UCRL-52233

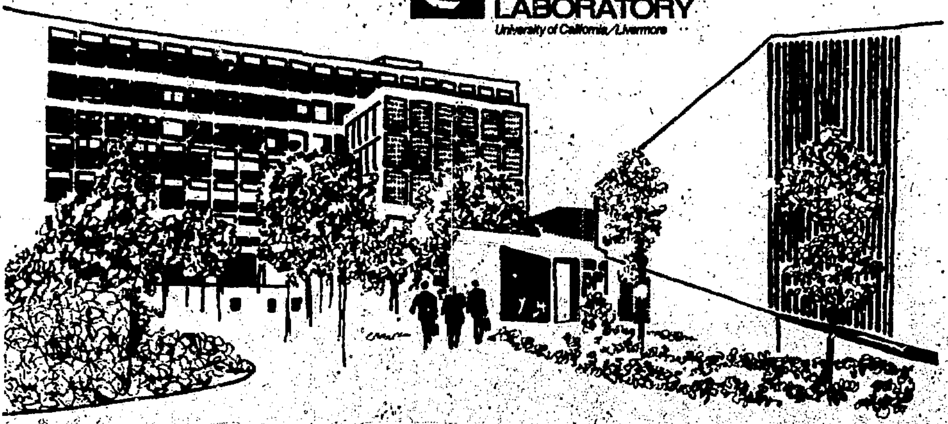
COMPUTATION OF THE BOUNCE-AVERAGE CODE

T. A. Cutler, L. D. Pearlstein and M. E. Rensink

May 23, 1977

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48

LAWRENCE LIVERMORE LABORATORY
University of California/Livermore



Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $3.00

Page Range	Domestic Price	Page Range	Domestic Price
001–025	$ 3.50	326–350	10.00
026–050	4.00	351–375	10.50
051–075	4.50	376–400	10.75
076–100	5.00	401–425	11.00
101–125	5.50	426–450	11.75
126–150	6.00	451–475	12.00
151–175	6.75	476–500	12.50
176–200	7.50	501–525	12.75
201–225	7.75	526–550	13.00
226–250	8.00	551–575	13.50
251–275	9.00	576–600	13.75
276–300	9.25	601–up	*
301–325	9.75		

*Add $2.50 for each additional 100 page increment from 601 to 1,000 pages; add $4.50 for each additional 100 page increment over 1,000 pages.

Distribution Category
UC-21

LAWRENCE LIVERMORE LABORATORY
University of California/Livermore, California/94550

UCRL-52233

COMPUTATION OF THE BOUNCE-AVERAGE CODE

T. A. Cutler, L. D. Pearlstein and M. E. Rensink

MS. date: May 23, 1977

CONTENTS

Abstract 1

Geometry 1

Bounce-Averaged Transport Equation 2

Evaluation of the Distribution Functions Off the Midplane 4

Evaluation of the Fokker-Planck Collision Operator 5

Inverting ∇^2 and ∇^4 7

Transforming to Midplane Coordinates 9

Source Terms 12

Potential Profile 13

Pressure Profiles 14

Updating the Magnetic Field Profile 15

$\dot{B}$ Terms 16

Electron Advancement 17

Numerical Details 17

Grid 17

Numerical Advancement of Distribution Functions 18

Boundary 19

Orbit Integral 20

Line Integral Diagnostics 21

References 23

Appendix: Listing of the Bounce-Average Code 24

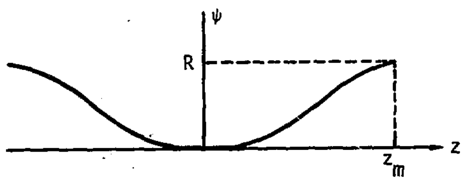
COMPUTATION OF THE BOUNCE-AVERAGE CODE

Abstract

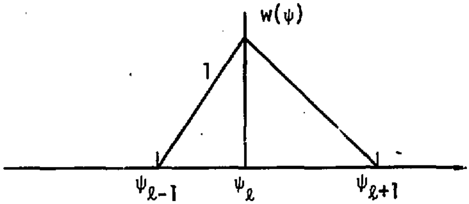
The bounce-average computer code simulates the two-dimensional velocity transport of ions in a mirror machine. The code evaluates and bounce-averages the collision operator and sources along the field line. A self-consistent equilibrium magnetic field is also computed using the long-thin approximation. Optionally included are terms that maintain μ, J invariance as the magnetic field changes in time. In this report, we describe the assumptions and analysis that form the foundation of the bounce-average code. When references can be cited, the required results are merely stated and explained briefly. A listing of the code is appended.

Geometry

The spatial and velocity geometry can be illustrated as



where z is along a field line. The magnetic well has a monotonic but otherwise general profile:



Here, $\Psi(z)$ equals $B(z)/B(0)$ and is symmetric about the midplane, z_m equals the half-length of the mirror machine, and $B(z)$ is the magnetic field strength.

The ion distribution is described by

$$f = f(v,\theta,z) \ .$$

There is no radial dependence. We assume that radial effects are either slow as compared to collisional processes or are averaged out. Also, the gyromotion is integrated out.[1]

The electrons are given by

$$f_e = f_e(v,z) \ .$$

Bounce-Averaged Transport Equation

Integrating the Boltzmann equation along orbits, we have[1]

$$\frac{\partial f}{\partial t}(v_0,\theta_0) = \frac{1}{\tau}\oint_{(v_0,\theta_0)} dt \left[\left(\frac{\partial f}{\partial t}\right)_{coll} + S\right] , \tag{1}$$

where $f(v_0,\theta_0)$ is the midplane ion distribution [i.e., $f(v,\theta,z)$ at $z = 0$] and $(\partial f/\partial t)_{coll}$ and S are the local (i.e., dependent on v, θ, and z) collision operator and source, respectively.

The orbit integral operator applied to $S(v,\theta,z)$ gives an orbit averaged quantity,

$$\bar{S}(v_0,\theta_0) = \frac{1}{\tau}\oint_{(v_0,\theta_0)} dt\ S(v,\theta,z) \ ,$$

$$= \frac{1}{\tau}\int_0^{z_b} \frac{dz}{v(z)\cos\ \theta(z)}\ S\Big[v(z),\ \theta(z),\ z\Big] \ ,$$

where τ is the bounce time, z_b is the turning point of the orbit where $\theta(z_b) = \pi/2$, and $v(z)$ and $\theta(z)$ are given by the orbit equations for energy conservation and magnetic moment invariance, respectively;

$$v^2 = v_0^2 + v_p^2 \ ,$$

$$v^2 \sin^2\ \theta = \psi(z) v_0^2 \sin^2\ \theta_0 \ .$$

The v_p^2 in the energy conservation equation is the velocity arising from the ambipolar potential $\phi(z)$,

$$v_p^2 = \frac{z_i\ e\ \phi(z)}{(1/2)\ m_i} \ .$$

If a square well is assumed for $\psi(z)$, then

$$\bar{S}(v_0,\ \theta_0) = S(v_0,\ \theta_0,\ 0) \ ,$$

and Eq. (1) is reduced to the usual starting point for Fokker-Planck work. Equation (1) becomes a parabolically partial differential equation (with slowly varying coefficients) and can be numerically integrated efficiently with implicit methods.

However, instead of the square-well assumption, the bounce-average code employs a numerical bounce average. The collision operator is still reduceable to a nonlinear parabolic differential operator on the midplane distribution functions. This reduction is accomplished in the following steps:

- The coefficients of the local collision operator,

$$\left(\frac{\partial f}{\partial t}\right)_{coll} = A^{vv}\frac{\partial^2 f}{\partial v^2} + A^{v\theta}\frac{\partial^2 f}{\partial v \partial\theta} + A^{\theta\theta}\frac{\partial^2 f}{\partial\theta^2} + A^{v}\frac{\partial f}{\partial v} + A^{\theta}\frac{\partial f}{\partial\theta} + A \ ,$$

  are evaluated.

