PSFC/RR-00-9

A 1-D Space, 2-D Velocity, Kinetic Neutral Transport Algorithm

B. LaBombard

November 2000

Plasma Science and Fusion Center
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

This work was supported by the U.S. Department of Energy, Cooperative Grant No. DE-FC02-99ER54512. Reproduction, translation, publication, use and disposal, in whole or in part, by or for the United States government is permitted.

A 1-D Space, 2-D Velocity, Kinetic Neutral Transport Algorithm

11/16/2000

B. LaBombard

Massachusetts Institute of Technology, Plasma Science and Fusion Center,

175 Albany St., Cambridge, MA 02139 USA

Abstract

A simple numerical algorithm has been written in IDL[1] to compute the hydrogen or deuterium atomic neutral distribution function, f_n, in a slab geometry for inputted plasma profiles (density, ion and electron temperature). The distribution function is described in terms of two velocity components (making use of the rotational symmetry in velocity space about the *x*-axis) and one spatial component (*x*). The velocity distribution of neutrals entering the slab (positive velocity) and the neutral density at the edge of the slab are prescribed boundary conditions. The algorithm constructs f_n by summing up successive generations of charge-exchange neutrals. Rates for electron impact ionization and charge exchange of hydrogen or deuterium atoms are computed using data compiled by Janev [2]. The output of the computation is $f_n(v_x, v_r, x)$ and velocity moments of f_n including neutral density, fluid velocity, temperature, pressure, diagonal elements of the stress tensor, and heat fluxes. In addition, the spatial profiles of the net rate of energy and *x*-directed momentum transfer between the ion and neutral species are outputted. A number of numerical consistency checks are performed in the code involving mesh size limitations and errors associated with discrete representation of ion and neutral distribution functions. The source code is entirely written in IDL and is freely available to the community.

1. Method

1.1 Computational Domain, Inputted Conditions

The parameter to be computed is the atomic neutral distribution function, f_n, over spatial region $[x_a, x_b]$. It is assumed that f_n has rotational symmetry about the v_x axis so that it can be described in terms of two velocity coordinates and one spatial coordinate, $f_n = f_n(v_x, v_r, x)$, with $v_r^2 = v_y^2 + v_z^2$

The computational domain is specified by input parameters as

$$-v_{x,\max} \le v_x \le v_{x,\max}, \quad v_{r,\min} \le v_r \le v_{r,\max}, \quad x_a \le x \le x_b$$

The boundary conditions and constraints on the neutral distribution function are:

$f_n(v_x > 0, v_r, x_a)$- specified (input)

$f_n(v_x < 0, v_r, x_b) = 0$

$\frac{\partial f_n}{\partial x}(v_x = 0, v_r, x)$ - finite

Background plasma conditions are inputted over spatial region $[x_a, x_b]$:

$n(x)$ - plasma density profile

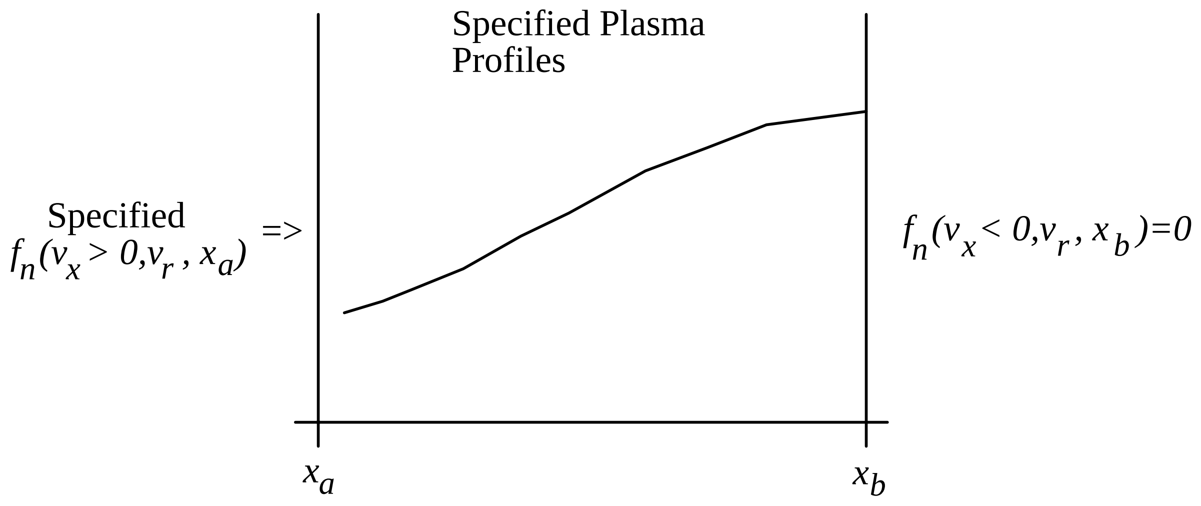
$T_e(x)$ - electron temperature profile
$T_i(x)$ - ion temperature profile

1.2 Atomic Physics

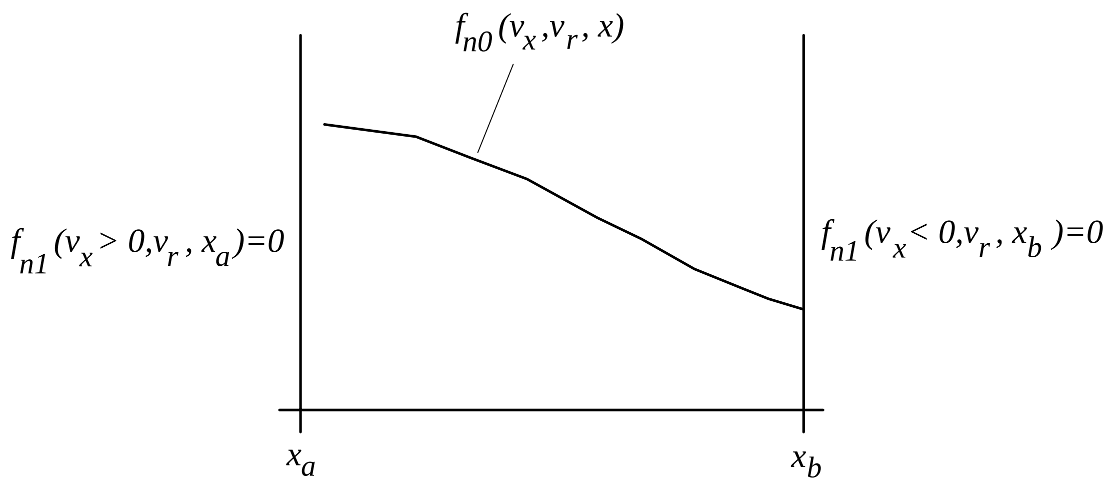
Only ionization and charge exchange processes from the ground state of atomic hydrogen or deuterium in respective hydrogen or deuterium plasmas are considered.

