^{-a^2 u}\, \sqrt{u}\, \ln(1-u) = \gamma\, \frac{e^{-a^2}}{a^2} + \frac{\sqrt{\pi}}{2a^2}\left[-\gamma \frac{\mathrm{erf}(a)}{a} + {}_1\bar{F}_1(1/2,\, 3/2, -a^2) \right. \tag{7.27}$$

$$\left. + {}_1\bar{F}_1(3/2,\, 3/2,\, -a^2)\right]\ ,$$

where $\gamma = 0.577216\cdots$ is the Euler gamma constant, and where the functions ${}_1\bar{F}_1(x,y,z)$ are defined to be the regularized hypergeometric functions ${}_1F_1(x,y,z)/\Gamma(y)$ with a derivative of the second argument:

$${}_1\bar{F}_1(x,y,z) \equiv \frac{\partial}{\partial y}\, \frac{{}_1F_1(x,y,z)}{\Gamma(y)}\ . \tag{7.28}$$

However, these forms for $J_1 \cdots J_4$ are not very useful numerically.

1. $J_1(x)$

We now consider the integral

$$J_1(x) \equiv \int_0^1 du\, e^{-x\,u}\, \frac{\ln(1-u)}{\sqrt{u}}\ , \tag{7.29}$$

where $x \geq 0$. We can obtain an analytic solutions for small and large values of the argument,

$$
\begin{aligned}
&x \ll 1: \\
&\qquad J_1(x) \;=\; mx + b \qquad\qquad m = \frac{4}{9}\,(4 - \ln 8) \quad b = -(4 - \ln 16) \\
&\qquad \text{error } 0.3\% \text{ for } x < x_{\min} = 0.1
\end{aligned}
\tag{7.30}
$$

$$
\begin{aligned}
&x \gg 1: \\
&\qquad J_1(x) \;=\; a\,x^{-3/2} + g\,x^{-5/2} \qquad a = -\frac{\sqrt{\pi}}{2} \quad g = -\frac{3\sqrt{\pi}}{8}\;. \\
&\qquad \text{error } 0.3\% \text{ for } x > x_{\max} = 20
\end{aligned}
\tag{7.31}
$$

In the former case, we expand the exponent to linear order in x, while in the latter case the support lies near $u \sim 1$ and we replace the upper limit by infinity and expand $\ln(1-u) = -u - u^2/2 + \mathcal{O}(u^3)$.

For intermediate points between $x_{\min}$ and $x_{\max}$ we will approximate the function by the product of rational functions

$$
J_1(x) = R(x)\,Q_6(x)\;. \tag{7.32}
$$

The function $R(x)$ is designed to capture the asymptotic behavior of J_1, and we take

$$
R(x) = \underbrace{\frac{mx+b}{c\,x^{7/2}+1}}_{x\ll 1:\; mx+b} + \underbrace{\frac{e\,a\,x^2}{e\,x^{7/2}+1}}_{x\gg 1:\; a\,x^{-3/2}}\;, \tag{7.33}
$$

where x^2 in the numerator of the second term is chosen so that this term dominates over the first term at large x. In summary, we take

coeff	numerical	exact
m	0.843565	$4(4-\ln 8)/9$
b	-1.22741	$-(4-\ln 16)$
c	0.1	arbitrary
e	0.2	arbitrary
a	-0.886227	$-\sqrt{\pi}/2$

(7.34)

For $n = 6$ we need $2n - 1 = 11$ data points: we will take m to start at zero and end at ten,

and we choose

m	x_m	$J_1(x_m)$
0	0.1	−1.14532
1	0.2	−1.06942
2	0.5	−0.874098
3	1.0	−0.633428
4	2.0	−0.35207
5	3.0	−0.211855
6	4.0	−0.137756
7	5.0	−0.0960144
8	6.0	−0.0709186
9	7.0	−0.0548658
10	8.0	−0.04401

(7.35)

and solving the linear equations gives

ℓ	b_ℓ	a_ℓ
0	−921.277	−921.277
1	781.631	774.26
2	−327.675	−211.002
3	39.2781	−1.1106
4	−0.278985	33.5015
5	−1.68781	−11.749

(7.36)

globalvars.f90:

```
! j1() approximates the j1 integral by rations
! functions with coefficients:
!
  INTEGER, PARAMETER         :: NNJ1=3, NMJ1=2*NNJ1-1 ! NMJ1=5
  REAL,    DIMENSION(0:NMJ1) :: J1B, J1A
  PARAMETER (           &
  J1B=(/                &
   -926.65E0  ,         & !b0
   787.016E0  ,         & !b1
   -329.764E0 ,         & !b2
   39.7406E0  ,         & !b3
   -0.173896E0,         & !b4
   -1.66913E0 /),       & !b5
  J1A=(/                &
    J1B(0),             & !a0
    787.165E0 ,         & !a1
    -213.584E0,         & !a2
    -1.04219E0,         & !a3
    33.594E0  ,         & !a4
   -11.8391E0/)         ) !a5
  REAL, PARAMETER :: J1MM= (4./9.)*(4.-LOG8)    ! 0.8535815
  REAL, PARAMETER :: J1BB=-(4.-LOG16)           !-1.2274113
  REAL, PARAMETER :: J1AA=-SQPI/2.              !-0.8862270
  REAL, PARAMETER :: J1CC= 0.1E0
  REAL, PARAMETER :: J1EE= 0.2E0
  REAL, PARAMETER :: J1GG=-3.*SQPI/8.
```

dedx.f90:

```
 FUNCTION j1(x)
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: j1

   REAL,    PARAMETER   :: XMIN=0.1D0, XMAX=20.D0
   REAL    :: x2, x4
   REAL    :: xx, ra, rc
   REAL    :: y, y3, y5
   INTEGER :: n
!
! analytic asymptotic forms
!
  IF (x .LE. XMIN) THEN
     j1=J1MM*x+J1BB
  ELSEIF (x .GT. XMAX) THEN
     y=SQRT(x)           ! x^1/2
     y3=x*y              ! x^3/2
     y5=y3*x             ! x^5/2
     j1=J1AA/y3 +J1GG/y5
  ELSE
!
! numerical asymptotic form
!
     x2=x*x
     x4=x**3.5
     j1=(J1MM*x+J1BB)/(J1CC*x4+1) + J1EE*J1AA*x2/(J1EE*x4+1)
!
! spline correction
!
     ra=0.E0
     rc=0.E0
     xx=1.E0
     DO n=0,NMJ1
        ra=ra+J1A(n)*xx
        rc=rc+J1B(n)*xx
        xx=x*xx
     ENDDO
     ra=ra+xx
     rc=rc+xx
     j1=j1*rc/ra
  ENDIF
 END FUNCTION j1
```

2. $J_2(a)$

We will now consider the integral

$$J_2(x) \equiv \int_0^1 du\, e^{-xu} \sqrt{u}\, \ln(1-u) \tag{7.37}$$

where $x \geq 0$. The analytic solutions at asymptotic values of x are

$$x \ll 1: \qquad J_2(x) \;=\; mx + b \qquad\qquad m = \frac{4}{75}\,(23 - 15\ln 2) \quad b = -\frac{4}{9}\,(4 - \ln 8) \tag{7.38}$$
$$\text{error } 0.3\% \text{ for } x < x_{\min} = 0.1$$

$$x \gg 1: \qquad J_2(x) \;=\; a\,x^{-5/2} + g\,x^{-7/2} \qquad a = -\frac{3\sqrt{\pi}}{4} \quad g = -\frac{15\sqrt{\pi}}{16}\;. \tag{7.39}$$
$$\text{error } 0.3\% \text{ for } x > x_{\min} = 30$$

As before, in the former case we expand the exponent to linear order in x, and in the latter case the support lies near $u \sim 1$ and we replace the upper limit by infinity and expand $\ln(1-u) = -u - u^2/2 + \mathcal{O}(u^2)$.

For intermediate values of x between $x_{\min}$ and $x_{\max}$ we approximate the integral by

$$J_2(x) = R(x)\,Q_6(x)\;, \tag{7.40}$$

where we take

$$R(x) = \underbrace{\frac{mx+b}{c\,x^{9/2}+1}}_{x\ll 1:\; mx+b} + \underbrace{\frac{e\,a\,x^2}{e\,x^{9/2}+1}}_{x\gg 1:\; a\,x^{-5/2}}\;, \tag{7.41}$$

with x^2 in the numerator of the second term being chosen so that this term dominates over the first term at large x. In summary, we take

coeff	numerical	exact
m	0.663779	$4(23-15\ln 2)/75$
b	-0.853582	$-4(4-\ln 8)/9$
c	0.5	arbitrary
e	0.2	arbitrary
a	-1.32934	$-3\sqrt{\pi}/4$

(7.42)

For $n = 6$ we need $2n - 1 = 11$ data points: we will take m to start at zero and end at ten, and we choose

m	x_m	$J_2(x_m)$
0	0.1	-0.789096
1	0.2	-0.729766
2	0.3	-0.675165
3	0.5	-0.578622
4	1.0	-0.396407
5	2.0	-0.192906
6	3.0	-0.0992046
7	4.0	-0.054271
8	5.0	-0.0316815
9	6.0	-0.0197116
10	7.0	-0.0130054

(7.43)

and solving the linear equations gives

ℓ	b_ℓ	a_ℓ
0	82.3208	82.3208
1	−263.406	−262.701
2	316.627	315.347
3	−176.8	−178.477
4	55.9024	58.7802
5	−8.50148	−9.99801