- With $f(v,\theta,z) = f(v_0, \theta_0)$, where f is constant along an orbit, and using the chain rule, local partial derivatives are expressed in terms of derivatives of $f(v_0, \theta_0)$. Thus, the collision operator can be written as

$$\left(\frac{\partial f}{\partial t}\right)_{coll} = B^{v_0 v_0} \frac{\partial^2 f}{\partial {v_0}^2} + B^{v_0 \theta_0} \frac{\partial^2 f}{\partial v_0 \partial \theta_0}$$

$$+ B^{\theta_0 \theta_0} \frac{\partial^2 f}{\partial {\theta_0}^2} + B^{v_0} \frac{\partial f}{\partial v_0} + B^{\theta_0} \frac{\partial f}{\partial \theta_0} + Bf ,$$

where, for example,

$$B^{v_0 v_0} = \left(\frac{\partial v_0}{\partial v}\right)^2 A^{vv} .$$

- The coefficients $B^{v_0 v_0}(v,\theta,z)$ are then bounce-averaged and we obtain

$$\left(\frac{\partial f}{\partial t}(v_0,\theta_0)\right)_{coll} = C^{v_0 v_0} \frac{\partial^2 f}{\partial {v_0}^2} + C^{v_0 \theta_0} \frac{\partial^2 f}{\partial v_0 \partial \theta_0}$$

$$+ C^{\theta_0 \theta_0} \frac{\partial^2 f}{\partial {\theta_0}^2} + C^{v_0} \frac{\partial f}{\partial v_0} + C^{\theta_0} \frac{\partial f}{\partial \theta_0} + Cf .$$

Evaluation of the Distribution Functions Off the Midplane

To evaluate various quantities that are local in z, the distribution functions off the midplane must be known. For this purpose we use

$$f(v,\theta,z) = f(v_0,\theta_0) , \qquad f_e(v,z) = f_e(v_0) , \tag{3}$$

where (v,θ,z) and (v_0,θ_0) are connected by orbit equations. [See Ref. 1 for a careful argument for Eq. (3)]. More qualitatively, Eq. (3) follows from

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \bar{v} \cdot \frac{\partial f}{\partial \bar{x}} + \bar{a} \cdot \frac{\partial f}{\partial \bar{v}} = 0 \ , \qquad (4)$$

the Vlasov equation, which says that f is constant along an orbit. This equation is valid so long as the collision and source times are long compared to the bounce time.

We obtain f off the midplane by integrating Eq. (4),

$$f(v,\theta,z,t) = f(v_0,\theta_0,t_0) \ .$$

If we assume $f(v_0,\theta_0,t_0)$ is constant during a few bounce times or if we regard $f(v_0,\theta_0)$ as an average over a few bounce times, Eq. (3) follows.

The most straightforward use of Eq. (3) in the bounce-average code is in the reconstruction of the distribution function off the midplane so that the integrals

$$a_m(v) = (2m + 1)\int_0^1 d\mu \ f(v,\theta,z)P_m(\mu) \ , \qquad \mu = \cos\theta \ ,$$

the Legendre polynomial projections of f, can be computed. A local μ mesh is generated from $\mu = 0$ to μ_{loss} and $f(v,\mu,z) = f(v_0,\mu_0)$ from orbit equations and linear interpolation. However, the more usual application of Eq. (3) is to transform integrals from local coordinates to midplane coordinates.

Evaluation of the Fokker-Planck Collision Operator

We take

$$\left(\frac{\partial f_a}{\partial t}\right)_{coll} = - \nabla \cdot (f_a H_a) + \frac{1}{2} \nabla\nabla : (f_a \nabla\nabla G_a) \ ,$$

where

$$H_a(\bar{v}) = \sum_b \ln\Lambda_{ab} \ c_{ab}^1 h_b(\bar{v}) \ ,$$

$$G_a(\bar{v}) = \sum_b \ln\Lambda_{ab} \ c_{ab}^2 g_b(\bar{v}) \ ,$$

$$c_{ab}^{1} = \Gamma_a\left(\frac{Z_b}{Z_a}\right)^2\left(1 + \frac{m_a}{m_b}\right) ,$$

$$c_{ab}^{2} = \Gamma_a\left(\frac{Z_b}{Z_a}\right)^2 ,$$

$$\Gamma_a = 4\pi Z_a^4 \, e^4/m_a^2 ,$$

and h_b, g_b are the Rosenbluth potentials,

$$\nabla^2 h_b = -4\pi f_b ,$$

$$\nabla^4 g_b = -8\pi f_b ,$$

and

$$\ln\Lambda_{ab} = \ln\left[\frac{m_a m_b}{m_a + m_b} \frac{2\alpha\lambda_D}{r_e m_e c} \sup_{k=a,b}\sqrt{\frac{2E_k}{m_k}}\right] - \frac{1}{2} ,$$

the Coulomb logarithm, where r_e = 2.8179(-13) cm, the classical electron radius; $\lambda_D = \sqrt{E_e/6\pi n_e e^2}$, the Debye length; α = 1/137, the fine structure constant; c = 3 10^{10} cm/s; and e = 4.803 10^{-10}. Also, E_k and m_k are energy (erg) and mass (g) of species k while n_e = electron density (particle/cm^3).[2,3]

When the form of $(\partial f_a/\partial t)_{coll}$ is specialized to (v,θ) space and to one ion species with electrons, we have (from Refs. 4,5)

$$\left(\frac{\partial f_i}{\partial f}\right)_{coll} = A^{vv} \frac{\partial^2 f}{\partial v^2} + A^{v\theta} \frac{\partial^2 f}{\partial v \partial \theta}$$

$$+ A^{\theta\theta} \frac{\partial^2 f}{\partial \theta^2} + A^{v} \frac{\partial f}{\partial v} + A^{\theta} \frac{\partial f}{\partial \theta} + Af ,$$

where

$$A^{vv} = \frac{\Gamma i}{2} \frac{\partial^2 G_i}{\partial_v 2} ,$$

$$A^{v\theta} = \frac{\Gamma i}{2}\left(\frac{1}{v^2}\frac{\partial^2 G_i}{\partial v \partial \theta} - \frac{1}{v^3}\frac{\partial G_i}{\partial \theta}\right) ,$$

$$A^{\theta\theta} = \frac{\Gamma i}{2}\left(\frac{1}{v^4}\frac{\partial^2 G_i}{\partial \theta^2} + \frac{1}{v^3}\frac{\partial G_i}{\partial v}\right) ,$$

$$A^{v} = \Gamma_i \left(\frac{1}{2v^3}\frac{\partial^2 G_i}{\partial \theta^2} + \frac{\mathrm{ctn}\theta}{2v^3}\frac{\partial G_i}{\partial \theta} + \frac{1}{v^2}\frac{\partial G_i}{\partial v} + \frac{\partial C}{\partial v}\right) ,$$

$$A^{\theta} = \Gamma_i \left[\frac{1}{v^4 \sin^2\theta}\left(1 - \frac{\cos^2\theta}{2}\right)\frac{\partial G_i}{\partial \theta} - \frac{1}{v^3}\frac{\partial^2 G_i}{\partial v \partial \theta} + \frac{\mathrm{ctn}\theta}{2v^3}\frac{\partial G_i}{\partial v}\right] ,$$

$$A = 4\pi\Gamma_i \left(\frac{m_i}{m_e}\frac{f_e}{Z_i^2} + f_i\right) ,$$

and

$$G_i = \ln\Lambda_{ii}\, g_i + \frac{\ln\Lambda_{ie}}{Z_i^2}\, g_e ,$$

$$\nabla^4 g_a = -8\pi f_a,\ a = i,e ,$$

$$C = \left(1 - \frac{m_i}{m_e}\right)\frac{\ln\Lambda_{ie}}{Z_i^2}\, h_e ,$$

$$\nabla^2 h_e = -4\pi f_e .$$

INVERTING ∇^2 AND ∇^4

The evaluation of the Fokker-Planck operator is accomplished with the following procedure for solving $\nabla^4 g = -8\pi f$ and $\nabla^2 h = -4\pi f$, given $f = f(v,\theta)$.

Let

$$f = \sum_{m=0}^{\infty} a_m(v)P_m(\mu), \quad \mu = \cos\theta ,$$

where

$$a_m(v) = \frac{1}{\int_{-1}^{1} P_m^{\,2} d\mu} \int_{-1}^{1} fP_m \, d\mu .$$

We define the following functionals with dimensions of f as

$$M_m(v) = v^{m-2}\int_v^{\infty} a_m(s)\, S^{1-m} ds ,$$

$$N_m(v) = v^{-3-m}\int_0^{v} a_m(s)\, S^{2+m} ds ,$$

$$R_m(v) = v^{-4+m}\int_v^{\infty} a_m(s)\, S^{3-m} ds ,$$

$$E_m(v) = v^{-5-m}\int_0^{v} a_m(s)\, S^{4+m} ds .$$

Then, the expansions of g and h,

$$g = \sum g_m(v)\, P_m(\mu) ,$$

$$h = \sum h_m(v)\, P_m(\mu) ,$$

can be solved[1-3] for with

$$g_m(v) = \frac{4\pi}{2m+1} v^4 \left[\frac{1}{2m+3} (E_m + M_m) - \frac{1}{2m-1} (N_m + R_m) \right],$$

$$\frac{\partial g_m}{\partial v} = \frac{4\pi}{2m+1} v^3 \left\{ \frac{1}{2m+3} \left[(m+2)\, M_m - (m+1)E_m \right] - \frac{1}{2m-1} \left(mR_m - (m-1)N_m \right) \right\},$$

$$\frac{\partial^2 g_m}{\partial v^2} = \frac{4\pi}{2m+1} v^2 \left[\frac{(m+1)(m+2)}{2m+3} (E_m + M_m) - \frac{m(m-1)}{2m-1} (N_m + R_m) \right] ,$$

$$h_m = \frac{4\pi}{2m+1} v^2 (N_m + M_m) ,$$

$$\frac{\partial h_m}{\partial v} = \frac{4\pi}{2m+1} v \left(mM_m - (m+1)N_m\right) .$$