1.3 Overview of Computational Method

The background plasma profiles of density, electron temperature and ion temperature is specified. The incident distribution function of atomic neutrals, $f_n(v_x > 0, v_r, x_a)$, attacking plasma is also specified:


Specified Plasma Profiles
Specified $f_n(v_x > 0, v_r, x_a)$ =>
$f_n(v_x < 0, v_r, x_b) = 0$
x_a
x_b

First, the portion of the neutral distribution function which continues to penetrate the plasma without ionization or charge exchange is computed. This is designated as $f_{n0}(v_x, v_r, x)$, or the '0th generation'.


$f_{n0}(v_x, v_r, x)$
$f_{n1}(v_x > 0, v_r, x_a) = 0$
$f_{n1}(v_x < 0, v_r, x_b) = 0$
x_a
x_b

Then the neutral distribution function arising from '1st generation' charge exchange, $f_{n1}(v_x, v_r, x)$, is computed by considering the three cases of (A) $v_x > 0$, (B) $v_x < 0$ and (C) $v_x = 0$.

A. $f_{n1}(v_x, v_r, x)$ for $v_x > 0$ is determined by integrating the Boltzmann equation over $[x_a, x]$ and applying the boundary condition of $f_{n1}(v_x > 0, v_r, x_a) = 0$. Here the charge exchange 'source' term only includes the contribution from the previous generation (0th generation).

B. $f_{n1}(v_x, v_r, x)$ for $v_x < 0$ is determined by integrating the Boltzmann equation over $[x, x_b]$ and applying the boundary condition of $f_{n1}(v_x < 0, v_r, x_b) = 0$. Again the charge exchange 'source' term only includes the contribution from the previous generation (0th generation).

C. $f_{n1}(v_x, v_r, x)$ for $v_x = 0$ is computed from the Boltzmann equation for the special case of $v_x = 0$, making use of the 'regularity condition' $\frac{\partial f_n}{\partial x}(v_x = 0, v_r, x)$ - finite

The above calculations are repeated for subsequent charge exchange generations, $f_{n2}(v_x, v_r, x)$, $f_{n3}(v_x, v_r, x)$, …The total neutral distribution function in the plasma is then computed by summing over a finite number of charge-exchange generations:

$$f_n(v_x, v_r, x) = \sum_j f_{nj}(v_x, v_r, x)$$

(Note: now $f_n(v_x < 0, v_r, x_a)$ and $f_n(v_x > 0, v_r, x_b)$ is also known, i.e. the albedo of the plasma has been determined.)

2. Formulation

A. Boltzmann Equation

The neutral distribution function satisfies the Boltzmann equation:

$$\begin{aligned} v_x \frac{\partial f_n}{\partial x} = & \hat{f}_i n_i \int f_n(\underline{v}', x) |\underline{v} - \underline{v}'| \sigma_{cx}(|\underline{v} - \underline{v}'|) \partial^3 v' \\ & - f_n n_i \int \hat{f}_i(\underline{v}', x) |\underline{v} - \underline{v}'| \sigma_{cx}(|\underline{v} - \underline{v}'|) \partial^3 v' - f_n n_e \langle \sigma v \rangle_{ion} \end{aligned} \quad (1)$$

which makes use of the assumed spatial symmetries, $\frac{\partial f_n}{\partial y} = 0$ and $\frac{\partial f_n}{\partial z} = 0$. Integration is over all velocity space. $\hat{f}_i(\underline{v}, x)$ is the velocity space distribution function of ions normalized so that $\int \hat{f}_i(\underline{v}', x)\, \partial^3 v' = 1$. For ionization, we have made use of the fact that the neutrals are nearly stationary relative to the electrons, $v_n << v_e$, replacing the velocity-space integral with the ionization rate, $n_e \langle \sigma v \rangle_{ion}$. Figure 1 shows $\langle \sigma v \rangle_{ion}$ evaluated for a range of electron temperatures (from Janev [2]).