(7.44)

globalvars.f90:

```
! j2() approximates the j2 integral by rations
! functions with coefficients:
!
  INTEGER, PARAMETER         :: NNJ2=3, NMJ2=2*NNJ2-1 ! NMJ2=5
  REAL,    DIMENSION(0:NMJ2) :: J2B, J2A
  PARAMETER (           &
  J2B=(/                &
    87.1714E0 ,         & !b0
    -277.584E0,         & !b1
    329.082E0 ,         & !b2
    -180.982E0,         & !b3
    56.7202E0 ,         & !b4
    -8.60238E0/),       & !b5
   J2A=(/               &
    J2B(0),             & !a0
    -277.693E0,         & !a1
    329.801E0 ,         & !a2
    -184.219E0,         & !a3
    59.9325E0 ,         & !a4
    -10.1138E0/)        ) !a5
  REAL, PARAMETER ::  J2MM= 4.*(23.-15.*LOG2)/75. ! 0.6721489
  REAL, PARAMETER ::  J2BB=-4.*(4.-LOG8)/9.       !-0.8535815
  REAL, PARAMETER ::  J2AA=-3.*SQPI/4.            !-1.3293405
  REAL, PARAMETER ::  J2CC= 0.5E0
  REAL, PARAMETER ::  J2EE= 0.2E0
  REAL, PARAMETER ::  J2GG=-15.*SQPI/16.
```

dedx.f90:

```
 FUNCTION j2(x)
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: j2

   REAL,    PARAMETER   :: XMIN=0.1, XMAX=30.D0
   REAL    :: x2, x4
   REAL    :: y, y5, y7
   REAL    :: xx, ra, rc
   INTEGER :: n
!
! analytic asymptotic forms
!
  IF (x .LE. XMIN) THEN
     j2=J2MM*x+J2BB
```

```
  ELSEIF (x .GT. XMAX) THEN
     y=SQRT(x)           ! x^1/2
     y5=x*x*y            ! x^5/2
     y7=y5*x             ! x^7/2
     j2=J2AA/y5 +J2GG/y7
  ELSE
!
! numerical asymptotic form
!
     x2=x*x
     x4=x**4.5
     j2=(J2MM*x+J2BB)/(J2CC*x4+1) + &
        J2EE*J2AA*x2/(J2EE*x4+1)
!
! spline correction
!
     ra=0.E0
     rc=0.E0
     xx=1.E0
     DO n=0,NMJ2
        ra=ra+J2A(n)*xx
        rc=rc+J2B(n)*xx
        xx=x*xx
     ENDDO
     ra=ra+xx
     rc=rc+xx
     j2=j2*rc/ra
  ENDIF
 END FUNCTION j2
```

3. $J_3(x)$

We will now consider the integral

$$J_3(x) \equiv \int_0^1 du\, e^{-xu} \frac{\ln u}{\sqrt{u}} \tag{7.45}$$

where $x \geq 0$. We can obtain an analytic solutions for small and large values of the argument: $x \ll 1$ and $x \gg 1$. In the former case, we expand the exponent to linear order and find

$$x \ll 1 : \qquad J_3(x) = mx + b \qquad m = 4/9 \quad b = -4 \tag{7.46}$$

error 0.3% for $x < x_{\min} = 0.4$

$$x \gg 1 : \qquad J_3(x) = (a_1 \ln x + a_2) x^{-1/2} \qquad a_1 = -\sqrt{\pi} \quad a_2 = -\sqrt{\pi}\,(\gamma + \ln 4) \tag{7.47}$$

error 0.3% for $x > x_{\max} = 2.3$

The former case is obtained, as before, by expanding the exponent to linear order in x. In the latter case the support lies near $u \sim 1$ and we replace the upper limit by infinity and

write

$$J_3(x) \approx \int_0^\infty du\, e^{-xu}\, \frac{\ln u}{\sqrt{u}} = \int_0^\infty \frac{du}{x}\; e^{-u}\, \frac{\ln(u/x)}{\sqrt{(u/x)}} \quad : u \to u/x$$
$$= x^{-1/2} \int_0^\infty du\; e^{-u}\, \frac{\ln u - \ln x}{\sqrt{u}}\ , \tag{7.48}$$

which gives (7.47) after doing the integrals.

For intermediate values of x between $x_{\rm min}$ and $x_{\rm max}$, we approximate

$$J_3(x) = R(x)\, Q_6(x)\ , \tag{7.49}$$

where we take

$$R(x) = \underbrace{\frac{mx+b}{c\,x^{7/2}+1}}_{x\ll 1:\ mx+b} + \underbrace{\frac{e\,[a_1 \ln(x+1)+a_2]x^2}{e\,x^{5/2}+1}}_{x\gg 1:\ (a_1 \ln x + a_2)\, x^{-1/2}}\ , \tag{7.50}$$

with x^2 in the numerator of the second term chosen so that this term dominates over the first term at large x. In summary, we take

coeff	numerical	exact
m	0.444444	4/9
b	-4	-4
c	0.1	arbitrary
e	0.2	arbitrary
a_1	-1.77245	$-\sqrt{\pi}$
a_2	-3.48023	$-\sqrt{\pi}\,(\gamma + \ln 4)$

(7.51)

For $n = 6$ we need $2n - 1 = 11$ data points: we will take m to start at zero and end at ten, and we choose

m	x_m	$J_3(x_m)$
0	0.1	-3.95634
1	0.2	-3.91421
2	0.5	-3.7962
3	0.7	-3.72387
4	1.2	-3.56202
5	1.4	-3.50388
6	1.6	-3.44903
7	2.0	-3.3481
8	3.0	-3.13707
9	4.0	-2.96948
10	8.0	-2.53355

(7.52)

and solving the linear equations gives

ℓ	b_ℓ	a_ℓ
0	24.9508	24.9508
1	−0.925829	−0.913013
2	9.63281	13.5336
3	−2.94514	−1.2945
4	6.44874	4.60507
5	−2.10201	−1.88678

(7.53)

globalvars.f90:

```
! j3() approximates the j3 integral by rations
! functions with coefficients:
!
  INTEGER, PARAMETER         :: NNJ3=3, NMJ3=2*NNJ3-1 ! NMJ3=5
  REAL,    DIMENSION(0:NMJ3) :: J3B, J3A
  PARAMETER (           &
  J3B=(/                &
   24.9719E0  ,         & !b0
   -0.923982E0,         & !b1
   9.62659E0  ,         & !b2
   -2.93352E0 ,         & !b3
   6.44425E0  ,         & !b4
   -2.10031/) ,         & !b5
   J3A=(/               &
    J3B(0),             & !a0
    -0.926079E0,        & !a1
    13.5296E0  ,        & !a2
    -1.28659E0 ,        & !a3
    4.59814E0  ,        & !a4
    -1.88505E0/)        ) !a5
  REAL, PARAMETER ::  J3MM = 4./9.                 ! 0.444444
  REAL, PARAMETER ::  J3BB =-4.0E0                 !-4.0
  REAL, PARAMETER ::  J3AA1=-SQPI                  !-1.7724539
  REAL, PARAMETER ::  J3AA2=-SQPI*(GAMMA+LOG4)     !-3.4802318
  REAL, PARAMETER ::  J3CC = 0.1E0
  REAL, PARAMETER ::  J3EE = 0.2E0
```

dedx.f90:

```
 FUNCTION j3(x)
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: j3

   REAL,    PARAMETER   :: XMIN=0.4, XMAX=2.3D0
   REAL    :: x2, x3, x4
   REAL    :: xx, ra, rc
   REAL    :: y
   INTEGER :: n
!
! analytic asymptotic forms
!
  IF (x .LE. XMIN) THEN
     j3=J3MM*x+J3BB
```

```
  ELSEIF (x .GE. XMAX) THEN
     y=SQRT(x)
     j3=(J3AA1*LOG(x) + J3AA2)/y
  ELSE
!
! numerical asymptotic form
!
     x2 =x*x
     x3 =x**2.5
     x4 =x**3.5
     j3=(J3MM*x+J3BB)/(J3CC*x4+1) + &
        J3EE*(J3AA1*LOG(1+x)+J3AA2)*x2/(J3EE*x3+1)
!
! spline correction
!
     ra=0.E0
     rc=0.E0
     xx=1.E0
     DO n=0,NMJ3
        ra=ra+J3A(n)*xx
        rc=rc+J3B(n)*xx
        xx=x*xx
     ENDDO
     ra=ra+xx
     rc=rc+xx
     j3=j3*rc/ra
  ENDIF
END FUNCTION j3
```

4. $J_4(x)$

We will now consider the integral

$$J_4(x) \equiv \int_0^1 du\, e^{-xu} \sqrt{u}\, \ln u \; . \tag{7.54}$$

where $x \geq 0$. We can obtain an analytic solutions for small and large values of the argument: $x \ll 1$ and $x \gg 1$. In the former case, we expand the exponent to linear order and find

$x \ll 1$:

$$J_4(x) = b + mx + fx^2 \qquad b = -4/9 \quad m = 4/25 \quad f = -2/17 \tag{7.55}$$

error 0.01% for $x < x_{\min} = 0.18$

$x \gg 1$:

$$J_4(x) = (a_1 \ln x + a_2) x^{-3/2} \qquad a_1 = -\frac{\sqrt{\pi}}{2} \quad a_2 = \frac{\sqrt{\pi}}{2}(2 - \gamma - \ln 4) \tag{7.56}$$

error 0.3% for $x > x_{\max} = 4.7$

The former case is evaluated as before, while in the latter case the support lies near $u \sim 1$ and we replace the upper limit by infinity and write

$$J_4(x) \approx \int_0^\infty du\, e^{-xu}\, \sqrt{u}\, \ln u = \int_0^\infty \frac{du}{x}\, e^{-u}\, \sqrt{(u/x)}\, \ln(u/x) \quad : u \to u/x \tag{7.57}$$

$$= x^{-3/2} \int_0^\infty du\, e^{-u}\, \frac{\ln u - \ln x}{\sqrt{u}}\ . \tag{7.58}$$

At intermediate values of x between $x_{\rm min}$ and $x_{\rm max}$ we approximate

$$J_4(x) = R(x)\, Q_6(x)\ . \tag{7.59}$$

We take the rational approximation R to be

$$R(x) = \underbrace{\frac{mx+b}{c\,x^{7/2}+1}}_{x\ll 1:\ mx+b} + \underbrace{\frac{e\,[a_1 \ln(x+1)+a_2]x^2}{e\,x^{7/2}+1}}_{x\gg 1:\ (a_1 \ln x + a_2)\, x^{-3/2}}\ , \tag{7.60}$$

where x^2 in the numerator of the second term is chosen so that this term dominates over the first term at large x. In summary, we take

coeff	numerical	exact
m	0.16	$4/25$
b	-0.444444	$-4/9$
c	0.1	arbitrary
e	0.1	arbitrary
a_1	-0.886227	$-\sqrt{\pi}/2$
a_2	0.0323384	$\sqrt{\pi}\,(2-\gamma-\ln 4)/2$