Transforming to Midplane Coordinates

Given

$$\left(\frac{\partial f}{\partial t}\right)_{coll} = A^{vv} \frac{\partial^2 f}{\partial v^2} + A^{v\theta} \frac{\partial^2 f}{\partial v \partial \theta} + A^{\theta\theta} \frac{\partial^2 f}{\partial \partial^2} + A^{v} \frac{\partial f}{\partial v} + A^{\theta} \frac{\partial f}{\partial \theta} + Af , \tag{5}$$

Eq. (5) can be rewritten as

$$\left(\frac{\partial f}{\partial t}\right)_{coll} = B^{v_0 v_0} \frac{\partial^2 f}{\partial_{v_0}^2} + B^{v_0 \theta_0} \frac{\partial^2 f}{\partial v_0 \partial \theta_0} + B^{\theta_0 \theta_0} \frac{\partial^2 f}{\partial \theta_0^2} + B^{v_0} \frac{\partial f}{\partial v_0} + B^{\theta_0} \frac{\partial f}{\partial \theta_0} + Bf . \tag{6}$$

However, this requires transforming the local partial derivatives in (v,θ) to midplane derivatives in (v_0,θ_0). Using the chain rule, we have

$$\frac{\partial}{\partial v} = \frac{\partial v_0}{\partial v} \frac{\partial}{\partial v_0} + \frac{\partial \theta_0}{\partial v} \frac{\partial}{\partial \theta_0} ,$$

and

$$\frac{\partial}{\partial \theta} = \frac{\partial \theta_0}{\partial \theta} \frac{\partial}{\partial \theta_0} ,$$

using

$$\frac{\partial v_0}{\partial \theta} = 0 .$$

Taking derivatives of the above first-order equations, we obtain the second-order derivatives:

$$\frac{\partial^2}{\partial v^2} = \frac{\partial^2 v_0}{\partial v^2}\frac{\partial}{\partial v_0} + \frac{\partial^2 \theta_0}{\partial v^2}\frac{\partial}{\partial \theta_0} + \left(\frac{\partial v_0}{\partial v}\right)^2$$

$$\times \frac{\partial^2}{\partial v_0^2} + 2\,\frac{\partial v_0}{\partial v}\frac{\partial \theta_0}{\partial v}\frac{\partial^2}{\partial v_0 \partial \theta_0} + \left(\frac{\partial \theta_0}{\partial v}\right)^2 \frac{\partial^2}{\partial \theta_0^2} ,$$

$$\frac{\partial^2}{\partial v \partial \theta} = \frac{\partial^2 \theta_0}{\partial v \partial \theta}\frac{\partial}{\partial \theta_0} + \frac{\partial \theta_0}{\partial \theta}\frac{\partial v_0}{\partial v}\frac{\partial^2}{\partial v_0 \partial \theta_0} + \frac{\partial \theta_0}{\partial \theta}\frac{\partial \theta_0}{\partial v}\frac{\partial^2}{\partial \theta_0^2} ,$$

$$\frac{\partial^2}{\partial \theta^2} = \frac{\partial^2 \theta_0}{\partial \theta^2}\frac{\partial}{\partial \theta_0} + \left(\frac{\partial \theta_0}{\partial \theta}\right)^2 \frac{\partial^2}{\partial \theta_0^2} .$$

With the above differential operator equations, we can write the coefficients of Eq. (6) in terms of the coefficients of Eq. (5) as follows:

$$B^{v_0 v_0} = \left(\frac{\partial v_0}{\partial v}\right)^2 A^{vv} ,$$

$$B^{v_0 \theta_0} = \frac{2\partial v_0}{\partial v}\frac{\partial \theta_0}{\partial v} A^{vv} + \frac{\partial \theta_0}{\partial \theta}\frac{\partial v_0}{\partial v} A^{v\theta} ,$$

and

$$B^{\theta_0 \theta_0} = \left(\frac{\partial \theta_0}{\partial v}\right)^2 A^{vv} + \frac{\partial \theta_0}{\partial \theta}\frac{\partial \theta_0}{\partial v} A^{v\theta} + \left(\frac{\partial \theta_0}{\partial \theta}\right)^2 A^{\theta\theta} .$$

In addition,

$$B^{v_0} = \frac{\partial v_0}{\partial v} A^{v} + \frac{\partial^2 v_0}{\partial v^2} A^{vv} ,$$

$$B^{o} = \frac{\partial \theta_0}{\partial v} A^{v} + \frac{\partial \theta_0}{\partial \theta} A + \frac{\partial^2 \theta_0}{\partial v^2} A^{vv} + \frac{\partial^2 \theta_0}{\partial v \partial \theta} A^{v\theta} + \frac{\partial^2 \theta_0}{\partial \theta^2} A^{\theta\theta} ,$$

and

$$B = A .$$

From the orbit equations, we have

$$\frac{\partial v_0}{\partial v} = \frac{v}{v_0} ,$$

$$\frac{\partial v_0}{\partial \theta} = 0 ,$$

$$\frac{\partial^2 v_0}{\partial v^2} = - v_p^{\ 2}/v_0^{\ 3} ,$$

and

$$\frac{\partial \theta_0}{\partial v} = - \tan \theta_0 \frac{v_p^{\ 2}}{v v_0^{\ 2}} .$$

Also,

$$\frac{\partial^2 \theta_0}{\partial v^2} = \frac{1}{v} \frac{\partial \theta_0}{\partial v} \left[\left(\frac{v_p}{v_0} \right)^2 \left(\tan^2 \theta_0 - 1 \right) - 3 \right] ,$$

$$\frac{\partial \theta_0}{\partial \theta} = \frac{\tan \theta_0}{\tan \theta} ,$$

$$\frac{\partial^2 \theta_0}{\partial \theta^2} = \tan \theta_0 \left[\left(\frac{\partial \theta_0}{\partial \theta} \right)^2 - 1 \right] ,$$

and

$$\frac{\partial^2 \theta_0}{\partial v \partial \theta} = - \frac{\partial \theta_0}{\partial \theta} \frac{\sec^2 \theta_0}{v} \left(\frac{v_p}{v_0} \right)^2 .$$

Bounce-averaging Eq. (6) yields

$$\left[\frac{\partial f}{\partial t}(v_0, \theta_0)\right]_{coll} = C^{v_0 v_0}\frac{\partial^2 f}{\partial {v_0}^2} + C^{v_0 \theta_0}\frac{\partial^2 f}{\partial v_0 \partial \theta_0} + C^{\theta_0 \theta_0}\frac{\partial^2 f}{\partial \theta_0^2} + C^{v_0}\frac{\partial f}{\partial v_0} + C^{\theta_0}\frac{\partial f}{\partial \theta_0} + Cf \ , \tag{7}$$

where, for example,

$$C^{v_0 v_0} = \frac{1}{\tau}\int^{z_b} \frac{dz}{v \cos \theta} B^{v_0 v_0}(v,\theta,z) \ ,$$

and similarly for each coefficient. With the collision operator in this form, the ion equation can be advanced fully implicitly.

Source Terms

The local source appearing in Eq. (1) is given the form

$$S(v,\theta,z) = \begin{cases} J + (\sigma_i + \sigma_{cx})n(z)\ S(v,\theta) - \sigma_{cx}\ f(v,\theta,z) \ , \\ 0 \text{ if } z > z_{beam} \ , \end{cases}$$

where $n(z)$ = local density, J, σ_i, σ_{cx}, and z_{beam} are input, and $S(v,\theta)$ is a normalized double Gaussian in v and $\cos\theta$. The shape of $S(v,\theta)$ is specified by the input parameters:

$$4\pi\int_0^{\infty} dv\ v^2 \int_0^1 d\mu\ S(v,\theta) = 1 \ .$$

More than one source can be specified, in which case, the overall source is a sum of such forms.

If $J = 0$, then

$$4\pi\int_0^{\infty} dv\ v^2 \int_0^1 d\mu\ S(v,\theta,z) = \sigma_i\ n(z) \ ,$$

is the ionization rate. The bounce-averaged source term appears as

$$\frac{1}{\tau}\int dt\ S = \bar{S}(v_0,\ \theta_0) - C_x(v_0,\ \theta_0)\ f\ (v_0,\ \theta_0)\ ,$$

using $f(v,\theta,z) = f(v_0,\ \theta_0)$.

Potential Profile

The potential is determined as a function of ψ by requiring charge neutrality. The ion density is obtained with

$$\begin{aligned} n_i(\psi,\phi) &= 4\pi\int_0^\infty dv\ v^2\int_0^1 d\mu\ f(v,\theta,\psi)\ , \\ &= 4\pi\ \psi^{1/2}\int_0^\infty dv_0\ v_0^2\int_{\mu_r}^1 d\mu_0\ \frac{\mu_0}{\sqrt{\mu_0^2 - 1+\lambda}}\ f(v_0,\ \theta_0)\ , \end{aligned} \tag{8}$$

through the change of variables $v_0^2 = v^2 - v_p^2$ and $1 - \mu_0^2 = \lambda(1 - \mu^2)$, where

$$v_p^2 = \frac{e\phi}{\frac{1}{2}m_i}\ ,$$

$$\lambda = \frac{v^2}{\psi v_0^2}\ ,$$

and $\mu_r^2 = \max\ (0,\ 1-\lambda)$. Here, μ_r is the cutoff resulting from orbits turning before the current value of ψ.