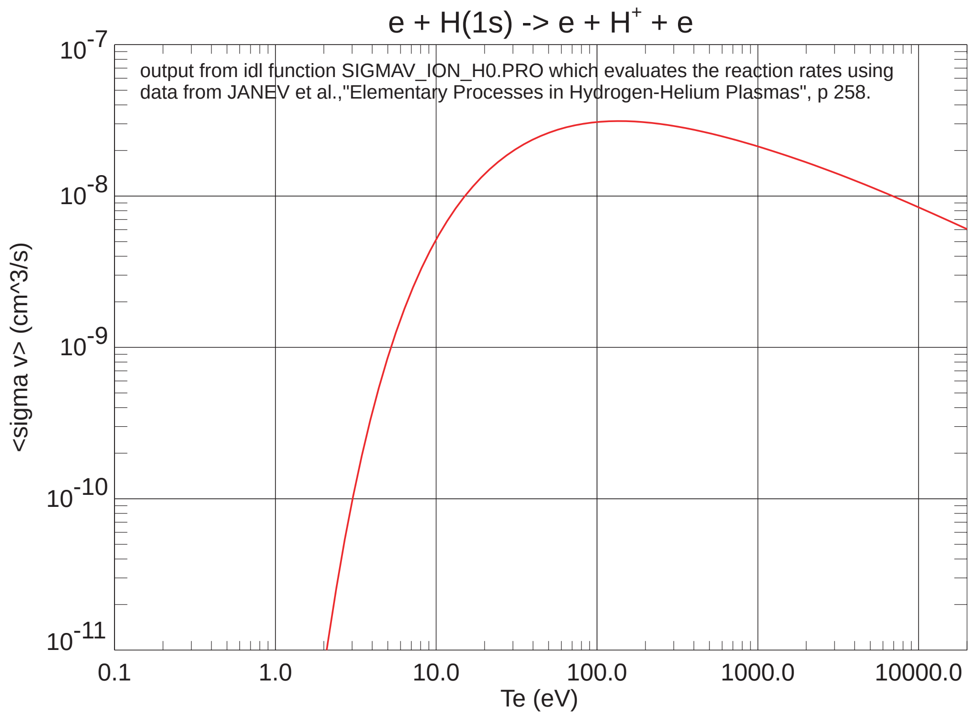


Fig. 1 - $\langle \sigma v \rangle_{ion}$ as a function of Te.

The second integral on the right hand side of Eq. (1) depends on the ion temperature and the energy of the neutral species and can be written in the form

$$\int \hat{f}_i(\underline{v}', x) |\underline{v} - \underline{v}'| \sigma_{cx}(|\underline{v} - \underline{v}'|) \partial^3 v' = \langle \sigma v \rangle_{cx}[T_i, E_0], \qquad (2)$$

where $\langle \sigma v \rangle_{cx}[T_i, E_0]$ is the hydrogen charge exchange cross-section averaged over a Maxwellian hydrogen ion distribution (temperature, T_i), accounting for the energy of the hydrogen neutral species, E_0, which for a specified neutral velocity, (v_x, v_r), is $E_0 = \frac{1}{2} m_H (v_x^2 + v_r^2)$. Figure 2 shows values of $\langle \sigma v \rangle_{cx}[T_i, E_0]$ and Fig.3 shows values of $\sigma_{cx}[E_0]$, also tabulated by Janev [2].

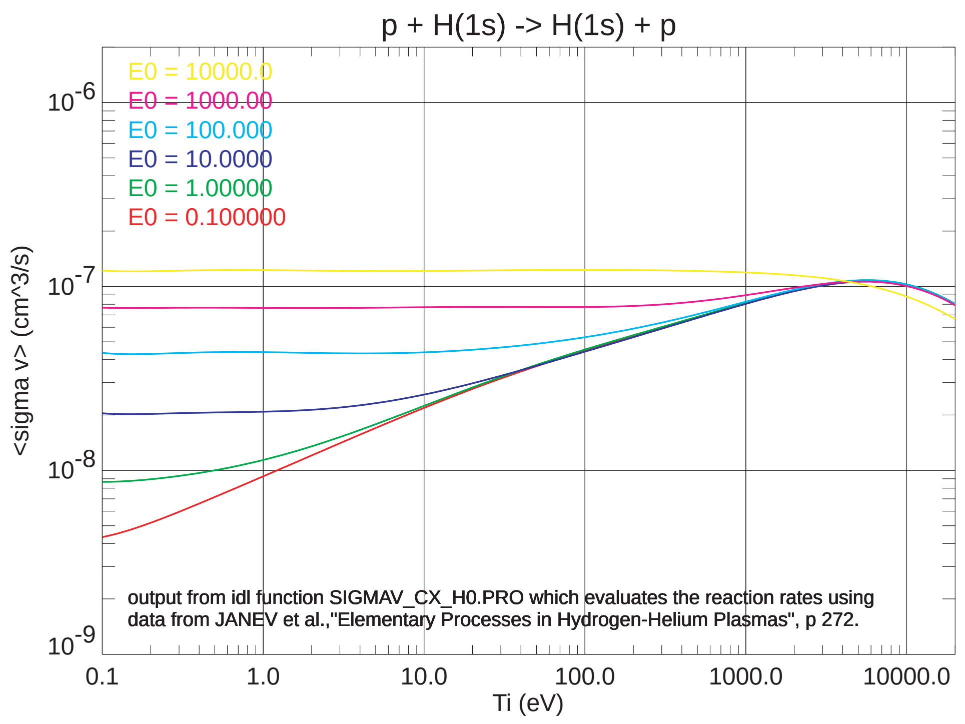


Fig. 2 – Proton charge exchange rate as a function of ion temperature and neutral energy.