(7.61)

For $n = 6$ we need $2n - 1 = 11$ data points: we will take m to start at zero and end at ten, and we choose

m	x_m	$J_4(x_m)$
0	0.1	-0.428845
1	1.0	-0.318233
2	10	-0.0635077
3	20	-0.0293211
4	30	-0.0181472
5	40	-0.0127948
6	50	-0.00971452
7	60	-0.00773775
8	70	-0.00637363
9	80	-0.00538212
10	90	-0.00463275

(7.62)

and solving the linear equations gives

ℓ	b_ℓ	a_ℓ
0	$1.745934124305194E7$	$1.745934124305194E7$
1	$-2.227978010029586E7$	$-2.238319621183343E7$
2	$1.295264548871798E6$	$1.477987858851782E6$
3	-785340	-788364
4	6120.99	6041.5
5	533.886	533.792

(7.63)

globalvars.f90:

```
! j4() approximates the j4 integral by rations
! functions with coefficients:
!
  INTEGER, PARAMETER         :: NNJ4=3, NMJ4=2*NNJ4-1 ! NMJ4=5
  REAL,    DIMENSION(0:NMJ4) :: J4B, J4A
  PARAMETER (              &
  J4B=(/                   &
   1.368871985536256E7,    & !b0
   -1.51863372154072E7,    & !b1
   920692.E0  ,            & !b2
   -590484.E0 ,            & !b3
   1763.2E0   ,            & !b4
   415.931E0/),            & !b5
   J4A=(/                  &
    J4B(0),                & !a0
    -1.5303334226695618E7,& !a1
    1.064250585959179E6  ,& !a2
    -592292.E0    ,        & !a3
    1697.91E0     ,        & !a4
    415.831E0/)            ) !a5
```

dedx.f90:

```
 FUNCTION j4(x)
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: j4

   REAL,    PARAMETER   :: XMIN=0.18D0, XMAX=4.7D0
   REAL    :: x2, x4
   REAL    :: xx, ra, rc
   REAL    :: y, y3
   INTEGER :: n
!
! analytic asymptotic forms
!
  IF (x .LE. XMIN) THEN
     j4=J4MM*x+J4BB
  ELSEIF (x .GE. XMAX) THEN
     y=SQRT(x)                    ! x^1/2
     y3=x*y                       ! x^3/2
     j4=(J4AA1*LOG(x) + J4AA2)/y3
  ELSE
!
```

```
! numerical asymptotic form
!
     x2 =x*x
     x4 =x**3.5
     j4=(J4MM*x+J4BB)/(J4CC*x4+1) + &
         J4EE*(J4AA1*LOG(1+x)+J4AA2)*x2/(J4EE*x4+1)
!
! spline correction
!
     ra=0.E0
     rc=0.E0
     xx=1.E0
     DO n=0,NMJ4
        ra=ra+J4A(n)*xx
        rc=rc+J4B(n)*xx
        xx=x*xx
     ENDDO
     ra=ra+xx
     rc=rc+xx
     j4=j4*rc/ra
  ENDIF
 END FUNCTION j4
```

Appendix A

We now list the complete source code sequentially.

dedx_main.f90:

```
PROGRAM dedxmain
!
! This is the driver for the BPS dedx subroutine,
! version 3.12
!
  USE globalvars

  IMPLICIT NONE
  INTEGER :: nni ! number of ions excluding electrons
  REAL,    DIMENSION(:), ALLOCATABLE :: beta, mb, nb, zb
  REAL    :: vp, zp, mp, ee
  REAL    :: dedxtot, dedxsumi
  REAL    :: ne, te, ti, betae, betai
  REAL    :: vpe, betam, vv
  REAL    :: dedxctot, dedxqtot, dedxcsumi, dedxqsumi

! projectile - DT plasma (NNB=3 or nni=2)
!
  nni=2
  ALLOCATE(mb(1:nni+1),nb(1:nni+1),zb(1:nni+1))

  ee=3.54E0    ! projectile energy (MeV)
  ee=0.01E0    ! projectile energy (MeV)
  zp=2         ! projectile charge
  mp=4*MPEV    ! projectile mass
  te=60.       ! temperature of electrons (keV)
  ti=te        ! temperature of ions      (keV)
  ne=1.4448E26 ! electron numr density (cm^-3)
  zb(1)=-1.    ! species charges
  zb(2)=+1.    !
  zb(3)=+1.    !
  mb(1)=MEEV   ! species masses
  mb(2)=2*MPEV !
  mb(3)=3*MPEV !
  nb(1)=1.     ! number density with charge neutrality
  nb(2:nni+1)=1./(zb(2:nni+1)*nni)

  ALLOCATE(beta(1:nni+1))            ! inverse temperature
  betae=1.E-3/te                     ! array (eV^-1)
  betai=1.E-3/ti
  beta(1)=betae
  IF ( nni .GE. 1) THEN
     beta(2:nni+1)=betai
  ENDIF
  nb=nb*ne/CMTOA0**3                 ! number density in
                                     ! atomic units (a0^-3)
! convert E to vp
!
  betam=beta(I)
  vv=SQRT(R/(betam*mb(I))) ! vthc: thermal velocity units of c
  vp=vpe(ee,mp,vv)
```

```
  PRINT *, "E :", ee ! MeV
  PRINT *, "vp:", vp ! units of vth for species I
  CALL dedx_bps(nni,vp,zp,mp,beta,zb,mb,nb,dedxtot,dedxsumi,&
       dedxctot,dedxcsumi,dedxqtot,dedxqsumi)
  PRINT *, ee, dedxtot
  PRINT *, ee, dedxsumi

  DEALLOCATE(beta,mb,nb,zb)
END PROGRAM dedxmain

FUNCTION vpe(ee, mp, vv)
!
!
!
! vpe= projectile velocity in units of vth
! vv = thermal velocity in units of c (vthc)
!
  USE globalvars
  IMPLICIT NONE
  REAL ee, mp, vv, vpe
  vpe=SQRT(2*ee/(mp/EV))/vv
END FUNCTION vpe
```