Normally the integrand is singular at μ_r. We use a generalized trapezoidal rule, equivalent to integrating analytically, and assume that f is linear in μ_0 between grid points.

Thus, with the electron density profile obtained from

$$n_e(\phi) = n_e\ e^{-\phi/T_e}\ ,$$

the potential as a function of ψ is determined by setting

$$n_i(\psi,\phi) = n_e\, e^{-\phi/T_e} .$$

A minor complication arises because n_i at the mirror throat is zero but $n_e(\phi)$ is always positive. To circumvent this problem, we assume the presence of a small, constant ion component, n_c, where

$$n_c = n_e\, e^{-\phi_m/T_e} ,$$

and adjust the above equation to

$$n_c + n_i(\psi,\phi) = n_e\, e^{-\phi_m/T_e} ,$$

where

$$n_e = n_c + n_i(1,0) .$$

Pressure Profiles

As a diagnostic[3] or to update the magnetic field, the perpendicular and parallel pre sures are computed:

$$p_\perp(\psi) = 4\pi \int_0^\infty dv\, v^2 \int_0^1 d\mu\; f(v,\psi,\phi)\, \frac{1}{2}\, m v_\perp^2 ,$$

$$= \frac{1}{2}\, mn\, \langle v_\perp^2 \rangle ,$$

$$p_\parallel(\psi) = mn\, \langle v_\parallel^2 \rangle ,$$

where $v_\perp = v \sin\theta$ and $v_\parallel = v \cos\theta$. When the integrals are transformed to (v_0, μ_0), we obtain the integral appearing in Eq. (8) but with $1/2\ mv^2(1 - \mu^2)$ and $mv^2\mu^2$ as factors in the integrand. The singularity is handled the same way as in the potential profile calculation. We compute β_{max} and B_{min} as diagnostics and B_{min} is the minimum central resultant field required for mirror mode stability. Thus,

$$B_{min}^{2} = \max_{\psi} \left\{ -8\pi \frac{\partial p_\perp}{\partial \psi^2} \right\} ,$$

$$B_{max} = \frac{8\pi\ p_{\perp_0}}{8\pi\ p_{\perp_0} + B_{min}^{2}} ,$$

$$= \frac{8\pi p_{\perp_0}}{B_{v\ min}^{2}} .$$

Updating the Magnetic Field Profile

As a plasma builds up in a vacuum field, the magnetic field is modified by the plasma. This effect also can be simulated in the bounce-average code. The long-thin approximation is used:

$$B^2(z) + 8\pi p_\perp(\psi) = B_v^2(z) ,$$

where

$$p_\perp(\psi) = n_i(\psi) \left< \frac{1}{2} m_i v^2 \right> ,$$

$$= 4\pi \int_0^\infty dv\ v^2 \int_0^1 d\mu \left(\frac{1}{2} m_i v_\perp^{\ 2} \right) f_i(v,\theta,\psi) ,$$

and

$$\mu = \cos\theta,\ v_\perp = v \sin\theta .$$

For this computation, f is a function of ψ rather than of z. First, $p_\perp(\psi)$ is computed and B_0 is obtained from $B_0^2 + 8\pi p_\perp(1) = B_v^0(0)$. From this, we find

$$B_0^2\ \psi^2 + 8\pi p_\perp(\psi) = B_v^2(z) ,$$

which determines $B_v^2 = B_v^2(\psi)$. If $B_v^2(\psi)$ fails to be monotonically increasing in ψ, then the computation fails and equivalently mirror-mode stability is violated. Given $B_v = B_v(\psi)$, we obtain $z = z(\psi)$ and the update is completed.

$\dot{B}$ Terms

As B(z) varies in time, the quantities

$$\mu = \frac{v_0^2 \sin^2\theta_0}{B_0} ,$$

$$J = \int_0^{z_b} dz \, v_\parallel ,$$

remain constant for each orbit. If f has (μ,J) coordinates, there would be no $\dot{B}$ terms and we would have

$$\left.\frac{\partial f}{\partial t}\right|_{\mu,J} = \text{coll, etc.}$$

However, we have

$$\left.\frac{\partial f}{\partial t}\right|_{v_0,\theta_0} + \dot{v}_0 \frac{\partial f}{\partial v_0} + \dot{\theta}_0 \frac{\partial f}{\partial \theta_0} = \text{coll, etc.} ,$$

where $v_0(t)$ and $\theta_0(t)$ are such that $\mu(v_0, \theta_0)$ and $J(v_0, \theta_0)$ are constant.

The form for $\dot{v}_0$ and $\dot{\theta}_0$ is derived as follows:

$$J(v_0,\theta_0) = \int^{z_b} dz \, v_\parallel = \int^{z_b} dz \left(v_0^2 + v_p^2 - v_0^2 \sin^2\theta_0 \frac{B}{B_0}\right)^{1/2} ,$$

$$\frac{dJ}{dt} = \dot{J} + \dot{v}_0 J_{v_0} + \dot{\theta}_0 J_{\theta_0} = 0 ,$$

$$J_{v_0} = \int \frac{dz}{v_\parallel} \left(v_0 - v_0 \sin^2\theta_0 \frac{B}{B_0}\right) ,$$

$$= \int \frac{dz}{v_\parallel} \frac{1}{v_0} \left(v_0^2 - v_0^2 \sin^2\theta_0 \frac{B}{B_0} + v_p^2 - v_p^2\right) ,$$

$$= \frac{1}{v_0} \left(J - \tau \langle v_p^2 \rangle\right) ,$$

where

$$\langle v_p^2 \rangle = \frac{1}{\tau} \int^{z_b} \frac{dz}{v_\parallel} v_p^2(z) ,$$

$$J_{\theta_0} = \int \frac{dz}{v_\parallel}\left(- v_0^2 \sin\theta_0 \cos\theta_0 \frac{B}{B_0}\right),$$

$$= \frac{\cos\theta_0}{\sin\theta_0}\left(J - \tau v_0^2 - \tau \langle v_p^2 \rangle\right),$$

$$\frac{d}{dt}\mu(v_0,\theta_0) = \frac{\dot{v}_0}{v} + \frac{\cos\theta_0}{\sin\theta_0}\dot{\theta}_0 - \frac{\dot{B}_0}{2B_0} = 0 .$$

Thus we can solve for $\dot{v}_0$, $\dot{\theta}_0$ by

$$\dot{v}_0 = v_0\left[-\frac{\dot{J}}{\tau v_0^2} + \frac{\dot{B}_0}{2B_0}\left(1 - \frac{J}{\tau v_0^2} + \frac{\langle v_p^2\rangle}{v_0^2}\right)\right],$$

$$\dot{\theta}_0 = \tan\theta_0\left[\frac{\dot{J}}{\tau v_0^2} + \frac{\dot{B}_0}{2B_0}\left(\frac{J}{\tau v_0^2} - \frac{\langle v_p^2\rangle}{v_0^2}\right)\right].$$

Electron Advancement

At present, three options are available to advance the electron distribution function — E_e(electron energy) is fixed, E_e/E_i is fixed, or E_e is obtained from a rate equation based on the Spitzer energy exchange rate. All options assume f_e is Maxwellian, and $n_e = n_i$.

Numerical Details

GRID

The ion velocity mesh is given by

$$\left\{ v_j \;\middle|\; j = 1, \text{JVI} \right\},$$

and

$$\left\{ \theta_i \;\middle|\; i = 1, \text{ITH} \right\},$$

which is possibly nonuniform. Also, $v_1 = 0$, V_{JVI} = VMAXI (input), $\theta_{ITH} = \pi/2$, and $\sin^2\theta_1 = 1/R_{max}$, where R_{max} = BRMAX (input).

The electron velocity mesh is

$$\left\{ v_j \;\middle|\; j = 1, JV \right\} ,$$

an extension of the ion velocity mesh. We obtain the z mesh from

$$\left\{ \psi_\ell , z_\ell \;\middle|\; \ell = 1, LZ \right\} ,$$

where $\psi_1 = 1$, $Z_1 = 0$, $\psi_{LZ} = R$ (possibly time-varying), and $Z_{LZ} = z_m$ = ZLENGTH.