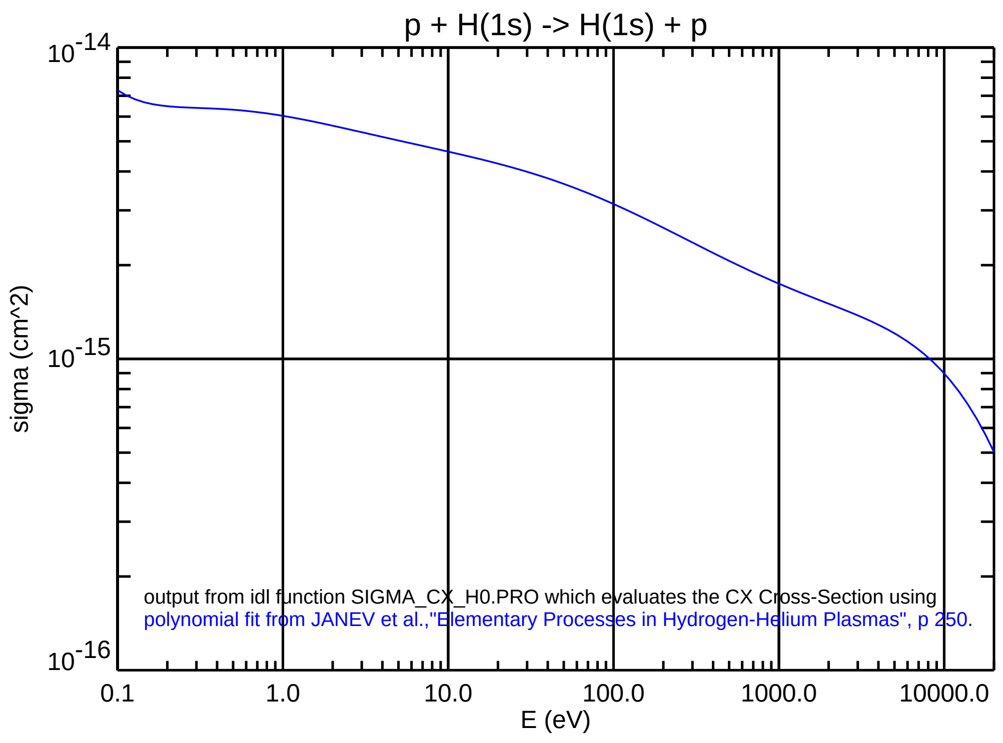


Fig. 3 – Proton charge exchange cross section as a function of neutral energy (evaluated using relative proton and neutral velocities).

The first integral on the right hand side of Eq. (1) is not so readily evaluated in terms of tabulated values. It requires a velocity-space integral to be performed over (v_x', v_r') for each combination of (v_x, v_r), weighted by $f_n(v_x', v_r', x)$. This integral computes the charge exchange frequency for a specified combination of (v_x, v_r). For large velocity space meshes, this computation can take some time. Therefore two options for computing this integral are included in this algorithm: (A) an 'exact' integration and (B) an approximation of the integral. For option (B) the approximation,

$$\begin{gathered}\hat{f}_i n_i \int f_n(\underline{v}', x) |\underline{v} - \underline{v}'| \sigma_{cx}(|\underline{v} - \underline{v}'|) \partial^3 v' \approx \\ \hat{f}_i n_i \iint \hat{f}_i f_n(\underline{v}', x) |\underline{v}'' - \underline{v}'| \sigma_{cx}(|\underline{v}'' - \underline{v}'|) \partial^3 v' \, \partial^3 v'' = \\ \hat{f}_i n_i \int f_n(\underline{v}', x) \langle \sigma v \rangle_{cx} [T_i, E_0'] \partial^3 v' \end{gathered} \quad , \tag{3}$$

is made, replacing the charge exchange frequency for a specified combination of (v_x, v_r) with the velocity-spaced average charge exchange frequency [which is independent of (v_x, v_r)]. In effect, this option assumes that all charge exchange neutrals are 'born' with a distribution that is the same as the ion distribution function. As a result, distortions in this source distribution function due to the charge exchange rate being different at different values of relative ion-neutral velocities are not taken into account.

We are interested in using the data from Fig. 2 to compute the charge exchange rate both for hydrogen neutrals in a hydrogen plasma and deuterium neutrals in a deuterium plasma. The charge exchange cross-sections, σ_{cx}, are virtually identical for hydrogen and deuterium and depend only on the relative velocities of the interacting ions and atoms. The Maxwellian-averaged deuterium cross sections can therefore be related to the hydrogen case by accounting for the mass of the deuterium nuclei relative to the mass of hydrogen,

$$\langle \sigma v \rangle_{cx} [T_i, E_0] \rightarrow \langle \sigma v \rangle_{cx} [\frac{T_i}{\mu}, \frac{1}{2} m_H \left(v_x^2 + v_r^2 \right)] \quad . \tag{4}$$

Here T_i is still the temperature of the ion species and $\left(v_x, v_r\right)$ the neutral velocities but the factor, μ, (=1 for hydrogen and 2 for deuterium) appropriately scales the ion velocities. Similarly, for direct evaluation of $\sigma_{cx}[E_0]$ using tabulated hydrogen data, the relative energy is evaluated using the proton mass for both hydrogen and deuterium cases.

It is useful to define the following two quantities which can be evaluated directly for all (v_x, v_r, x) mesh locations using the inputted plasma density and temperature profiles with $n = n_i = n_e$:

Charge exchange sink rate (option A - direct evaluation from $\sigma_{cx}(E_0)$*)*

$$\boxed{\begin{aligned}\alpha_{cx}(v_x,v_r,x) &\equiv n\iint \hat{f}_i(v'_x,v'_r,x)\left[\int |\underline{v}-\underline{v}'|\sigma_{cx}(|\underline{v}-\underline{v}'|)\partial\theta'\right]v'_r\partial v'_r\partial v'_x \\ &\equiv n\iint \hat{f}_i(v'_x,v'_r,x)\Sigma_{cx}(v_x,v_r,v'_x,v'_r)\partial v'_x\partial v'_r\end{aligned}} \quad (5a)$$

with

$$\boxed{\Sigma_{cx}(v_x,v_r,v'_x,v'_r) \equiv v'_r\int |\underline{v}-\underline{v}'|\sigma_{cx}(|\underline{v}-\underline{v}'|)\partial\theta'.} \quad (5a')$$

Charge exchange sink rate (option B - using $\langle\sigma v\rangle_{cx}$*)*

$$\boxed{\alpha_{cx}(v_x,v_r,x) \equiv n\langle\sigma v\rangle_{cx}[\frac{T_i}{\mu},\frac{1}{2}m_H\left(v_x^2+v_r^2\right)]} \quad (5b)$$

Total sink rate

$$\boxed{\alpha_c(v_x,v_r,x) \equiv \alpha_{cx}(v_x,v_r,x)+n\langle\sigma v\rangle_{ion}[T_e]} \quad (6)$$

It is also useful to define:

Charge exchange source rate (option A)

$$\boxed{\beta_{cx}(v_x,v_r,x) \equiv \hat{f}_i n\iint f_n(v'_x,v'_r,x)\Sigma_{cx}(v_x,v_r,v'_x,v'_r)\partial v'_x\partial v'_r} \quad (7a)$$