globalvars.f90:

```
MODULE globalvars
!
! mathematical constants
!
  REAL,    PARAMETER :: PI   =3.141592654   ! pi
  REAL,    PARAMETER :: SQPI =1.772453851   ! sqrt(pi)
  REAL,    PARAMETER :: GAMMA=0.577215665   ! Euler Gamma
  REAL,    PARAMETER :: LOG2 =0.6931471806  ! ln(2)
  REAL,    PARAMETER :: LOG4 =1.386294361   ! ln(4)
  REAL,    PARAMETER :: LOG8 =2.079441542   ! ln(8)
  REAL,    PARAMETER :: LOG16=2.772588722   ! ln(16)
  REAL,    PARAMETER :: ZETA3=1.202056903   ! zeta(3)
  REAL,    PARAMETER :: EXP2E=3.172218958   ! exp(2*GAMMA)
!
! physical parameters and conversion factors
!
  REAL,    PARAMETER :: BE=13.6             ! binding energy of Hydrogen
  REAL,    PARAMETER :: CC=2.998E10         ! speed of light
  REAL,    PARAMETER :: MPEV =0.938271998E9 ! proton mass in eV
  REAL,    PARAMETER :: MEEV =0.510998902E6 ! electron mass in eV
  REAL,    PARAMETER :: AMUEV=0.931494012E9 ! AMU in eV
  REAL,    PARAMETER :: KTOEV=8.61772E-5    ! conversion factor
  REAL,    PARAMETER :: CMTOA0=1.8867925E8  ! conversion factor
  REAL,    PARAMETER :: MTR=1.E-6           ! length unit
  REAL,    PARAMETER :: EV=1.E6             ! energy unit
  REAL,    PARAMETER :: CONVFACT=CMTOA0*(MTR*100.)/EV
!
! misc parameters
!
  INTEGER, PARAMETER :: R=3        ! thermal velocity parameter
```

```
  INTEGER, PARAMETER :: I=1        ! plasma species index
  REAL               :: K          ! arbitrary wave number units a0^-1

! plasma parameters: values set in dedx_bps
!
! REAL,    DIMENSION(1:NNB)          :: kb2, ab, bb, cb, eb, fb, rb, gb
! REAL,    DIMENSION(1:NNB)          :: ab2, etb, rmb0, rrb0, mb0
  REAL,    DIMENSION(:), ALLOCATABLE :: kb2, ab, bb, cb, eb, fb, rb, gb
  REAL,    DIMENSION(:), ALLOCATABLE :: ab2, etb, rmb0, rrb0, mb0
  LOGICAL, DIMENSION(:), ALLOCATABLE :: lzb
  REAL    :: cp1, cp2, cp3, vth, vthc, mp0, kd
  INTEGER :: NNB  ! number of plasma species = ni+1

! daw() approximates Dawson's integral by rational
! functions with coefficients:
!
  INTEGER, PARAMETER          :: NNDAW=3, NMDAW=2*NNDAW-1 ! NMDAW=5
  REAL,    DIMENSION(0:NMDAW) :: DWB, DWA
  PARAMETER (             &
  DWB=(/                  &
    5.73593880244318E0,   & !b0
   -6.73666007137766E0,   & !b1
    1.99794422787154E1,   & !b2
   -1.85506350260761E1,   & !b3
    1.22651360905700E1,   & !b4
   -4.67285812684807E0/),& !b5
  DWA=(/                  &
    DWB(0),               & !a0
   -6.82372048950896E0,   & !a1
    1.33804115903096E1,   & !a2
   -1.42130723670491E1,   & !a3
    1.11714434417979E1,   & !a4
   -4.66303387468937E0/) ) !a5

! j1() approximates the j1 integral by rations
! functions with coefficients:
!
  INTEGER, PARAMETER         :: NNJ1=3, NMJ1=2*NNJ1-1 ! NMJ1=5
  REAL,    DIMENSION(0:NMJ1) :: J1B, J1A
  PARAMETER (           &
  J1B=(/                &
   -926.65E0  ,         & !b0
   787.016E0  ,         & !b1
   -329.764E0 ,         & !b2
   39.7406E0  ,         & !b3
   -0.173896E0,         & !b4
   -1.66913E0 /),       & !b5
  J1A=(/                &
    J1B(0),             & !a0
    787.165E0 ,         & !a1
    -213.584E0,         & !a2
    -1.04219E0,         & !a3
    33.594E0  ,         & !a4
   -11.8391E0/)         ) !a5
  REAL, PARAMETER :: J1MM= (4./9.)*(4.-LOG8)    ! 0.8535815
  REAL, PARAMETER :: J1BB=-(4.-LOG16)           !-1.2274113
  REAL, PARAMETER :: J1AA=-SQPI/2.              !-0.8862270
```

```
  REAL, PARAMETER :: J1CC= 0.1E0
  REAL, PARAMETER :: J1EE= 0.2E0
  REAL, PARAMETER :: J1GG=-3.*SQPI/8.

! j2() approximates the j2 integral by rations
! functions with coefficients:
!
  INTEGER, PARAMETER         :: NNJ2=3, NMJ2=2*NNJ2-1 ! NMJ2=5
  REAL,    DIMENSION(0:NMJ2) :: J2B, J2A
  PARAMETER (           &
  J2B=(/                &
    87.1714E0 ,         & !b0
    -277.584E0,         & !b1
    329.082E0 ,         & !b2
    -180.982E0,         & !b3
    56.7202E0 ,         & !b4
    -8.60238E0/),       & !b5
   J2A=(/               &
    J2B(0),             & !a0
    -277.693E0,         & !a1
    329.801E0 ,         & !a2
    -184.219E0,         & !a3
    59.9325E0 ,         & !a4
    -10.1138E0/)        ) !a5
  REAL, PARAMETER ::  J2MM= 4.*(23.-15.*LOG2)/75. ! 0.6721489
  REAL, PARAMETER ::  J2BB=-4.*(4.-LOG8)/9.       !-0.8535815
  REAL, PARAMETER ::  J2AA=-3.*SQPI/4.            !-1.3293405
  REAL, PARAMETER ::  J2CC= 0.5E0
  REAL, PARAMETER ::  J2EE= 0.2E0
  REAL, PARAMETER ::  J2GG=-15.*SQPI/16.

! j3() approximates the j3 integral by rations
! functions with coefficients:
!
  INTEGER, PARAMETER         :: NNJ3=3, NMJ3=2*NNJ3-1 ! NMJ3=5
  REAL,    DIMENSION(0:NMJ3) :: J3B, J3A
  PARAMETER (           &
  J3B=(/                &
   24.9719E0  ,         & !b0
   -0.923982E0,         & !b1
   9.62659E0  ,         & !b2
   -2.93352E0 ,         & !b3
   6.44425E0  ,         & !b4
   -2.10031/) ,         & !b5
   J3A=(/               &
    J3B(0),             & !a0
    -0.926079E0,        & !a1
    13.5296E0  ,        & !a2
    -1.28659E0 ,        & !a3
    4.59814E0  ,        & !a4
    -1.88505E0/)        ) !a5
  REAL, PARAMETER ::  J3MM = 4./9.                ! 0.444444
  REAL, PARAMETER ::  J3BB =-4.0E0                !-4.0
  REAL, PARAMETER ::  J3AA1=-SQPI                 !-1.7724539
  REAL, PARAMETER ::  J3AA2=-SQPI*(GAMMA+LOG4)    !-3.4802318
  REAL, PARAMETER ::  J3CC = 0.1E0
  REAL, PARAMETER ::  J3EE = 0.2E0
```

```
! j4() approximates the j4 integral by rations
! functions with coefficients:
!
  INTEGER, PARAMETER         :: NNJ4=3, NMJ4=2*NNJ4-1 ! NMJ4=5
  REAL,    DIMENSION(0:NMJ4) :: J4B, J4A
  PARAMETER (             &
  J4B=(/                  &
   1.368871985536256E7,   & !b0
   -1.51863372154072E7,   & !b1
   920692.E0  ,           & !b2
   -590484.E0 ,           & !b3
   1763.2E0   ,           & !b4
   415.931E0/),           & !b5
   J4A=(/                 &
    J4B(0),               & !a0
    -1.5303334226695618E7,& !a1
    1.064250585959179E6  ,& !a2
    -592292.E0    ,       & !a3
    1697.91E0     ,       & !a4
    415.831E0/)           ) !a5
  REAL, PARAMETER :: J4MM = 4./25.                    ! 0.16
  REAL, PARAMETER :: J4BB =-4./9.                     !-0.4444444E0
  REAL, PARAMETER :: J4AA1=-SQPI/2.                   !-0.8862269E0
  REAL, PARAMETER :: J4AA2=SQPI*(2.-GAMMA-LOG4)/2.    ! 3.23383974E-02
  REAL, PARAMETER :: J4CC = 0.1E0
  REAL, PARAMETER :: J4EE = 0.1E0

END MODULE globalvars

!
! D-T plasma:
!
!      PARAMETER (ZB(1)=-1)        ! charge of 1-st component
!      PARAMETER (ZB(2)=+1)        ! charge of 2-nd component
!      PARAMETER (ZB(3)=+1)        ! charge of 3-rd component
!      PARAMETER (MB(1)=MEEV)      ! mass of   1-st component
!      PARAMETER (MB(2)=2*MPEV)    ! mass of   2-nd component
!      PARAMETER (MB(3)=3*MPEV)    ! mass of   3-rd component
!      PARAMETER (NB(1)=NE)        ! density of 1-st comp
!      PARAMETER (NB(2)=0.5*NE)    ! density of 2-nd comp
!      PARAMETER (NB(3)=0.5*NE)    ! density of 3-rd comp
```