NUMERICAL ADVANCEMENT OF DISTRIBUTION FUNCTIONS

The ion equation is expressed as

$$\frac{\partial f}{\partial t} = \text{collision terms} + \dot{B}\ \text{terms} + \text{source terms} ,$$

$$= C^{v_0 v_0} \frac{\partial^2 f}{\partial v_0^2} + C^{v_0 \theta_0} \frac{\partial^2 f}{\partial v_0 \partial \theta_0} + C^{\theta_0 \theta_0} \frac{\partial^2 f}{\partial \theta_0^2}$$

$$+ C^{v_0} \frac{\partial f}{\partial v_0} + C^{\theta_0} \frac{\partial f}{\partial \theta_0} + Cf$$

$$- \dot{v}_0 \frac{\partial f}{\partial v_0} - \dot{\theta}_0 \frac{\partial f}{\partial \theta_0} + S - C_x f .$$

This equation is centrally differenced in velocity space and the boundary conditions are taken to be

$$f = 0,$$

for orbits that leave the system, and at $\theta_0 = \pi/2$,

$$\frac{\partial f}{\partial \theta_0} = 0 .$$

The equation is then split as follows:

$$\frac{f^2 - f^1}{\Delta t} = C^{v_0 v_0} \frac{\partial^2 f^2}{\partial v_0^2} + \frac{1}{2} C^{v_0 \theta} \frac{\partial^2 f^2}{\partial v_0 \partial \theta_0}$$

$$+ C^{v_0} \frac{\partial f^2}{\partial v_0} + \frac{1}{2} C f^2 - \dot{v}_0 \frac{\partial f^2}{\partial v_0}$$

$$+ \frac{1}{2} S - \frac{1}{2} C_x \frac{f^2 + f^1}{2} ,$$

$$\frac{f^3 - f^2}{\Delta t} = \frac{1}{2} C^{v_0 \theta_0} \frac{\partial^2 f^2}{\partial v_0 \partial \theta_0} + C^{\theta_0 \theta_0} \frac{\partial^2 f^3}{\partial \theta_0^2}$$

$$+ C^{\theta_0} \frac{\partial f^3}{\partial \theta_0} + C f^3 - \dot{\theta}_0 \frac{\partial f^3}{\partial \theta_0}$$

$$+ \frac{1}{2} S - \frac{1}{2} C_x \frac{f^3 + f^2}{2} ,$$

where $f^1 = f(t)$, f^2 is an intermediate distribution function, and $f^3 = f(t + \Delta t)$. To within splitting errors, the ion equation is advanced fully implicitly.

BOUNDARY

Starting at $z = 0$ and $(\ell = 1)$, the quantity

$$v_{\parallel}{}^2 = v_0^2 + v_p{}^2 - \psi_\ell \, v_0{}^2 \sin^2\theta_0 \; , \; \ell=1, Lz \; ,$$

is tested. As soon as $v_{\parallel}^2 < 0$, then z_b (the bounce point) can be computed. If $v_{\parallel}^2$ is always positive, $z_b = z_m$.

With z_b computed for each v_j and θ_i in the velocity domain, the orbit through (v_j, θ_i) at the midplane is confined only if $z_b < z_m$. The boundary is defined by the index quantities $ILOSS_i$ and $JLOSS_i$ such that

$$ILOSS_j = \min_i \left[i \mid z_b(v_j, \theta_i) < z_m \right] ,$$

$$JLOSS_i = \min_j \left[j \mid z_b(v_j, \theta_i) < z_m \right] .$$

ORBIT INTEGRAL

The orbit integral routine, ORBIT(L), updates the storage $VCOS_{ij} = v_{\parallel}(v_j, \theta_i, z_\ell)$ and $DTAU_{ij} = \delta\tau_\ell$, defined below:

$$\tau\bar{S}(v_0, \theta_0) = \int_0^{z_b} \frac{dz}{v \cos\theta} S(v,\theta,z),$$

$$= \sum_\ell S_\ell \, \delta\tau_\ell , \qquad (9)$$

$$\delta\tau_\ell = \int_{z_{\ell-1}}^{z_\ell} \frac{dz}{v \cos\theta} \frac{\psi(z) - \psi_{\ell-1}}{\psi_\ell - \psi_{\ell-1}} + \int_{z_\ell}^{z_{\ell+1}} \frac{dz}{v \cos\theta} \frac{\psi_{\ell+1} - \psi(z)}{\psi_{\ell+1} - \psi_\ell} ,$$

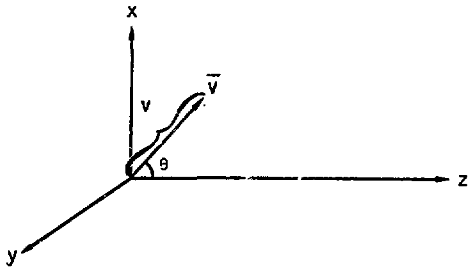
$$= WM_\ell + WP_\ell .$$

If $\ell = 1$, then $WM_\ell = 0$ and if $\delta\tau_\ell$ represents the contribution of the bounce point z_b, then $WP_\ell = 0$.

The form in Eq. (9) results from the assumption that $S(v,\theta,z)$ is linear in ψ between grid points so that

$$S(v,\theta,z) = \sum_\ell S_\ell \, W_\ell(\psi) ,$$

where $W_\ell(\psi)$ is a "tent" function:



The function $\psi(z)$ is usually linear in z,

$$\psi(z) = \psi_\ell + \frac{\Delta\psi_\ell}{\Delta t_\ell} (z - z_\ell) \ .$$

The exception is neai z = 0 where

$$\psi(z) = 1 + \frac{\psi_R - 1}{Z_R^{\ 2}} \ ,$$

for $\psi \leq \psi_R$ with Z_R specified from the input.

This numerical integration is a generalization of the trapezoidal rule. If (v cos θ) is constant and $\psi(z)$ is always linear between grid points, then the $\delta\tau_\ell$ values are the familiar trapezoidal rule weights.

LINE INTEGRAL DIAGNOSTICS

If we let $S(v,\theta,z)$ be some quantity of interest [e.g., $f(v,\theta,z)$], we can define the two quantities,

$$s(z) = 4\pi \int_0^\infty dv \ v^2 \int_0^1 d\mu \ S(v,\theta,z) \ ,$$

$$\bar{S}(v_0, \theta_0) = \frac{1}{\tau} \int_0^{z_b} \frac{dz}{v \cos\theta} S(v,\theta,z) \ .$$

With a change of variables for S(z) and interchanging the order of integration, we obtain the following relationship between S(z) and $\bar{S}(v_0, \theta_0)$:

$$\int_{-z_m}^{z_m} dz\, \frac{s(z)}{\psi} = 8\pi \int_0^\infty dv_0\, v_0^3 \int_0^{\pi/2} d\theta_0 \sin\theta_0 \cos\theta_0\, \tau\bar{S}(v_0, \theta_0) \; .$$

If for example $S(v,\theta,z) = f(v,\theta,z)$, then $s(z) = n(z)$, the local density, and $\bar{S}(v_0, \theta_0) = f(v_0, \theta_0)$, the midplane distribution function. Thus,

$$\int_{-z_m}^{z_m} dz\, \frac{n(z)}{\psi} = 8\pi \int_0^\infty dv_0\, v_0{}^3 \int_0^{\pi/2} d\theta_0 \sin\theta_0 \cos\theta_0\, \tau f(v_0, \theta_0) \; .$$

These two integrals are computed numerically and the values are then outputted; they normally agree to within 1%.

The source terms also can be conveniently verified with these integrals by

$$J(z) = 4\pi \int_0^\infty dv\, v^2 \int_0^1 d\mu\, S(v,\theta,z) \; ,$$

the local current; $\bar{S}(v_0, \theta_0)$ is now a nontrivial bounce-average of the local source. The agreement of the numerical integrals is usually 1 to 5%. The agreement worsens for local 90° sources that are peaked in angle (5° e-fold spread or less); this places a grid limitation on the bounce-average code.

References

1. D. J. BenDaniel and W. P. Allis, "Scattering Loss from Magnetic Mirror Systems," *Plasma Phys. J. Nucl. Energy, C* 4, 31 (1962).
2. M. N. Rosenbluth, W. M. MacDonald, and D. L. Judd, "Fokker-Planck Equation for an Inverse-Square Force," *Phys. Rev.* 107, 1 (1957).
3. A. A. Mirin, *Hybrid II, A Two-Dimensional Multispecies Fokker-Planck Computer Code,* Lawrence Livermore Laboratory, Rept. UCRL-51615, Rev. 1 (1975).
4. J. Killeen and K. D. Marx, *The Solution of the Fokker-Planck Equation for a Mirror-Confined Plasma,* Lawrence Livermore Laboratory, Rept. UCRL-71662 (1969).
5. A. A. Mirin, *Hybrid I, A Two-Dimensional Fokker-Planck Code,* Lawrence Livermore Laboratory, Rept. UCRL-51598 (1974).