Charge exchange source rate (option B)

$$\boxed{\begin{aligned}\beta_{cx}(v_x,v_r,x) &\equiv \hat{f}_i n\iiint f_n(v_x,v_r,x)|\underline{v}-\underline{v}'|\sigma_{cx}(|\underline{v}-\underline{v}'|)v_r\partial v_r\partial v_x\partial\theta \\ &\approx 2\pi\,\hat{f}_i\iint f_n(v'_x,v'_r,x)\alpha_{cx}(v'_x,v'_r,x)v'_r\partial v'_x\partial v'_r\end{aligned}} \quad (7b)$$

Now the Boltzmann equation becomes

$$v_x\frac{\partial f_n}{\partial x} = \beta_{cx}-\alpha_c f_n. \quad (8)$$

2.2 Expanding f_n as a Series of Neutral Charge Exchange 'Generations'

We now consider the neutral distribution function to be composed of a sum of sub-distribution functions of neutrals which have undergone *j* charge exchange collisions

$$\begin{aligned} f_n(v_x, v_r, x) &= \sum_j f_{nj}(v_x, v_r, x) \\ \beta_{cx}(v_x, v_r, x) &= \sum_j \beta_{cxj}(v_x, v_r, x) \end{aligned} \quad . \tag{9}$$

With the above definitions, Eq. (1) can be written as a series of separate Boltzmann equations, each representing the kinetic balance for that charge-exchange population of neutrals:

$$\begin{aligned} v_x \frac{\partial f_{n0}}{\partial x} &= \qquad -\alpha_c f_{n0} \\ v_x \frac{\partial f_{n1}}{\partial x} &= \beta_{cx0} - \alpha_c f_{n1} \\ v_x \frac{\partial f_{n2}}{\partial x} &= \beta_{cx1} - \alpha_c f_{n2} \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ v_x \frac{\partial f_{nj}}{\partial x} &= \beta_{cxj-1} - \alpha_c f_{nj} \end{aligned} \tag{10}$$

2.3 Numerical Grid & Scheme

The spatial and velocity grid coordinates are specified with arbitrary spacing:

$$\begin{aligned} v_x &= [-v_{x\max}, v_{x1}, ..., 0, ..., v_{xk}, v_{xk+1}, ..., v_{x\max}] \\ v_r &= [v_{r\min}, v_1, ..., v_{rl}, v_{xl+1}, ..., v_{r\max}] \\ x &= [x_a, x_1 ..., x_m, x_{m+1}, ..., x_b] \end{aligned}$$

We will be integrating Eqs. (10) along the x coordinate to obtain values for f_{nj} at each grid location. Therefore we need to approximate the RHS of Eqs. (10) between grid points. For integration along the x coordinate between x_m and x_{m+1}, we will replace the integrand with the average of its values evaluated at x_m and x_{m+1}.

2.3.1 Mesh Equation - 0th Generation

In this scheme, the mesh equation for the 0th generation (with $v_x > 0$) becomes

$$\frac{f_{n0,m+1} - f_{n0,m}}{x_{m+1} - x_m} = -\frac{1}{2v_x}(\alpha_{c,m+1} f_{n0,m+1} + \alpha_{c,m} f_{n0,m})$$

$$f_{n0,m+1}(1+\frac{x_{m+1}-x_m}{2v_x}\alpha_{c,m+1}) = f_{n0,m}(1-\frac{x_{m+1}-x_m}{2v_x}\alpha_{c,m})$$

$$f_{n0,m+1} = f_{n0,m}\frac{2v_x-(x_{m+1}-x_m)\alpha_{c,m}}{2v_x+(x_{m+1}-x_m)\alpha_{c,m+1}}\ . \qquad (11)$$

With the definition,

$$\boxed{A_m \equiv \frac{2v_x-(x_{m+1}-x_m)\alpha_{c,m}}{2v_x+(x_{m+1}-x_m)\alpha_{c,m+1}}, \qquad (12)}$$

the mesh equation for the 0th generation becomes

$$\boxed{f_{n0,m+1} = f_{n0,m}A_m. \qquad (13)}$$

With the boundary condition, $f_{n0}(v_x>0,v_r,x_a) = f_n(v_x>0,v_r,x_a)$ -> specified, this equation yields the 0th generation distribution function over the mesh. Note that Eq. (11) [and Eq.(18) below] indicates that the x mesh spacing must be small enough to avoid nonsensical negative distribution functions. The mesh spacing must satisfy

$$(x_{m+1}-x_m) < \frac{2|v_x|}{\max[\alpha_{c,m},\alpha_{c,m+1}]}. \qquad (14)$$

Thus the maximum allowed x grid spacing is related to the magnitude of the smallest non-zero grid element in the v_x axis.

2.3.1 Mesh Equation - jth Generation (j > 0) for $v_x \neq 0$

In this scheme, the mesh equation for the *j*th generation (with $v_x \neq 0$) becomes

$$\frac{f_{nj,m+1}-f_{nj,m}}{x_{m+1}-x_m} = \frac{1}{2v_x}(\beta_{cxj-1,m+1}+\beta_{cxj-1,m}) - \frac{1}{2v_x}(\alpha_{c,m+1}f_{nj,m+1}+\alpha_{c,m}f_{nj,m})$$

$$f_{nj,m+1}(1+\frac{x_{m+1}-x_m}{2v_x}\alpha_{c,m+1}) = f_{nj,m}(1-\frac{x_{m+1}-x_m}{2v_x}\alpha_{c,m}) + \frac{x_{m+1}-x_m}{2v_x}(\beta_{cxj-1,m+1}+\beta_{cxj-1,m})$$

$$f_{nj,m+1} = f_{nj,m}\frac{2v_x - (x_{m+1}-x_m)\alpha_{c,m}}{2v_x + (x_{m+1}-x_m)\alpha_{c,m+1}} + \frac{(x_{m+1}-x_m)(\beta_{cxj-1,m+1}+\beta_{cxj-1,m})}{2v_x + (x_{m+1}-x_m)\alpha_{c,m+1}}. \qquad (15)$$

With the definition,

$$\boxed{B_m = \frac{(x_{m+1}-x_m)}{2v_x + (x_{m+1}-x_m)\alpha_{c,m+1}} \qquad (16)}$$

the mesh equation for the *j*th generation ($v_x > 0$) becomes

$$\boxed{f_{nj,m+1} = f_{nj,m}A_m + B_m(\beta_{cx\,j-1,m+1}+\beta_{cx\,j-1,m}) \qquad (17)}$$

For the case of $v_x > 0$, the boundary condition $f_{nj}(v_x > 0, v_r, x_a) = 0$ is employed and a recursive use of Eq. (17) yields $f_{nj}(v_x > 0, v_r, x_m)$ for all m.