dedx.f90:

```
! These subroutines implements the BPS stopping power: ‘‘Charged
! Particle Motion in a Highly Ionized Plasma’’, L. Brown, D. Preston,
! and R. Singleton Jr., Physics Reports, Vol. 410, No. 4, 237-333
! (2005); or arXiv:physics/0501084.
!
! See BPSx.xx/doc/BPS_phys_rep.pdf for the full Phys. Rep. paper,
! and see BPSx.xx/doc/doc.pdf for code documentation.
!
! Robert Singleton, LANL, X-7
!
```

```
! v3.13: Jan-06, 4rd production version
! v3.12: May-05, 3rd production version
! v3.09: May-04, 2nd production version
! v3.04: Jan-03, 1st production version
!
 SUBROUTINE dedx_bps(nni, vp, zp, mp, betab, zb, mb, nb, &
     dedxtot, dedxsumi, dedxctot, dedxcsumi, dedxqtot, dedxqsumi)
   USE globalvars

   IMPLICIT NONE
   INTEGER,                      INTENT(IN)  :: nni    ! number of ions
   REAL,    DIMENSION(1:nni+1), INTENT(IN)  :: betab  ! plasma temp array
   REAL,    DIMENSION(1:nni+1), INTENT(IN)  :: mb, nb ! mass and density
   REAL,    DIMENSION(1:nni+1), INTENT(IN)  :: zb     ! charge array
   REAL,                        INTENT(IN)  :: vp     ! projectile velocity
   REAL,                        INTENT(IN)  :: zp     ! projectile charge
   REAL,                        INTENT(IN)  :: mp     ! projectile mass
   REAL,                        INTENT(OUT) :: dedxtot,  dedxsumi
   REAL,                        INTENT(OUT) :: dedxctot, dedxcsumi
   REAL,                        INTENT(OUT) :: dedxqtot, dedxqsumi

   REAL, DIMENSION(1:nni+1) :: mpb0, rpb0
   REAL :: betam, mm

   REAL                     :: e, gd
   REAL, DIMENSION(1:nni+1) :: gpb

   REAL, DIMENSION(1:nni+1) :: ub2, mpb, loglamb

   NNB=nni+1
   ALLOCATE(kb2(1:NNB),ab(1:NNB),bb(1:NNB),cb(1:NNB))
   ALLOCATE(eb(1:NNB),fb(1:NNB),rb(1:NNB),gb(1:NNB))
   ALLOCATE(ab2(1:NNB),etb(1:NNB),rmb0(1:NNB),rrb0(1:NNB))
   ALLOCATE(mb0(1:NNB),lzb(1:NNB))
!
! plasma parameters
!
   betam=betab(I)               ! inv temp of index plasma species
   rb=R*betab/betam             ! r_b array
   kb2=8*PI*BE*betab*zb*zb*nb   ! inv Debye length squared
   kd=SUM(ABS(kb2))             ! total inv Debye length
   kd=SQRT(kd)                  ! units a0^-1
   K =kd                        ! set K to Debye

   mm=mb(I)                     ! mass of index plasma species
   mp0=mp/mm                    ! rescaled proj mass
   cp1=2*BE*zp**2               ! units of eV-a0
   cp2=(BE*zp**2)/(2*PI)        ! units of eV-a0
   cp3=(BE*zp**2)/(PI*mp0)      ! dimensionless parameter
   vthc=SQRT(R/(betam*mm))      ! thermal velocity of mm: units of c
   vth =CC*vthc                 ! thermal velocity of mm: units cm/s

   mb0 =mb/mm                   ! rescaled plasma masses
   mpb0=mp0 + mb0               ! Mpb0
   rpb0=mp0*mb0/mpb0            ! mpb0
   rmb0=mpb0/mp0                ! rm0=rMb0=(mp0+mb0)/mp0
   rrb0=mb0/mp0                 ! rr0=rmb0=mb0/mp0
```

```
   ab  =SQRT(rb*mb0/2)
   ab2 =ab*ab
   bb  =rb*mpb0
   eb  =(mb0/mp0)/SQRT(2*PI*rb*mb0)
   etb =2*BE*ABS(zp*zb)*(2.686E-4)/vthc
   fb  =2/SQRT(2*PI*rb*mb0)
   WHERE ( zb /= 0 )             ! do not take log of zero
      lzb=.TRUE.                 ! flag for future use
      cb  =2 - 2*GAMMA - LOG(ABS((2*BE)*betab*zp*zb*K*mb0/rpb0))
      gb  =0.5 + 2*GAMMA + Log(ABS(0.5*BE*&    ! for small vp limit
           betab*zp*zb*kd*mb0/rpb0))           !
   ELSEWHERE
      cb=0
      gb=0
      lzb=.FALSE.
   ENDWHERE

! check for charge neutrality
!
   e=SUM(zb*nb)
   PRINT *, 'charge  = ', e

! print K and kd
!
   PRINT *, 'K(a0^-1)= ', K
   PRINT *, 'kd      = ', kd

! g-factors
!
   gpb=2*BE*betam*SQRT(kb2)       ! g_pb
   gd =2*BE*betam*kd              ! g_d
   PRINT *,'gD      = ', gd

! thermal velocity (cm/s)
!
   PRINT *, 'vth     = ', vth
!
! Coulomb log (I think I left out a factor of k_D from the
!              argument of the log)
!
   ub2=(vp*vthc)**2 + 2/(betab*mb)**2     ! velocity (units of c)
   mpb=rpb0*mm                            ! reduced mass array (eV)
   loglamb=(8*PI*zb*zb)**2/(mpb*ub2)**2   ! a0 = 5.29*10^-11 m
   loglamb=loglamb + (2.69E-4)**2/(2*mpb*mpb*ub2) ! = 2.69*10^-4 eV
   loglamb=-0.5*LOG(loglamb)
   PRINT *, 'log(Lam)=',loglamb

   CALL dedxc(vp,dedxctot,dedxcsumi)      ! returned in MeV/mu-m
   CALL dedxq(vp,dedxqtot,dedxqsumi)   !
   dedxtot =dedxctot  + dedxqtot
   dedxsumi=dedxcsumi + dedxqsumi
!!
   PRINT *,"       MeV/mu-m    MeV/mu-m"
   PRINT *,"tot:",dedxtot,  dedxsumi
   PRINT *,"cl :",dedxctot, dedxcsumi
   PRINT *,"qm :",dedxqtot, dedxqsumi
!!
```

```
   DEALLOCATE(kb2,ab,bb,cb,eb,fb,rb,gb)
   DEALLOCATE(ab2,etb,rmb0,rrb0,mb0,lzb)
 END SUBROUTINE dedx_bps


!
! quantum correction:
!
! This subroutine calculates the quantum correction dedxq.
! It returns dedxqtot and dedxqsumi. The following functions
! and subroutines are defined here:
!
! Subroutine: dedxq
!
! Function  : dedxqi
!
! Function  : d_dedxq
!
! Function  : repsi
!
 SUBROUTINE dedxq(vp, dedxqtot, dedxqsumi)
   USE globalvars

   IMPLICIT NONE
   REAL, INTENT(IN) :: vp
   REAL, INTENT(OUT):: dedxqtot, dedxqsumi

   REAL, DIMENSION(1:NNB) :: qmb
   REAL                   :: dedxqi, a1, a2, e, rm, rr
   INTEGER                :: ib


   qmb=0
   DO ib=1,NNB            ! sum over plasma species
      IF ( lzb(ib) ) THEN ! compute only if zb(ib) /= 0
         a1=ab(ib)
         a2=a1*a1
         e=etb(ib)
         rm=rmb0(ib)      ! rmb0=rMb0=(mp0+mb0)/mp0
         rr=rrb0(ib)      ! rrb0=rmb0=mb0/mp0
         qmb(ib)=qmb(ib)+dedxqi(vp,a2,e,rm,rr)
         qmb(ib)=qmb(ib)*kb2(ib)*fb(ib)
      ELSE
         qmb(ib)=0        ! don't compute if zb(ib) = 0
      ENDIF
   ENDDO
   dedxqsumi=SUM(qmb(2:NNB))
   dedxqtot =qmb(1)
   dedxqtot =dedxqtot+dedxqsumi

   dedxqtot=CONVFACT*(cp1/vp)*dedxqtot
   dedxqsumi=CONVFACT*(cp1/vp)*dedxqsumi
 END SUBROUTINE dedxq


!
! a = ab(ib)*ab(ib)
```

```
! e = etb(ib)
!
!
!           /Infinity
!           |
! dedxq  =  | du  d_dedxq(u)
!           |
!           /0
!
!
 FUNCTION dedxqi(v, a, e, rm, rr)
!
! This function performs the integration numerically by
! Gaussian Quadrature. The number of intervals NG that [u0,u1]
! is broken into is hardwired in a PARAMETER statement, but
! it be changed by the user (must be even).
!
! NOTE on Gaussian Quadrature:
!
! The polynomial P3(x)=(5*x^3-3*x)/2 is employed, and I have
! defined the appropriate weights W13, W2 and relative position
! UPM in parameter statements.
!
   IMPLICIT NONE

   REAL, INTENT(IN) :: v, a, e, rm, rr
   REAL :: dedxqi

   REAL,    PARAMETER :: UPM=0.7745966692E0
   REAL,    PARAMETER :: W13=0.5555555556E0, W2=0.8888888889E0
   INTEGER, PARAMETER :: NG=10000  ! must be even
   REAL,    PARAMETER :: NN=30.E0

   REAL,    PARAMETER :: SQPI =1.772453851    ! sqrt(pi)
   REAL,    PARAMETER :: SMAX=10., SMIN=0.05  ! cuts on s
   REAL     :: q1, q2, dg, ddg, g0, g1, g2, h0, h2
   REAL     :: s1, s2, s3, s05, s15
   REAL     :: repsi, repsi1, repsi2

   REAL     :: i1, i2, kqm1, kqm3

   REAL     :: x0, x1, dx
   REAL     :: x, xm, d_dedxq
   INTEGER  :: ix
   REAL     :: r, s

   s=a*v*v
   IF ( s .GT. SMAX) THEN          ! large s can be performed analytically
      r=e/v                        ! this case is usually realized for ions
      g0 =LOG(ABS(r)) - repsi(r)   !
      dg =1/r - repsi1(r)          !
      ddg=-1/(r*r) - repsi2(r)     !
      g1=-r*dg                     !
      g2=2*r*dg + r*r*ddg          !
      s1=1/s                       ! s1=1/s
      s2=s1/2                      ! s2=1/2*s
      s3=3*s2                      ! s3=3/2*s
```

```
      s05=SQRT(s)                   ! s05=s^(1/2)
      s15=s*s05                     ! s15=s^(3/2)
      h0=g0*(1-s2)
      h2=g2*(1-s2) - 2*g1*(1-s1) + 2*g0*(1-s3)
      q1=SQPI/2
      q2=SQPI/2
      q1=q1*(g0/s05 + 0.25E0*g2/s15)
      q2=q2*(h0/s05 + 0.25E0*h2/s15)
      dedxqi=rm*q2 - rr*q1
   ELSEIF ( s .LT. SMIN) THEN       ! small s can be performed analytically
      r=SQRT(a)*e                   ! this case is usually realized for elec.
      i1=kqm1(r)                    ! accuracy < 0.5%
      i2=kqm3(r)
      dedxqi=(4.*rm*s/3. - 2*rr)*i1
      dedxqi=dedxqi+ 4.*(4*rm*s*s/15.- rr*s/3.)*i2
      dedxqi=EXP(-s)*dedxqi
   ELSE                             ! otherwise do integral numerically
      r=e/v
      x0=1.E0 - NN/SQRT(s)          ! by Gaussian quadrature