PLL/gw/vt

APPENDIX. LISTING OF THE BOUNCE-AVERAGE CODE

```
      PROGRAM AABAV(TAPE2,TAPE3,TAPE4,OUTPUT)

      CLICHE BACOMM
C.... DIMENSIONING PARAMETERS
      PARAMETER( ITH=22, ITP1=ITH+1)
      PARAMETER( JVI=45, JV=150)
      PARAMETER( MX=2, TWOMX=2*MX, TMXP5=2*MX+5)
      PARAMETER( LZ=31)
      PARAMETER( NMQ=205)

C.... GRID STORAGE
      COMMON /GSTORE/ COSS(ITH),CTNN(ITH),
     & DV(JV),DELV(JV),DVI(JV),DELVI(JV),
     & DTH(ITH),DTI(ITH),DELT(ITH),DELTI(ITH),DXL(ITH),  XL(ITH),
     & DCOSS(ITH),DCOSSI(ITH),DZ(LZ),  GMESH,  QRAT(JVI,(-4,TMXP5)),  TH(ITP1),
     & SINN(ITH),  VMAX,VMAXI,V(JV),V2(JV),V3(JV),V4(JV),V5(JV),
     & THDEG(ITH),TINT(ITH),TINL(ITH),  VINT(JVI),VINL(JVI),
     & VI(JV),VI2(JV),VI3(JV),VI4(JV),VI5(JV),  ZLENGTH,Z(LZ)

C.... DISTRIBUTION FUNCTIONS AND DIAGNOSTICS***
      COMMON /FSTORE/ F(ITP1,JVI),FE(JV),FESAVE(JV),
     & ILOSS(JVI),  JLOSS(ITH),  KKOUNT(LZ),
     & PRDEN(LZ),PPAR(LZ),PPERP(LZ),
     & TDENI(NMQ),TDENE(NMQ),TNRGI(NMQ),TNRGE(NMQ),TTIME(NMQ),TPHIM(NMQ),
     & TEDENL(NMQ),TDENL(NMQ),XPLOT(NMQ),YPLOT(NMQ),EDIAG(2),LDIAG(2),
     & ZDENI(LZ),ZDENE(LZ),ZNRGI(LZ),ZNRGE(LZ)

C.... COEFFICIENT STORAGE***
      COMMON /CSTORE/ CC(ITH,JVI),CCT(ITH,JVI),CCTT(ITH,JVI),CCV(ITH,JVI),
     & CCX(ITH,JVI),  DMU(ITH,JVI),
     & CCVT(ITH,JVI),CCVV(ITH,JVI),  EC(JV),ECV(JV),ECVV(JV),EL(JV),ES(JV),
     & SS(ITH,JVI),  TAU(ITH,JVI),  ZBOUNCE(ITH,JVI)

      COMMON /BFSTORE/ BV0,B0,B0MIN,BETA,BETAMAX,
     & BRATIO,BRMAX,BRVAC,BVAC(LZ),BFE(LZ),BF(LZ),BDOT(LZ),  DBSTOP,
     & PHS(LZ),PS(LZ),PHIDOT(LZ)

C.... WORKING ARRAY STORAGE**
      COMMON /ZSTORE/  ALEG(JVI,(0,MX)),AE(JV),AEL(JV),AEV(JV),AEVV(JV),
     & CLOGEE,CLOGIE,CLOGII,  DPLEG((0,TWOMX)),ODPLEG((0,TWOMX)),  EQEE(JV),
     & EQMM(JV),EQNN(JV),ETAU(JV),  FEZ(JV),  PLEG((0,TWOMX)),
     & G(JVI,(0,MX)),GV(JVI,(0,MX)),GVV(JVI,(0,MX)),  CV(JVI),
     & QEE(JVI),QMM(JVI),QNN(JVI),QRR(JVI)

C.... ORBIT INTEGRAL STORAGE  ****
      COMMON /OSTORE/ VCOSP(ITH,JVI),VCOS(ITH,JVI),DTAU(ITH,JVI),OVPOT2(LZ),
     & DETAU(JV),EOV(JV),EOVP(JV),EZBNC(JV)
      DIMENSION EDTAU(ITH,JVI),EDETAU(JV)
      EQUIVALENCE (EDTAU,VCOSP), (EDETAU,EOVP)

C.... MISC STORAGE***
      COMMON /MSTORE/  AMASS,ANUMB,ANUMBI,ANUMB2,ANUMBI2,ANLOSS,APAR2,
     & CHARGE,CLIGHT,CLOGIE0,
     & CONV,CONEE,CONIE,CONII,CDEB,  DEN,DENL,DPOTR,DTIME,DTSET(5),
     & DENL2,
     & EMASS,ERGTKEV,EDENL,ESLOPE,
     & ECONV,ENERGY,EGAMMA,EDEN,EENERGY,ENKEV,EENKEV,
     & ETEMP,EPSSS,EPSMU,ETA,
     & FVS,FTS,FCOS,  GAMMA,GAM1,GAM2,GAM3,GAM4,
     & KBUG,KEEP,KELEC,KPOT,KFIXSS,KIDENT,KSSPS,KTCON,KZGRID,KUF,
```

```
      & KSOURCE,KFL,KIAD,KEAD,KAMBI,KBDOT,KFOUT,KFIN,
      & KTTY,KTTYCON,KSSPASS,KRECOUT,
      & MBELL(5),MAXWELL,MIDPLANE,MES(7),  NSET(5),NTEST,
      & N,NN,NSTOP,NSSMAX,NOUT,NPOT,NCHEC,NCOEF,NRUN,NELIO,
      & PI,PDEN(LZ),PHI(LZ),PSI(LZ),PSII(LZ),PSIH(LZ),POTRATIO,PLTANG,
      & POTENT,PMESH,PSIVAC(LZ),PQT,PQJ,
      & TIME,TEMRATIO,TMESH,TSTART,
      & THIRD,TAUSRC,TAUII,TAUDRAG,TAUMAG,  ZPARAB

C.... SOURCE STORAGE**
       PARAMETER ( ITHS=ITH, NSOR=4)
       COMMON /SSTORE/ EXT(ITHS,NSOR),EXV(JVI,NSOR),SCOS(ITHS),TSMESH,
      & ESENK,ESCUR,
      & STS(NSOR),SJCOS(NSOR),SVS(NSOR),SENKEV(NSOR),
      & SJCUR(NSOR),SJI(NSOR),SJCX(NSOR),SZ(LZ),SPSI(LZ),SPH(LZ),
      & SZDEN(LZ),SENK,SCUR,SCURL,SCURL2,SCX,SCXL,SCXL2,
      & TAUBEAM,ZBEAM

C.... LCM STORAGE  *****
       LCM LCMSTORE
       COMMON /LCMSTORE/ ELCM(JV,LZ),VDOT(ITH,JVI),TDOT(ITH,JVI),
      & FSAVE(ITPI,JVI),EJ1(ITH,JVI),EJ2(ITH,JVI),VP2(ITH,JVI)
       LCM LCMPLOT
       COMMON /LCMPLOT/ TDIAG(NMQ,2)

       ENDCLICHE

       USE BACOMM
       COMMON /GOBCOM/ IGOB(40)
       DIMENSION M(10),MM(10)
       DATA MEND/400204B/, N,IER,NHSP,NDDB0/4(0)/, TIME/0./
       DATA MBELL/4(40020700400207B),407777B/
       DATA KFIN,KFOUT/2(0)/

       NDATA=6RINPUT
 RESTART CONTINUE
       CALL GOB(2001B,IER,200B,M(1))  $$$  GET MESSAGE FROM EXECUTE LINE
       IF(IER.EQ.1) GO TO JUMP1
       NDATA=M(1)
       IF(M(2).EQ.MEND) GO TO JUMP1
       KFIN=M(2)
 JUMP1 CONTINUE
       M=777777B.INT.NDATA  $  NDROP=6R+BA     $  NDROP=NDROP.UN.M
       IF((M.SHR.6).EQ.0) NDROP=4R+BA .UN.M
       IF((M.SHR.12).EQ.0) NDROP=5R+BA  .UN.M
       M(1)=NDROP  $  M(2)=M(3)=0
       IF(N.EQ.0) CALL GOB(101B,IER,1B,M)
C.... GENERATE OUTPUT FILENAMES FROM DROP FILENAME
       CALL APPEND(NDROP,NHSP,NDDB0)
       CALL ASSIGN(2,NDATA)
       CALL ASSIGN(3,0,NHSP)
C.... GENERATE FILE NAME FOR ION DISTRIBUTION OUTPUT
C.... ALSO FILL IN ID

       DO LOOP1A I=0,9
       M=(77B.SHL.6*I)
       IF((M.INT.NDATA).NE.0) I1=I
       IF((M.INT.IGOB(25)).NE.0) I2=I
 LOOP1A CONTINUE
       KFOUT=NDATA.UN.(IRF.SHL.6*(I1+1))
```

```
      IGOB(36)=7HBOX U37
      I=9-I2
      IGOB(37)=IGOB(25).SHL.6*I
      IGOB(37)=IGOB(37).UN.(NDATA.SHR.6*(I1-I+2))
      IGOB(38)=NDATA.SHL.6*(I-I1+8)