A useful recursion formula for the case of $v_x < 0$ can be obtained from Eq. (15) by replacing $m+1$ with $m-1$,

$$f_{nj,m-1} = f_{nj,m}\frac{-2v_x - (x_m - x_{m-1})\alpha_{c,m}}{-2v_x + (x_m - x_{m-1})\alpha_{c,m-1}} + \frac{(x_m - x_{m-1})(\beta_{cxj-1,m-1}+\beta_{cxj-1,m})}{-2v_x + (x_m - x_{m-1})\alpha_{c,m-1}} \qquad (18)$$

With the definitions,

$$\boxed{\begin{aligned} C_m &\equiv \frac{-2v_x - (x_m - x_{m-1})\alpha_{c,m}}{-2v_x + (x_m - x_{m-1})\alpha_{c,m-1}} \\ D_m &= \frac{(x_m - x_{m-1})}{-2v_x + (x_m - x_{m-1})\alpha_{c,m-1}} \qquad (19) \end{aligned}}$$

the mesh equation for the *j*th generation ($v_x < 0$) becomes

$$\boxed{f_{nj,m-1} = f_{nj,m}C_m + D_m(\beta_{cx\,j-1,m-1}+\beta_{cx\,j-1,m}) \quad . \qquad (20)}$$

Now, applying the boundary condition $f_{nj}(v_x < 0, v_r, x_b) = 0$, a recursive use of Eq. (20) yields $f_{nj}(v_x < 0, v_r, x_m)$ for all m.

2.2.2 Mesh Equation - jth Generation (j > 0) for $v_x = 0$

For $v_x = 0$, Eqs. (10) yield the direct relationships,

$$\boxed{f_{nj}(0,v_r,x_m) = \frac{\beta_{cxj-1,m}(0,v_r,x_m)}{\alpha_{c,m}}. \qquad (21)}$$

3. Total Neutral Distribution Function

The procedure for computing the neutral distribution function can now be outlined as follows:

A. If option A is used, then compute $\Sigma_{cx}(v_x,v_r,v_x',v_r')$ from Eq. (5a')
B. Compute $\alpha_c(v_x,v_r,x)$ and $\alpha_{cx}(v_x,v_r,x)$ from inputted data using Eqs. (5 a/b) and (6 a/b).
C. Test x grid spacing using Eq. (14).
D. Compute A_m, B_m, C_m, and D_m from Eqs. (12), (16) and (19)
E. Compute $f_{n0}(v_x,v_r,x)$ for $v_x > 0$ from Eq. (13). Note: $f_{n0}(v_x,v_r,x)=0$ for $v_x \le 0$.
F. Compute β_{cx0} from $f_{n0}(v_x,v_r,x)$ and $\alpha_{cx}(v_x,v_r,x)$ using Eq. (7 a/b)
G. Compute $f_{nj}(v_x,v_r,x)$ for the next generation using Eq. (17) for $v_x > 0$, Eq. (20) for $v_x < 0$, and Eq. (21) for $v_x = 0$.
H. Compute β_{cxj} from $f_{nj}(v_x,v_r,x)$ using Eq. (7 a/b)
I. Repeat G and H for a number of generations until $\max(n_{nj}(x))/n_{n0} <$ a specified value
J. Compute the total neutral distribution function from the summation:

$$f_n(v_x,v_r,x) = \sum_j f_{nj}(v_x,v_r,x).$$

4. Numerical Algorithm

An IDL procedure, `Kinetic_Neutrals.pro`, has been written to perform the solution algorithm:

```
;+
; Kinetic_Neutrals.pro
;
;   Solves a 1-D spatial, 2-D velocity kinetic neutral transport
; problem for atomic hydrogen or deuterium by computing successive generations of
; charge exchange neutrals. Rates for charge exchange and
; electron impact ionization of atomic hydrogen are used.
;
;   The positive vx half of the neutral distribution function is
; inputted at x(0). Profiles of Ti(x), Te(x), and n(x) are inputed. The code
; returns the neutral distribution function, fn(vr,vx,x) for all vx, vr, and x
; of the specified vr,vx,x grid.
;
;   Since the problem involves only the x spatial dimension, the distribution
; function has rotational symmetry about the vx axis. Consequently, the
; distribution function only depends on x, vx and vr where vr =sqrt(vy^2+vz^2)
;
;  History:
;
;    B. LaBombard   First coding based on 1-D, 2-V model 11/3/2000
;
```

```
;     For more information, see write-up: "A 1-D Space, 2-D Velocity,
;     Kinetic Neutral Transport Algorithm", B. LaBombard
;
;________________________________________________________________________
;
pro Kinetic_Neutrals,vx,vr,x,Tnorm,mu,Ti,Te,n,fnBC,n0BC,$
     fn,n0,gammax0,vx0,p0,pi0_xx,pi0_yy,pi0_zz,$
     T0,qx0,qx0_total,Sion,Qin,Rxin,Qin_total,Albedo,truncate=truncate,$
     Simple_CX=Simple_CX,max_gen=max_gen,Max_dx=Max_dx,$
     error=error,compute_errors=compute_errors,$
     vbar_error=vbar_error,mesh_error=mesh_error,max_mesh_error=max_mesh_error,$
     ave_mesh_error=ave_mesh_error,moment_error=moment_error,$
     max_moment_error=max_moment_error,qx0_total_error=qx0_total_error,$
     Qin_total_error=Qin_total_error,Sion_Error=Sion_Error,$
     plot=plot,debug=debug,debrief=debrief
;
;  Input:
;             vx(*)   - fltarr(nvx), normalized x velocity coordinate
;                       [negative values, 0 , positive values],
;                       monotonically increasing. Note: a nonuniform mesh can
;                       be used. Dimensional velocity
;                       is v = Vth * vx where Vth=sqrt(2 k Tnorm/(mH*mu))
;                       Note: nvx must be odd and vx(*) symmetric about
;                       0 and contain a zero element
;             vr(*)   - fltarr(nvr), normalized radial velocity coordinate
;                       [positive values], monotonically increasing. Note: a
;                       non-uniform mesh can be used.
;                       Dimensional velocity is v = Vth * vr where
;                       Vth=sqrt(2 k Tnorm/(mH*mu))
;                       Note: vr must not contain a zero element
;             x(*)    - fltarr(nx), spatial coordinate (meters),
;                       positive, monontonically increasing. Note: a
;                       non-uniform mesh can be used.
;             Tnorm   - Float, temperature corresponding to the thermal speed
;                       (see vx and vr above) (eV)
;             mu      - Float, 1=hydrogen, 2=deuterium
;             Ti      - fltarr(nx), Ion temperature profile (eV)
;             Te      - fltarr(nx), electron temperature profile (eV)
;             n       - fltarr(nx), plasma density profile (m^-3)
;             fnBC    - fltarr(nvr,nvx), this is an input boundary condition
;                       specifying the shape of the neutral velocity
;                       distribution function at location x(0). Normalization ;
;                       is arbitrary. Only values with positive vx,
;                       fnBC(*,(nvx-1)/2:*) are used by the code.
;             n0BC    - float, desired neutral density at location x(0) (m^-3)
;                       On output, fn is scaled to correspond to this density
;                       (see below).
;  Output:
;             fn      - fltarr(nvr,nvx,nx), neutral velocity distribution
;                       function. fn is normalized so that the
;                          neutral density, n0(k), is defined as the velocity
;                          space integration:
;                              n0(k)=vth3*total(Vr2pidVr*(fn(*,*,k)#dVx))
;                          fn is scaled so that n0(0) is equal to n0BC, the
;                          desired neutral density at x(0).
;               n0      - fltarr(nx), neutral density profile (m^-3)
;               gammax0 - fltarr(nx), neutral flux profile (# m^-2 s^-1)
;               vx0     - fltarr(nx), neutral velocity profile (m s^-1)
;               p0      - fltarr(nx), neutral pressure (eV m^-2)
;               pi0_xx  - fltarr(nx), xx element of stress tensor (eV m^-2)
;               pi0_yy  - fltarr(nx), yy element of stress tensor (eV m^-2)
;               pi0_zz  - fltarr(nx), zz element of stress tensor (eV m^-2)
;                              = pi0_yy
;                          Note: cylindrical system relates r^2 = y^2 + z^2
;                          All other stress tensor elements are zero.
;               T0      - fltarr(nx), neutral temperature profile (eV)
;               qx0     - fltarr(nx), neutral random heat flux profile (W m^-2)
;             qx0_total - fltarr(nx), total neutral heat flux profile (W m^-2)
;                         This is the total heat flux transported by the neutrals:
;         qx0_total=(0.5*n0*(mu*mH)*vx0*vx0 + 2.5*p0*q)*vx0 + pi0_xx*vx0 + qx0
;               Sion    - fltarr(nx), ionization rate (# m^-3)
;               Qin     - fltarr(nx), rate of net thermal energy transfer from
;                          the ion to neutral species(watts m^-3)
;               Rxin    - fltarr(nx), rate of x momentum transfer from the ion
;                          to neutral species (=force, N m^-2).
;               Qin_total- fltarr(nx), net rate of total energy transfer from
;                           the ion to neutral species
;                         = Qin + Rxin*vx0 - 0.5*(mu*mH)*Sion*vx0*vx0 (W m^-3)
;               Albedo   - float, fraction of incident flux of neutrals (at
;                            x=x(0)) that is reflected back towards x < x(0)
; KEYWORDS:
;   Input:
;               truncate- float, stop computation when the maximum
;                          increment of neutral density normalized to
;                          inputed neutral density is less than this
;                          value in a subsequent charge exchange
;                          generation. Default value is 1.0e-4
;
;             Simple_CX - if set, then use CX source option (B): Neutrals are
;                          born in velocity with a distribution proportional to
;                          the local ion distribution function. Simple_CX=1 is
;                          default.
;
;                          if not set, then use CX source option (A): The CX
;                          source neutral distribution function is computed by
;                          evaluating the the CX cross section for each
;                          combination of (vr,vx,vr',vx') and convolving it with
;                          the neutral distribution function.
;                          This option requires much more CPU time and memory.
;
```

```
;               Max_gen - integer, maximum number of charge exchange
;                         generations to try including before giving up.
;                         Default is 20.
;
;        Compute_Errors - if set, then return error estimates below
;
; KEYWORDS:
;   Output:
;           Vbar_error  - float(nx), returns numerical error in computing
;                         the speed of ions averged over Maxwellian
;                         distribution.
;                         The average speed should be:
;                                vbar_exact=2*Vth*sqrt(Ti(*)/Tnorm)/sqrt(!pi)
;                         Vbar_error returns: abs(vbar-vbar_exact)/vbar_exact
;                         where vbar is the numerically computed value.
;
;               Max_dx  - float(nx), Max_dx(k) for k=0:nx-2 returns maximum
;                         allowed x(k+1)-x(k) that avoids unphysical negative
;                         contributions to fn
;
;                error -  Returns error status: 0=no error, solution returned
;                                               1=error, no solution returned
;
;       if COMPUTE_ERRORS keyword is set then the following is returned:
;
;            mesh_error - fltarr(nvr,nvx,nx), normalized error of solution
;                         based on substitution into Boltzmann equation.
;        max_mesh_error - float, max(mesh_error)
;        min_mesh_error - float, min(mesh_error)
;          moment_error - fltarr(nx,m), normalized error of solution
;                         based on substitution into velocity space
;                         moments (v^m) of Boltzmann equation, m=[0,1,2,3,4]
;      max_moment_error - fltarr(5), max(moment_error(*,m))
;       qx0_total_error - fltarr(nx), normalized error estimate in computation
;                         of qx0_total
;       Qin_total_error - fltarr(nx), normalized error estimate in computation
;                         of Qin_total
;            Sion_error - fltarr(nx), normalized error in charge exchange
;                         collision operator
;                         This is a measure of how well the charge exchange
;                         collision operator conserves particles.
;______________________________________________________________________
;-
```