      x0=MAX(0.,x0)
      x1=1.E0 + NN/SQRT(s)
      dx=(x1-x0)/NG
      dedxqi=0.E0
      x=x0-dx
      DO ix=1,NG,2
!
         x=x+2.E0*dx
         dedxqi=dedxqi+W2*d_dedxq(r,s,rm,rr,x)
!
         xm=x-dx*UPM
         dedxqi=dedxqi+W13*d_dedxq(r,s,rm,rr,xm)
!
         xm=x+dx*UPM
         dedxqi=dedxqi+W13*d_dedxq(r,s,rm,rr,xm)
      ENDDO
      dedxqi=dedxqi*dx
   ENDIF
 END FUNCTION dedxqi


 FUNCTION d_dedxq(r, s, rm, rr, x)
   IMPLICIT NONE
   REAL, INTENT(IN) :: r, s, rm, rr, x
   REAL, PARAMETER  :: SXMAX=0.05
   REAL :: d_dedxq
   REAL :: repsi, rx, sx, sh, ch
   REAL :: ep, em, xm1, xp1
   rx=r/x
   sx=2*s*x
   xm1=x-1
   xp1=x+1
   ep=EXP(-s*xp1*xp1)
   em=EXP(-s*xm1*xm1)
   sh=0.5E0*(em-ep)          ! sh and ch are
   IF (sx .GT. SXMAX) THEN   ! not sinh or cosh
      ch=0.5E0*(em+ep)       !
      ch=(ch - sh/sx)/x
```

```
   ELSE
      ch=2.E0*sx/3 + (1.E0/15.E0 - 1.E0/(6.E0*s))*sx*sx*sx
      ch=s*ch*EXP(-s)
   ENDIF
   d_dedxq=LOG(ABS(rx)) - repsi(rx)
   d_dedxq=d_dedxq*(rm*ch - rr*sh)
 END FUNCTION d_dedxq


!
! This is a fit to repsi(x) = Re Psi(1 + I*x), where
! Psi is the PolyGamma function. The accuracy is 0.1%.
!
 FUNCTION repsi(x)
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL, PARAMETER :: XMIN=0.16E0, XMAX=1.5E0
   REAL, PARAMETER :: TZETA3=2.404113806319188 ! 2*ZETA(3)
   REAL, PARAMETER :: A=0.1E0, B=1.33333E0, C=1.125E0
   REAL :: repsi
   REAL :: x2, x4
   IF (x .LE. XMIN) THEN
      x2=x**2
      repsi=-GAMMA + ZETA3*x2
   ELSEIF (x .GE. XMAX) THEN
      x2=x**2
      x4=x2*x2
      repsi=LOG(x)+1.D0/(12.D0*x2)+1.D0/(120.D0*x4)
   ELSE
      x2=x*x
      repsi=0.5E0*LOG(1 + (EXP2E*x2*x2 + TZETA3*x2)/(1+x2))
      repsi=repsi/(1 - A*EXP(-B*x - C/x)) - GAMMA
   ENDIF
 END FUNCTION repsi

!              d
! repsi1(x) = --- Re[ Psi(1 + I*x) = -Im Psi'(1 + I*x).
!             dx
 FUNCTION repsi1(x)
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: repsi1

   REAL, PARAMETER :: XMIN=0.14E0, X1=0.7E0 ,XMAX=1.9E0
   REAL, PARAMETER :: ZETA32=2.404113806319188 ! 2*ZETA(3)
   REAL, PARAMETER :: ZETA54=4.147711020573480 ! 4*ZETA(5)
   REAL, PARAMETER :: a0= 0.004211521868683916
   REAL, PARAMETER :: a1= 2.314767988469241000
   REAL, PARAMETER :: a2= 0.761843932767193200
   REAL, PARAMETER :: a3=-7.498711815965575000
   REAL, PARAMETER :: a4= 7.940030433629257000
   REAL, PARAMETER :: a5=-2.749533936429732000
   REAL, PARAMETER :: b0=-0.253862873373708200
   REAL, PARAMETER :: b1= 4.600929855835432000
   REAL, PARAMETER :: b2=-6.761540444078382000
   REAL, PARAMETER :: b3= 4.467238548899841000
```

```
   REAL, PARAMETER :: b4=-1.444390097613873500
   REAL, PARAMETER :: b5= 0.185954029179227070
   REAL :: xi
   IF ( x .LE. XMIN) THEN              ! x < xmin=0.14
      repsi1=ZETA32*x - ZETA54*x*x*x   ! accurate to 0.1%
   ELSEIF (x .LE. x1) THEN
      repsi1=a5                        ! xmin < x < x1=0.7
      repsi1=a4 + repsi1*x             ! accurate to 0.002%
      repsi1=a3 + repsi1*x             ! a0 + a1*x + a2*x^2 +
      repsi1=a2 + repsi1*x             ! a3* x^3 + a4*x^4 + a5*x^5
      repsi1=a1 + repsi1*x
      repsi1=a0 + repsi1*x
   ELSEIF (x .LE. xmax) THEN           ! x1 < x < xmax=1.9
      repsi1=b5                        ! accurate to 0.1%
      repsi1=b4+repsi1*x               ! b0 + b1*x + b2*x^2 +
      repsi1=b3+repsi1*x               ! b3*x^3 + b4*x^4 +
      repsi1=b2+repsi1*x               ! b5*x^5
      repsi1=b1+repsi1*x
      repsi1=b0+repsi1*x
   ELSE
      xi=1/x                           ! x > xmax=1.9
      repsi1=-1.E0/30.E0               ! accurate to 0.08%
      repsi1=repsi1*xi                 ! 1/x - 1/6x^3 - 1/30x^5
      repsi1=-1.E0/6.D0 + repsi1*xi    !
      repsi1=repsi1*xi                 !
      repsi1=1.E0 + repsi1*xi          !
      repsi1=repsi1*xi                 !
   ENDIF
 END FUNCTION repsi1


!              d^2
! repsi2(x) = ---- Re[ Psi(1 + I*x) = -Re Psi''(1 + I*x).
!             dx^2
 FUNCTION repsi2(x)
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: repsi2

   REAL, PARAMETER :: XMIN=0.18E0, X1=1.2E0 ,XMAX=2.5E0
   REAL, PARAMETER :: ZETA32=2.4041138063191880 ! 2*ZETA(3)
   REAL, PARAMETER :: ZETA512=12.443133061720 44 ! 12*ZETA(5)
   REAL, PARAMETER :: ZETA730=30.250478321457685! 30*ZETA(7)
   REAL, PARAMETER :: a0= 2.42013533575662130
   REAL, PARAMETER :: a1=-0.41115258967949336
   REAL, PARAMETER :: a2=-8.09116694062588400
   REAL, PARAMETER :: a3=-24.9364824558827640
   REAL, PARAMETER :: a4=114.8109056152471800
   REAL, PARAMETER :: a5=-170.854545232781960
   REAL, PARAMETER :: a6=128.8402466765824700
   REAL, PARAMETER :: a7=-50.2459090010302060
   REAL, PARAMETER :: a8= 8.09941032385266400
   REAL, PARAMETER :: b0= 4.98436272402513600
   REAL, PARAMETER :: b1=-16.6546466531059530
   REAL, PARAMETER :: b2= 20.6941300362041100
   REAL, PARAMETER :: b3=-13.3726837850936920
   REAL, PARAMETER :: b4= 4.83094787289278800
```

```
   REAL, PARAMETER :: b5=-0.92976482601030100
   REAL, PARAMETER :: b6= 0.07456475055097825
   REAL :: xi, xx
   IF ( x .LE. XMIN) THEN
      xx=x*x
      repsi2=ZETA32 - ZETA512*xx + ZETA730*xx*xx ! x < xmin=0.18
   ELSEIF (x .LE. x1) THEN                       ! accurate to 0.1%
      repsi2=a8                                  !
      repsi2=a7 + repsi2*x                       ! xmin < x < x1=1.2
      repsi2=a6 + repsi2*x                       ! accurate to 0.01%
      repsi2=a5 + repsi2*x                       ! a0 + a1*x + a2*x^2 +
      repsi2=a4 + repsi2*x                       ! a3*x^3 + a4*x^4 +
      repsi2=a3 + repsi2*x                       ! a5*x^5 + a6*x^6 +
      repsi2=a2 + repsi2*x                       ! a7*x&7 + a8*x^8
      repsi2=a1 + repsi2*x
      repsi2=a0 + repsi2*x
   ELSEIF (x .LE. xmax) THEN                     ! x1 < x < xmax=2.5
      repsi2=b6                                  ! accurate to 0.2%
      repsi2=b5+repsi2*x                         ! b0 + b1*x + b2*x^2 +
      repsi2=b4+repsi2*x                         ! b3*x^3 + b4*x^4 +
      repsi2=b3+repsi2*x                         ! b5*x^5 + b6*x^6
      repsi2=b2+repsi2*x
      repsi2=b1+repsi2*x
      repsi2=b0+repsi2*x
   ELSE
      xi=1/x                                     ! x > xmax=2.5
      xi=xi*xi                                   ! accurate to 0.07%
      repsi2= 1.E0/6.E0                          ! -1/x^2 + 1/2x^4 +
      repsi2= 0.5E0 + repsi2*xi                  ! 1/6x^6
      repsi2=-1. + repsi2*xi                     !
      repsi2=repsi2*xi
   ENDIF
 END FUNCTION repsi2

!
!             /Infinity
!             |
! kqm1(x)  =  | dy   y exp(-y^2) [ln(x/y) - repsi(x/y)]
!             |
!            /0
!
!
 FUNCTION kqm1(x)
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: kqm1

   REAL, PARAMETER :: XMIN=0.15E0, XMAX=3.2E0

   REAL, PARAMETER :: a0= 0.43291174867614964545494494 29 ! 3*GAMMA/4
   REAL, PARAMETER :: a1= 1.20205690315959428542504315 61 ! ZETA(3)
   REAL, PARAMETER :: a2= 0.14879679441773450264109933 31 ! ZETA'(3)+GAMMA*
   REAL, PARAMETER :: b2=-0.04166666666666666666666666667 !-1/24  ZETA(3)/2
   REAL, PARAMETER :: b4=-0.00833333333333333333333333333 !-1/120
   REAL, PARAMETER :: b6=-0.01190476190476190476190476190 !-1/84
   REAL, PARAMETER :: c0= 0.25109815055856566000
   REAL, PARAMETER :: c1=-0.02989283169766254700
```

```
   REAL, PARAMETER :: c2= 0.03339880139150325000
   REAL, PARAMETER :: c3=-0.00799128953390392700
   REAL, PARAMETER :: c4= 0.00070251863606810650
   REAL, PARAMETER :: d0=-0.18373957827052560000
   REAL, PARAMETER :: d1=-0.33121125339783110000
   REAL, PARAMETER :: d2= 0.04022076263527408400
   REAL, PARAMETER :: d3=-0.00331897950305779480
   REAL, PARAMETER :: d4= 0.00012313574797356784
   REAL :: x2, lx, xi
   IF ( x .LE. XMIN) THEN                          ! x < xmin=0.15: to 0.06%
      x2=x*x                                       ! ln(x)/2 + 3*GAMMA/4 +
      lx=LOG(x)                                    ! ZETA(3)*X^2*ln(x) +
      kqm1=0.5E0*lx + a0 + a1*x2*lx + a2*x2        ! [ZETA'(3) + GAMMA*
                                                   ! ZETA(3)/2]*x^2
                                                   !
   ELSEIF ( x .GE. XMAX ) then                     ! x > xmax=3.2: to 0.12%
      xi=1/x                                       ! -1/24*x^2 - 1/120*x^4 -