      WRITE(3,F02) NDATA,NDROP
  F02 FORMAT(¤DATA FILENAME=¤,A10,2X,¤DROPFILE=¤,A10)
      CALL DD80ID(2HTC,1,1)
      CALL KEEP80(NDD80)
      IF(N.EQ.0) READ(2,F03) (MES(I),I=1,7)
C.... DATA FOR TIMING ROUTINE AND INITIALIZING CALL    ***
      M(1)=¤UFIELD¤ $ M(2)=¤AMBI¤ $ M(3)=¤POTSHAPE¤ $ M(4)=¤COEF¤
      M(5)=¤SOURCE¤ $ M(6)=¤ROSEN¤ $ M(7)=¤PRESSURE¤ $ M(8)=¤IONPROJ¤
      IF(N.EQ.0) CALL TIMEIN(M,8,3)
      CALL DATAIN
      IF(N.GT.0) GO TO JUMP10
      CALL INITIAL
 JUMP10 CONTINUE
      CALL TIME1(1)
      CALL UFIELD
      CALL TIME2(1)
      CALL BOUNDARY
      CALL COEF
      CALL SOURCE
      CALL DENSITY
      CALL GTDIAG
      CALL SSTEST
      CALL DTCON
      CALL TTYINT
      KSTOP=0
      IF(N.GE.NSTOP) KSTOP=KRECOUT=1
      IF(SENSE SWITCH 1) ,JUMP10B
      KSTOP=KRECOUT=1
 JUMP10B CONTINUE
      IF(N.EQ.(N/NOUT)*NOUT) KRECOUT=1
      IF(KSSPASS.LT.3) GO TO JUMP10A
      KRECOUT=1
      CALL RECORD
      CALL DATAIN
      IF(N.GE.NSTOP) GO TO JUMP11
      GO TO JUMP10
 JUMP10A CONTINUE
      CALL RECORD
      IF(KSTOP.EQ.1) GO TO JUMP11
      CALL TIME1(2)
      CALL AMBI
      CALL TIME2(2)
      N=N+1
      TIME=TIME+DTIME
      GO TO JUMP10

      ENTRY ERR(MM)
      KRECOUT=1  $  CALL RECORD
      IEND=1

      DO LOOP11 I=1,8
      IF(MM(I).EQ.77B) GO TO JUMP11A
 LOOP11 IEND=I
 JUMP11A WRITE(3,F03) (MM(I),I=1,IEND)
      IF(KTTYCON.EQ.0) GO TO JUMP11
```

```
      CALL GOB(1400B,IER,5,MBELL)
      PRINT F03, (MM(I),I=1,IEND)
 F03  FORMAT(8A10)
 JUMP11 CONTINUE
      CALL EXIT(0,1)
      IF(KEEP.EQ.1) GO TO JUMP20
      M(1)=□FRB0 DEST.□  $  M(2)=NHSP  $  M(3)=□ ETC. :□
      M(4)=77B  $$$  EOM FLAG FOR CHATCON *******
      CALL CHATCON(6HALLOUT,M,IER)
      M(1)=□DDB0.□  $  M(2)=□FAM.□  $  M(3)=NDDB0
      IF(KEEP.EQ.-1) M(1)=□DEST.□
      CALL CHATCON(4HFROG,M,IER)
 JUMP20 CONTINUE
      CALL ASSIGN(2,0,NDATA,-1)
C.... WRITE OUT ION DISTRIBUTION  *********
      CALL OUTDIST
      CALL OFFMON
      GO TO RESTART
      END
```

```
      SUBROUTINE AMBI
C.... KAMBI=0 FIXED POTENTIAL
C....        1 POTENT=EENKEV*POTRATIO
      USE BACOMM
      DATA PM1,P0,TM1,T0/4(0.)/

      PM1=P0  $  P0=POTENT
      TM1=T0  $  T0=TIME
      PDOT=0.  $  IF(T0.GT.TM1) PDOT=(P0-PM1)/(T0-TM1)
      DPHI=PDOT*DTIME
      IF(KAMBI.EQ.0) DPHI=0.
      POTENT=POTENT+DPHI
      SCALE=POTENT/PHI(LZ)
      DO LOOP1 L=1,LZ
LOOP1 PHI(L)=PHI(L)*SCALE
      CALL BOUNDARY
      CALL IADVANCE
      CALL EADVANCE
      IF(KAMBI.EQ.1) POTENT=EENKEV*POTRATIO
      CALL DENSITY
      QPLUS=ANUMB*DEN
      SCALE=QPLUS/EDEN  $  DO LOOP50 J=1,JV
LOOP50 FE(J)=FE(J)*SCALE
      CALL DENSITY
JUMP100 CONTINUE
      WRITE(3,100) N+1,TIME+DTIME,DEN,DENL,ENKEV,EENKEV,POTENT
100   FORMAT(¤N,T=¤,I5,E10.2,¤SEC¤,5X,¤DEN,L=¤2E12.5,3X,¤E,EE,P=¤,
     1 3E12.5,¤KEV¤)
      RETURN
      END
```

```
      SUBROUTINE APPEND(NDROP,NHSP,NDD80)
      NHSP=(NDROP.SHL.6).UN.1R0  $  NDD80=(NDROP.SHL.12).UN.2R00
      IH=ID=0

      DO LOOP5 IC=1,7
      I=6*(10-IC)  $  I1=77B.SHL.I
      IF(IH) JUMP2,  $  IF(NHSP.INT.I1) ,JUMP2
      IH=1
      MASK=((77B.SHL.54).SHR.54)-(7777B.SHL.(I-6))
      NHSP=NHSP.INT.MASK
      NHSP=NHSP.UN.(2RHH.SHL.(6*(9-IC)))
JUMP2 IF(ID) LOOP5,  $  IF(NDD80.INT.I1) ,LOOP5
      ID=1
      MASK=((77B.SHL.54).SHR.54)-(7777B.SHL.(I-6))
      NDD80=NDD80.INT.MASK
      NDD80=NDD80.UN.(2R00.SHL.(6*(9-IC)))
LOOP5 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BOUNDARY
      USE BACOMM

      DO LOOP10 J=1,JVI
LOOP10 ZBOUNCE(ITH,J)=Z(1)

      DO LOOP15 I=1,ITH
LOOP15 ZBOUNCE(I,1)=Z(LZ)

      DO LOOP20 J=2,JVI
      DO LOOP20 I=1,ITH-1
      VCOSL=V2(J)*COSS(I)**2

      DO LOOP25 L=2,LZ
      VPOT2=PHI(L)*CONV
      VCOS2=V2(J)+VPOT2-PSI(L)*V2(J)*SINN(I)**2
      IF(VCOS2.GT.0.00) GO TO JUMP25
      ZBOUNCE(I,J)=Z(L)+(Z(L)-Z(L-1))*VCOS2/(VCOSL-VCOS2)
      GO TO LOOP20
JUMP25 CONTINUE
      VCOSL=VCOS2
LOOP25 CONTINUE
      ZBOUNCE(I,J)=Z(LZ)
LOOP20 CONTINUE
      IF(KFL.EQ.0) GO TO JUMP30

      DO LOOP28 I=1,ITH
LOOP28 JLOSS(I)=2

      DO LOOP29 J=1,JVI
LOOP29 ILOSS(J)=2
      RETURN

JUMP30 CONTINUE
      ILOSS(1)=ITH

      DO LOOP30 J=2,JVI
      DO LOOP35 I=2,ITH
      ILOSS(J)=I
      IF(ZBOUNCE(I,J).LT.Z(LZ)) GO TO LOOP30
LOOP35 CONTINUE
LOOP30 CONTINUE
      JLOSS(1)=JVI

      DO LOOP40 I=2,ITH-1
      DO LOOP45 J=2,JVI
      JLOSS(I)=J
      IF(ZBOUNCE(I,J).LT.Z(LZ)) GO TO LOOP40
LOOP45 CONTINUE
LOOP40 CONTINUE
      JLOSS(ITH)=JLOSS(ITH-1)

      DO LOOP50 I=1,ITH
      DO LOOP50 J=1,JLOSS(I)-1
LOOP50 F(I,J)=0.
      RETURN
      END
```

```
      SUBROUTINE BTERMS
C.... COMPUTE BDOT TERMS IF DESIRED ACCORDING TO OPTION FLAG KBDOT
C.... KBDOT0,1=DIGIT OF KBDOT CORRESPONDING TO 0,1 POWER OF TEN
C.... KBDOT0=0  NO BDOT TERMS COMPUTED
C....        1  MIDPLANE PARABOLIC WELL VERSION
C....        2  COMPUTE TAU(I,J) ONLY
C....        3  FROM J
C....        4  COMPUTE TAU AND J BUT V,TDOT REMAIN ZERO