5. Validation of Numerics

The total neutral distribution function, $f_n(v_x, v_r, x)$, computed from the numerical algorithm should satisfy the Boltzmann equation,

$$v_x \frac{\partial f_n}{\partial x} = \beta_{cx} - \alpha_c f_n, \tag{22}$$

at every location on the mesh. According to the numerical scheme outlined in section 3, this equation can be written as the mesh equation,

$$2v_x \frac{f_{n,m+1} - f_{n,m}}{x_{m+1} - x_m} = \beta_{cx,m+1} + \beta_{cx,m} - (\alpha_{c,m+1} f_{n,m+1} + \alpha_{c,m} f_{n,m}) \quad . \tag{23}$$

5.1 Mesh Point Error

By defining the three terms in Eq. (23) as

$$\begin{aligned} T1_m &= 2v_x \frac{f_{n,m+1} - f_{n,m}}{x_{m+1} - x_m}, \\ T2_m &= \beta_{cx,m+1} + \beta_{cx,m}, \\ T3_m &= \alpha_{c,m+1} f_{n,m+1} + \alpha_{c,m} f_{n,m} \end{aligned} \tag{24}$$

a normalized error parameter can be defined for spatial mesh points with $x_a \le x < x_b$, and all velocity mesh points as

$$\varepsilon_{k,l,m} \equiv \left|T1_{k,l,m} - T2_{k,l,m} + T3_{k,l,m}\right| / \max(\left|T1_{k,l,m}\right|, \left|T2_{k,l,m}\right|, \left|T3_{k,l,m}\right|). \tag{25}$$

Values of $\varepsilon_{k,l,m}$ are returned in parameter `mesh_error`. The average value of $\varepsilon_{k,l}$ and the maximum value of $\varepsilon_{k,l,m}$ over the mesh is returned in parameters `ave_mesh_error` and `max_mesh_error`.

5.2 Velocity Moment Error

$f_n(v_x, v_r, x)$ should also satisfy velocity moments (*M*) of the Boltzmann equation,

$$\iint \frac{\partial f_n}{\partial x} v_x^{M+1} \partial v_x v_r \partial v_r = \iint \alpha_{cx} \hat{f}_i n_0 v_x^M \partial v_x v_r \partial v_r - \iint \alpha_c f_n v_x^M \partial v_x v_r \partial v_r . \tag{26}$$

Making use of Eqs. (23) and (24), this can be expressed as

$$\iint T1_m v_x^M \partial v_x v_r \partial v_r = \iint T2_m v_x^M \partial v_x v_r \partial v_r - \iint T3_m v_x^M \partial v_x v_r \partial v_r .$$

Defining

$$T1_{M,m} = \iint T1_m v_x^M \partial v_x v_r \partial v_r ,$$

$$T2_{M,m} = \iint T2_m v_x^M \partial v_x v_r \partial v_r ,$$

$$T3_{M,m} = \iint T3_m v_x^M \partial v_x v_r \partial v_r \,, \tag{27}$$

a normalized 'velocity moment error' for spatial mesh points with $x_a \le x < x_b$, can be constructed as

$$\eta_{M,m} \equiv \left|T1_{M,m} - T2_{M,m} + T3_{M,m}\right| / \max(\left|T1_{M,m}\right|, \left|T2_{M,m}\right|, \left|T3_{M,m}\right|). \tag{28}$$

Values of $\eta_{M,m}$ are returned in parameter `moment_error`. The maximum errors for moments $0 \le M \le 4$ are returned in parameter `max_moment_error`.

5.3 Error Associated with Digital Representation of Distribution Functions

Discretization introduces errors in evaluating moments of the distribution functions. If the velocity space mesh is too coarse such that f varies strongly between mesh points or if the mesh does not cover a sufficient range such that f is has a significant component *outside* the mesh, then numerically evaluated moments of f will not be accurate. As a test of the numerical accuracy in the digital representation of f, `Kinetic_Neutrals.pro` computes the average speed of the Maxwellian plasma ions ($\hat{f}_i$) at each x_m location and compares it to the theoretical value:

$$\bar{v}_{code} = 2\pi \iint \sqrt{v_r^2 + v_x^2}\, \hat{f}_i \partial v_x v_r \partial v_r$$

$$\bar{v}_{exact} \equiv \int |\underline{v}|\, \hat{f}_i \;\; \partial^3 v \; = 4\pi \int \hat{f}_i v^3 \partial v \;\; = \frac{2 v_{th}}{\sqrt{\pi}}$$

The parameter `Vbar_error` in `Kinetic_Neutrals.pro` returns a normalized error defined as

$$\bar{v}_{error,m} \equiv \left|\bar{v}_{code,m} - \bar{v}_{exact,m}\right| / \bar{v}_{exact,m} \,.$$

Values of `Vbar_error` = 0.01 or less indicate that $\hat{f}_i$ (and by inference, f_n) are reasonably well represented by their digital expressions and that the choice of velocity mesh size and spacing is appropriate. For the case when $T_i(x)$ varies significantly across the mesh, a non-uniform velocity space mesh can be used to more evenly distribute `Vbar_error` over the mesh.

[Note: See author for IDL procedure `Create_VrVxMesh.pro`, which can be used to generate a (v_x, v_r) mesh which is close to the optimum for a specified $T_i(x)$ profile.]

References

[1] Research Systems Inc., Interactive Data Language (IDL®).

[2] Janev, R.K., Elementary processes in hydrogen-helium plasmas : cross sections and reaction rate coefficients (Springer-Verlag, Berlin ; New York, 1987).