      x2=xi*xi                                     ! 1/84*x^6
      kqm1=b6                                      !
      kqm1=b4 + kqm1*x2                            !
      kqm1=b2 + kqm1*x2                            !
      kqm1=kqm1*x2                                 !
   ELSE
      xi=1/x                                       ! xmin < x , xmax
      lx=LOG(x)                                    ! fit accurate to 0.2%
      kqm1=c4                                      ! c0 + c1*x + c2*x^2 +
      kqm1=c3+kqm1*x                               ! c3*x^3 + c4*x^4 +
      kqm1=c2+kqm1*x                               ! d0*ln(x) +
      kqm1=c1+kqm1*x                               ! d1/x + d2/x^2 +
      kqm1=c0+kqm1*x + d0*lx                       ! d3/x^3 + d4/x^4
      lx=d4                                        !
      lx=d3+lx*xi                                  !
      lx=d2+lx*xi                                  !
      lx=d1+lx*xi                                  !
      lx=lx*xi                                     !
      kqm1=kqm1+lx
   ENDIF
 END FUNCTION kqm1

!
!             /Infinity
!             |
! kqm3(x)  =  | dy   y^3 exp(-y^2) [ln(x/y) - repsi(x/y)]
!             |
!            /0
!
!
 FUNCTION kqm3(x)
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL :: kqm3

   REAL, PARAMETER :: XMIN=0.15E0, XMAX=2.5E0
   REAL, PARAMETER :: a0= 0.18291174867614964545494494 29 ! 3*GAMMA/4 - 1/4
   REAL, PARAMETER :: a2=-0.60102845157979714270733281 02 !-ZETA(3)/2
   REAL, PARAMETER :: b2=-0.08333333333333333333333333 33 !-1/12
   REAL, PARAMETER :: b4=-0.025                            !-1/40
```

```
   REAL, PARAMETER :: b6=-0.046875                          !-3/64
   REAL, PARAMETER :: c0= 0.691191700599840900000
   REAL, PARAMETER :: c1=-1.094849572900974000000
   REAL, PARAMETER :: c2= 0.318264617154601400000
   REAL, PARAMETER :: c3=-0.060275957444801354000
   REAL, PARAMETER :: c4= 0.005112428730167831000
   REAL, PARAMETER :: d0= 0.835543536679762600000
   REAL, PARAMETER :: d1= 0.047821976622976340000
   REAL, PARAMETER :: d2= 0.000053594881446931025
   REAL, PARAMETER :: d3=-0.000268040997573199600
   REAL, PARAMETER :: d4= 0.000015765134162582942
   REAL :: x2, lx, xi
   IF ( x .LE. XMIN) THEN                          ! x < xmin=0.15: to 0.1%
      x2=x*x                                       ! ln(x)/2 + 3*GAMMA/4 -1/4
      lx=LOG(x)                                    ! -[ZETA(3)/2]*x^2
      kqm3=0.5E0*lx + a0 + a2*x2                   !
                                                   !
                                                   !
   ELSEIF ( x .GE. XMAX ) then                     ! x > xmax=2.5: to 0.25%
      xi=1/x                                       ! -1/12*x^2 - 1/40*x^4 -
      x2=xi*xi                                     ! 3/64*x^6
      kqm3=b6                                      !
      kqm3=b4 + kqm3*x2                            !
      kqm3=b2 + kqm3*x2                            !
      kqm3=kqm3*x2                                 !
   ELSE
      xi=1/x                                       ! xmin < x < xmax
      lx=LOG(x)                                    ! fit accurate to 0.04%
      kqm3=c4                                      ! c0 + c1*x + c2*x^2 +
      kqm3=c3+kqm3*x                               ! c3*x^3 + c4*x^4 +
      kqm3=c2+kqm3*x                               ! d0*ln(x) +
      kqm3=c1+kqm3*x                               ! d1/x + d2/x^2 +
      kqm3=c0+kqm3*x + d0*lx                       ! d3/x^3 + d4/x^4
      lx=d4                                        !
      lx=d3+lx*xi                                  !
      lx=d2+lx*xi                                  !
      lx=d1+lx*xi                                  !
      lx=lx*xi                                     !
      kqm3=kqm3+lx
   ENDIF
 END FUNCTION kqm3

!
! classical contribution:
!
!
! This source code calculates the classical dedxc for each
! plasma component. It returns dedxctot and dedxcsumi. The
! following functions and subroutines are defined here:
!
! Subroutine: dedxc
!
! Function  : intone
!
! Function  : inttwo
!
! Function  :  hi
```

```
!
! Function  : j1, j2, j3 ,j4
!
! Subroutine: fri
!
! Function  : daw
!
 SUBROUTINE dedxc(vp, dedxctot, dedxcsumi)
   USE globalvars

   IMPLICIT NONE
   REAL, INTENT(IN) :: vp
   REAL, INTENT(OUT):: dedxctot, dedxcsumi

   REAL, PARAMETER        :: ABV20=1.E-4
   REAL, DIMENSION(1:NNB) :: abv, abv2, clb
   REAL                   :: abv2ib, ss, vp2, kd2, kd4
   REAL                   :: hi, inttwo, intone, a, b, c, ke
   INTEGER                :: ib
!
! define input variables
!
   vp2=vp*vp
   abv=ab*vp      ! for intone, inttwo, hi
   clb=0          ! initialize classical to zero
   DO ib=1,NNB    ! loop over plasma components
      IF ( lzb(ib) ) THEN ! computle only if zb(ib) /= 0
         abv2=abv*abv                 !
         abv2ib=abv2(ib)              ! cut on each component
         IF (abv2ib .LT. ABV20) THEN  ! small velocity limit is analytic
            kd2 =kd*kd
            kd4 =kd2*kd2
            clb(ib)=clb(ib)+2*gb(ib)*(1-abv2ib*(1 + &
                 (2./3.)*(mp0/mb0(ib))))
            clb(ib)=clb(ib)-(4./3.)*abv2ib -2*SUM(kb2*abv2)/kd2
            ss=(SUM(kb2*abv))
            ss=ss*ss
            clb(ib)=clb(ib)+(PI/6.)*ss/kd4
            clb(ib)=clb(ib)*cp1*kb2(ib)*eb(ib)/vp
         ELSE                  ! general velocities are numerical
!
! int1: dedxc=(cp1/vp)*intone(abv,bbv,kb2,eb,cb)
!
            a=abv(ib)
            b=bb(ib)*vp2
            c=cb(ib)
            ke=kb2(ib)*eb(ib)
            clb(ib)=clb(ib)+(cp1/vp)*ke*intone(a,b,c)
!
! int2: dedxc=dedxc + cp2*inttwo(abv,kb2)
!
            clb(ib)=clb(ib)+cp2*inttwo(ib,abv)
!
! H: dedxc=dedxc - (cp3/vp2)*h(abv,kb2)
!
            clb(ib)=clb(ib)-(cp3/vp2)*hi(ib,abv)/rb(ib)
         ENDIF
```

```
      ELSE
         clb(ib)=0  ! don't compute when zb(ib) = 0
      ENDIF
   ENDDO
   dedxcsumi=SUM(clb(2:NNB))
   dedxctot =clb(1)
   dedxctot =dedxctot+dedxcsumi

   dedxctot =CONVFACT*dedxctot   ! convert to MeV/mu-m
   dedxcsumi=CONVFACT*dedxcsumi  !
 END SUBROUTINE dedxc

!
! Much of this integration was performed analytically,
! and the rest can be expressed in terms of Hypergeometric
! functions j1 ... j4 (which we in turn fit)
!
 FUNCTION intone(a, b, c)
   USE globalvars

   IMPLICIT NONE
   REAL, INTENT(IN) :: a, b, c
   REAL             :: intone

   REAL    :: bc, a2, a3, erfa, expa, ferf
   REAL    :: j1, j2, j3, j4
   bc=b*c
   a2=a*a
   a3=a2*a
   erfa=SQPI*ferf(a)  ! see ferf.f
   expa=EXP(-a2)
   intone=erfa*((2-c)/a + bc/(2*a3))-bc*expa/a2
   intone=intone + j3(a2) - j1(a2) + b*(j2(a2) -j4(a2))
 END FUNCTION intone

!
!              /1
!              |
! inttwo = 2 * | du u*H(u*abv)
!              |
!              /0
!
! This subroutine performs the integration numerically by
! Gaussian Quadrature. The number of intervals NG that [0,1]
! is broken into is hardwired in PARAMETER statement, but
! it can be changed by the user (must be even).
!
! NOTE on Gaussian Quadrature:
!
! The polynomial P3(x)=(5*x^3-3*x)/2 is employed, and I have
! defined the appropriate weights W13, W2 and relative position
! UPM in parameter statements.
!
 FUNCTION inttwo(ib, abv)
   USE globalvars

   IMPLICIT NONE
```

```
   REAL, DIMENSION(1:NNB), INTENT(IN) :: abv
   INTEGER,                INTENT(IN) :: ib
   REAL                               :: inttwo

   REAL, PARAMETER :: UPM=0.7745966692E0
   REAL, PARAMETER :: W13=0.5555555556E0, W2=0.8888888889E0

   INTEGER, PARAMETER :: NG=2000  ! NG must be even
   REAL,    PARAMETER :: U0=0.E0, U1=1.E0, DU=1.E0/NG

   REAL    :: u, um, hi, uu(NNB)
   INTEGER :: iu, ibb

   inttwo=0
   u=U0-DU
   DO iu=1,NG,2
!
      u=u+2.E0*DU
      DO ibb=1,NNB
         uu(ibb)=u*abv(ibb)
      ENDDO
      inttwo=inttwo+W2*u*hi(ib,uu)
!
      um=u-DU*UPM
      DO ibb=1,NNB
         uu(ibb)=um*abv(ibb)
      ENDDO
      inttwo=inttwo+W13*um*hi(ib,uu)
!
      um=u+DU*UPM
      DO ibb=1,NNB
         uu(ibb)=um*abv(ibb)
      ENDDO
      inttwo=inttwo+W13*um*hi(ib,uu)
   ENDDO
   inttwo=2*inttwo*DU
 END FUNCTION inttwo

!
! The ib-th component of Hb is hi=Sqrt[Pi]*kb^2*xb*Exp[-xb^2]*H/Fi
! with Fi=Sqrt[Pi]Sum_c kc^2*xc*Exp[-xc^2] where xb=abv(ib) and
! kb^2=kkb2(ib)
!
 FUNCTION hi(ib, abv)
   USE globalvars
   IMPLICIT NONE
   REAL,    DIMENSION(1:NNB), INTENT(IN) :: abv
   INTEGER,                   INTENT(IN) :: ib
   REAL                                  :: hi
   REAL, PARAMETER :: EPS=1.E-6, XB2MAX=10.D0
   REAL    :: fr, fi, fabs, farg, h, ebc
   REAL    :: xb, xb2, xc, xc2, xbc
   INTEGER :: ic
   CALL fri(abv, fr, fi, fabs, farg)
   h=-2*(fi*LOG(fabs/K**2) + fr*farg)
   xb =abv(ib)
   xb2=xb*xb
```

```
!
! Express Fi in terms of its sum and write
!
!   hi = kb^2*xb*Exp[-xb^2]/Sum_c kc^2*xc*Exp[-xc^2]
!
! then calculate hi^-1 first, excluding kb^2*xb since
! some components of kb^2*xb might be zero.
   hi=0
   DO ic=1,NNB
      xc =abv(ic)
      xc2=xc*xc
      xbc=xb2-xc2
      IF (xbc .LT. XB2MAX) THEN
         ebc=EXP(xb2-xc2)
      ELSE
         hi=0    ! ebc is large and dominates the sum,
         RETURN  ! and its inverse is almost zero
      ENDIF
      hi=hi + kb2(ic)*xc*ebc
   ENDDO
   hi=kb2(ib)*xb/hi  ! invert and include kb^2*xb
   hi=h*hi
 END FUNCTION hi

!
!         /1
!         |               ln[1-u]
! j1(x) = | du e^{-x u} ---------
!         |               Sqrt[u]
!         /0
!
 FUNCTION j1(x)
!
! see globalvars for parameters
!
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL             :: j1

   REAL,    PARAMETER   :: XMIN=0.1D0, XMAX=20.D0
   REAL    :: x2, x4
   REAL    :: xx, ra, rc
   REAL    :: y, y3, y5
   INTEGER :: n
!
! analytic asymptotic forms
!
  IF (x .LE. XMIN) THEN
     j1=J1MM*x+J1BB
  ELSEIF (x .GT. XMAX) THEN
     y=SQRT(x)           ! x^1/2
     y3=x*y              ! x^3/2
     y5=y3*x             ! x^5/2
     j1=J1AA/y3 +J1GG/y5
  ELSE
!
```

```
! numerical asymptotic form
!
     x2=x*x
     x4=x**3.5
     j1=(J1MM*x+J1BB)/(J1CC*x4+1) + J1EE*J1AA*x2/(J1EE*x4+1)
!
! spline correction
!
     ra=0.E0
     rc=0.E0
     xx=1.E0
     DO n=0,NMJ1
        ra=ra+J1A(n)*xx
        rc=rc+J1B(n)*xx
        xx=x*xx
     ENDDO
     ra=ra+xx
     rc=rc+xx
     j1=j1*rc/ra
  ENDIF
 END FUNCTION j1
!
!         /1
!         |
! j2(x) = | du e^{-x u} ln[1-u]*Sqrt[u]
!         |
!         /0
!
 FUNCTION j2(x)
!
! see globalvars for parameters
!
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL             :: j2

   REAL,    PARAMETER   :: XMIN=0.1, XMAX=30.D0
   REAL    :: x2, x4
   REAL    :: y, y5, y7
   REAL    :: xx, ra, rc
   INTEGER :: n

!
! analytic asymptotic forms
!
  IF (x .LE. XMIN) THEN
     j2=J2MM*x+J2BB
  ELSEIF (x .GT. XMAX) THEN
     y=SQRT(x)          ! x^1/2
     y5=x*x*y           ! x^5/2
     y7=y5*x            ! x^7/2
     j2=J2AA/y5 +J2GG/y7
  ELSE
!
! numerical asymptotic form
!
```

```
     x2=x*x
     x4=x**4.5
     j2=(J2MM*x+J2BB)/(J2CC*x4+1) + &
        J2EE*J2AA*x2/(J2EE*x4+1)
!
! spline correction
!
     ra=0.E0
     rc=0.E0
     xx=1.E0
     DO n=0,NMJ2
        ra=ra+J2A(n)*xx
        rc=rc+J2B(n)*xx
        xx=x*xx
     ENDDO
     ra=ra+xx
     rc=rc+xx
     j2=j2*rc/ra
  ENDIF
 END FUNCTION j2
!
!         /1
!         |                ln[u]
! j3(x) = | du e^{-x u} --------
!         |              Sqrt[u]
!         /0
!
 FUNCTION j3(x)
!
! see globalvars for parameters
!
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL             :: j3

   REAL,    PARAMETER   :: XMIN=0.4, XMAX=2.3D0
   REAL    :: x2, x3, x4
   REAL    :: xx, ra, rc
   REAL    :: y
   INTEGER :: n
!
! analytic asymptotic forms
!
  IF (x .LE. XMIN) THEN
     j3=J3MM*x+J3BB
  ELSEIF (x .GE. XMAX) THEN
     y=SQRT(x)
     j3=(J3AA1*LOG(x) + J3AA2)/y
  ELSE
!
! numerical asymptotic form
!
     x2 =x*x
     x3 =x**2.5
     x4 =x**3.5
     j3=(J3MM*x+J3BB)/(J3CC*x4+1) + &
```

```
        J3EE*(J3AA1*LOG(1+x)+J3AA2)*x2/(J3EE*x3+1)
!
! spline correction
!
     ra=0.E0
     rc=0.E0
     xx=1.E0
     DO n=0,NMJ3
        ra=ra+J3A(n)*xx
        rc=rc+J3B(n)*xx
        xx=x*xx
     ENDDO
     ra=ra+xx
     rc=rc+xx
     j3=j3*rc/ra
  ENDIF
END FUNCTION j3
!
!         /1
!         |
! j4(x) = | du e^{-x u} ln[u]*Sqrt[u]
!         |
!         /0
!
 FUNCTION j4(x)
!
! see globalvars for parameters
!
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL             :: j4

   REAL,    PARAMETER   :: XMIN=0.18D0, XMAX=4.7D0
   REAL    :: x2, x4
   REAL    :: xx, ra, rc
   REAL    :: y, y3
   INTEGER :: n
!
! analytic asymptotic forms
!
  IF (x .LE. XMIN) THEN
     j4=J4MM*x+J4BB
  ELSEIF (x .GE. XMAX) THEN
     y=SQRT(x)                    ! x^1/2
     y3=x*y                       ! x^3/2
     j4=(J4AA1*LOG(x) + J4AA2)/y3
  ELSE
!
! numerical asymptotic form
!
     x2 =x*x
     x4 =x**3.5
     j4=(J4MM*x+J4BB)/(J4CC*x4+1) + &
         J4EE*(J4AA1*LOG(1+x)+J4AA2)*x2/(J4EE*x4+1)
!
! spline correction
```

```
!
     ra=0.E0
     rc=0.E0
     xx=1.E0
     DO n=0,NMJ4
        ra=ra+J4A(n)*xx
        rc=rc+J4B(n)*xx
        xx=x*xx
     ENDDO
     ra=ra+xx
     rc=rc+xx
     j4=j4*rc/ra
  ENDIF
 END FUNCTION j4

!
! Returns the dielectric function F in terms of the
! real part, the imaginary part, the absolute value,
! and the argument: fr, fi, fabs, farg
!
 SUBROUTINE fri(xb, fr, fi, fabs, farg)
   USE globalvars
   IMPLICIT NONE
   REAL, DIMENSION(1:NNB), INTENT(IN)  :: xb
   REAL,                   INTENT(OUT) :: fr,  fi, fabs, farg
   REAL    :: x, daw, d
   INTEGER :: ib
   fr=0
   fi=0
   DO ib=1,NNB
      x=xb(ib)
      d=daw(x)
      fr=fr+(kb2(ib)*(1-2*x*d))
      fi=fi+kb2(ib)*x*EXP(-x*x)
   ENDDO
   fi=fi*SQPI
   fabs=SQRT(fr*fr + fi*fi)
   farg=ATAN2(fi,fr)
 END SUBROUTINE fri

!
! The Dawson function takes the form
!
!
!          /x
!          |
! daw(x) = | dy e^{y^2 -x^2}
!          |
!          /0
!
!        =(sqrt(pi)/2)*exp(-x^2)*erfi(x)
!
!
! For small x < XMIN we use the asymptotic form
!
!                 2x^3     4x^5
! daw(x) =  x  +  ----  + ----- + O(x^7)
```

```
!                  3        15
!
! and for large x > XMAX we use
!
!            1     1       3
! daw(x) =  --- + ---- + ----- + O(x^-7)
!           2x    4x^3    8x^5
!
! The error is 0.03% for XMIN=0.4 and 0.01% XMAX=5.0. For
! intermediate values, we approximate daw(x) as a rational
! function of the form
!
!              x      x^6+b5*x^5+b4*x^4+b3*x^3+b2*x^2+b1*x+b0
! daw(x) = --------- ----------------------------------------
!          2 x^2 + 1  x^6+a5*x^5+a4*x^4+a3*x^3+a2*x^2+a1*x+b0
!
!
! With the values of bn and an chosen below, the error is 0.03%.
!
 FUNCTION daw(x)
!
! As x->0 and x->infty, the Dawson function takes the asymptotic
! forms daw(x)~x and daw(x)~1/(2x), respectively. The first
! rational function "R(x)=x/(2x**2+1)" reproduces this behavior;
! the 6-th order polynomial-ratio Q6(x) asymptotes to one at both
! ends (to preserve the asymptotic form of the previous function),
! with the coefficients a5, b5, ... b0 being chosen to provide
! agreement with the exact Dawson integral at the values:
!
! x0=0.92413 daw(x0)=0.541044
! x1=0.2     daw(x1)=0.194751
! x2=0.5     daw(x2)=0.424436
! x3=0.7     daw(x3)=0.510504
! x4=1.2     daw(x4)=0.507273
! x5=1.4     daw(x5)=0.456507
! x6=1.6     daw(x6)=0.399940
! x7=2.0     daw(x7)=0.301340
! x8=3.0     daw(x8)=0.178271
! x9=4.0     daw(x9)=0.129348
! x10=8.0    daw(x10)=0.0630002
!
! See daw.nb for details.
!
   USE globalvars
   IMPLICIT NONE
   REAL, INTENT(IN) :: x
   REAL  daw

   REAL,    PARAMETER  :: XMIN=0.4D0, XMAX=5.D0
   REAL    :: x3, x5, xx, ra, rc
   INTEGER :: n
   IF (x .LE. XMIN) THEN
      x3=x*x*x
      x5=x3*x*x
      daw=x - 2.D0*x3/3.D0 + 4.D0*x5/15.D0
   ELSEIF (x .GE. XMAX) THEN
      x3=x*x*x
```

```
      x5=x3*x*x
      daw=1.D0/(2.D0*x)+1.D0/(4.D0*x3)+3.D0/(8.D0*x5)
   ELSE
      ra=0.E0
      rc=0.E0
      xx=1.E0
      DO n=0,NMDAW
         ra=ra+DWA(n)*xx
         rc=rc+DWB(n)*xx
         xx=x*xx
      ENDDO
      ra=ra+xx
      rc=rc+xx
      daw=x/(1.E0+2.E0*x*x)
      daw=daw*rc/ra
   ENDIF
 END FUNCTION daw
```

[1] "Charged Particle Motion in a Highly Ionized Plasma", L. Brown, D. Preston, and R. Singleton Jr., LA-UR-042713, *Physics Reports*, **410/4**, 237 (2005), [arXiv:physics/0501084].

[2] Section 29 of "Physical Kinetics", E. M. Lifshitz and L. P. Pitaevskii, Pergamon Press, Oxford, 1981.

[3] "Handbook of Mathematical Functions", M. Abramowitz and I. A. Stegun, Dover Publications, 9th Edition, 1972.