C.... KBDOT1=0,1 FOR EXTRAPOLATION OFF,ON
      USE BACOMM
      DIMENSION B1(LZ),B2(LZ)
      DATA T1,T2/2(0.)/

      KBDOT0=MOD(KBDOT,10)  $  KBDOT1=MOD(KBDOT/10,10)
      T1=T2  $  T2=TIME

      DO LOOP10 L=1,LZ
      B1(L)=B2(L)  $  B2(L)=BF(L)
 LOOP10 CONTINUE

      DO LOOP20 L=1,LZ
      IF(KBDOT0.EQ.0) BDOT(L)=PHIDOT(L)=0.
      BFE(L)=BF(L)
      IF(KBDOT0.LE.0) GO TO LOOP20
      IF(T1.GE.T2) GO TO JUMP19
      BDOT(L)=(B2(L)-B1(L))/(T2-T1)
      PHIDOT(L)=(PHI(L)-PHS(L))/(T2-T1)
 JUMP19 CONTINUE
      IF(KBDOT1.LE.0) GO TO LOOP20
C.... EXTRAPOLATION OF BF  ****
      BFE(L)=BF(L)+BDOT(L)*DTIME
      PSI(L)=BFE(L)/BFE $  PSI1(L)=PSI(L)-1 $ PSIH(L)=SQRT(PSI(L))
      PHI(L)=PHI(L)+PHIDOT(L)*DTIME
 LOOP20 CONTINUE
      BRATIO=PSI(LZ)
      IF(KBDOT0.GT.1) GO TO JUMP50
C.... MIDPLANE VERSION OF BDOT TERMS  ***********

      DO LOOP30 J=1,JVI  $  DO LOOP30 I=1,ITH
      VDOT(I,J)=(BDOT/BFE)*V(J)*(2*(BRATIO-1)*SINN(I)**2-COSS(I)**2)/
     1 (4*(BRATIO-1))
      TDOT(I,J)=(BDOT/BFE)*SINN(I)*COSS(I)*(2*BRATIO-1)/
     1 (4*(BRATIO-1))
 LOOP30 CONTINUE
      GO TO JUMP100

 JUMP50 CONTINUE
C.... BOUNCE AVERAGE FOR V,TDOT   **********
      CALL QVSET(0.,TAU,ITH*JVI)
      CALL QACOPY(VDOT,TAU,ITH*JVI)
      CALL QACOPY(TDOT,TAU,ITH*JVI)
      CALL QACOPY(EJ1,EJ2,ITH*JVI)
      CALL QACOPY(EJ2,TAU,ITH*JVI)
      CALL QACOPY(VP2,TAU,ITH*JVI)
      CALL IORBIT(Z,PSI,PHI)
      IF(KBDOT0.GT.2) GO TO JUMP60

      DO LOOP55 L=1,LZ  $  CALL ORBIT(L)
      DO LOOP55 J=1,JVI  $  DO LOOP55 I=1,ITH
      ZB=ZBOUNCE(I,J)
```

```
      IF(ZB.LE.Z(L)) GO TO LOOP55
      TAU(I,J)=TAU(I,J)+DTAU(I,J)
      IF(ZB.GT.Z(L+1)) GO TO LOOP55
      TAU(I,J)=TAU(I,J)+EDTAU(I,J)
LOOP55 CONTINUE
      GO TO JUMP100
JUMP60 CONTINUE

      DO LOOP65 L=1,LZ  $  CALL ORBIT(L)
      DO LOOP65 J=2,JV1  $  DO LOOP65 I=1,ITH
      ZB=ZBOUNCE(I,J)
      IF(ZB.LE.Z(L)) GO TO LOOP65
      S=VCOS(I,J)
      EJ2(I,J)=EJ2(I,J)+DTAU(I,J)*S**2
      TAU(I,J)=TAU(I,J)+DTAU(I,J)
      VPOT2=PHI(L)*CONV
      VP2(I,J)=VP2(I,J)+VPOT2*DTAU(I,J)
      IF(ZB.GT.Z(L+1)) GO TO LOOP65
      VPOT2=VPOT2*(Z(L+1)-ZB)+CONV*PHI(L+1)*(ZB-Z(L))
      VPOT2=VPOT2/(Z(L+1)-Z(L))
      VP2(I,J)=VP2(I,J)+VPOT2*EDTAU(I,J)
      TAU(I,J)=TAU(I,J)+EDTAU(I,J)
LOOP65 CONTINUE
      IF(KBDOTO.EQ.4) GO TO JUMP100
      IF(T1.GE.T2) GO TO JUMP100

      DO LOOP66 J=2,JV1  $  DO LOOP66 I=1,ITH
      S=1/(V(J)**2*TAU(I,J)) $ S1=EJ2(I,J)*S $ S2=VP2(I,J)/V(J)**2
      EJDOT=(EJ2(I,J)-EJ1(I,J))/(T2-T1)
      VDOT(I,J)=.5*V(J)*( (1-S1+S2)*BDOT/BF-2*S*EJDOT )
      TDOT(I,J)=0.
      IF(COSS(I).EQ.0.) GO TO LOOP66
      TDOT(I,J)=(SINN(I)*.5/COSS(I))*( (S1-S2)*BDOT/BF+2*S*EJDOT )
LOOP66 CONTINUE
JUMP100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE COEF
C.... COMPUTE COLLISION OPERATOR
C.... KFIXSS=2 OR GREATER CAUSES COEFFICIENTS TO BE EVALUATED ONCE
C.... KIDENT=0,1 TO SUPRESS DEL2 H AND DEL4 G=F IDENTITY
      USE BACOMM
      DIMENSION CCTTO(ITH,JVI),CCS(ITH,JVI)
      EQUIVALENCE (CCTTO,CCTT), (CCS,CCT)
      DIMENSION GZ((0,MX)),GVZ((0,MX)),GVVZ((0,MX))

      DATA (NRUNS=0)

      IF(N.GT.0 .AND. KFIXSS.GT.1) GO TO JUMP100
      IF(NRUNS.NE.NRUN) GO TO JUMP2
      IF(NCOEF*(N/NCOEF).EQ.N) GO TO JUMP2
      SCALE=DEN/DENSAVE
      DENSAVE=DEN
      IF(ABS(SCALE-1.00).GT.1.00) GO TO JUMP2

      DO LOOP1 I=1,ITH  $  DO LOOP1 J=1,JVI
      CCVV(I,J)=CCVV(I,J)*SCALE
      CCVT(I,J)=CCVT(I,J)*SCALE
      CCTT(I,J)=CCTT(I,J)*SCALE
      CCV(I,J)=CCV(I,J)*SCALE
      CCT(I,J)=CCT(I,J)*SCALE
      CC(I,J)=CC(I,J)*SCALE
 LOOP1 CONTINUE

      DO LOOP2 L=1,LZ
      ZDENI(L)=ZDENI(L)*SCALE
      ZDENE(L)=ZDENE(L)*SCALE
 LOOP2 CONTINUE
      GO TO JUMP100

 JUMP2 CONTINUE
      CALL TIME1(4)
      DENSAVE=DEN  $  NRUNS=NRUN

      DO LOOP10 I=1,ITH  $  DO LOOP10 J=1,JVI
      CCVV(I,J)=CCVT(I,J)=CCTT(I,J)=0.
      CCV(I,J)=CCT(I,J)=CC(I,J)=0.
      TAU(I,J)=0.
 LOOP10 CONTINUE
      CALL IORBIT(Z,PSI,PHI)  $$$ INITIALIZE ORBIT INTEGRAL SR.

      DO LOOP20 L=1,LZ
      CALL ORBIT(L)
      IF(MIDPLANE.EQ.1 .AND. L.GT.1) GO TO LOOP20
      CALL TIME1(6)
      CALL ROSEN(L)
      CALL TIME2(6)
      VPOT2=OVPOT2(L)
      JVZ=1

      DO LOOP20 J=2,JVI
      VZ2=V2(J)+VPOT2
      IF(VZ2.GE.VMAXI**2) GO TO LOOP30  $$$ ZERO CONTRIBUTION TO
C                                         ORBIT INTEGRAL***********
      VZI2=1./VZ2
      VZ=SQRT(VZ2)  $  VZI=1./VZ
      VZ3=VZ2*VZ    $  VZI3=1./VZ3
```

```
      VZ4=VZ2**2       $  VZI4=1./VZ4
C....  INTERPOLATION TO ORBIT VELOCITY, VZ*****

      DO LOOP25 JJ=JVZ,JVI-1
      IF(VZ.LT.V(JJ)) GO TO JUMP25
 LOOP25 JVZ=JJ
 JUMP25 CONTINUE
      VRAT=(VZ-V(JVZ))*DVI(JVZ)
      CVZ=CV(JVZ)+VRAT*(CV(JVZ+1)-CV(JVZ))
      FFEZ=FEZ(JVZ)+VRAT*(FEZ(JVZ+1)-FEZ(JVZ))

      DO LOOP30 M=0,MX
      GZ(M)=G(JVZ,M)+VRAT*(G(JVZ+1,M)-G(JVZ,M))
      GVZ(M)=GV(JVZ,M)+VRAT*(GV(JVZ+1,M)-GV(JVZ,M))
      GVVZ(M)=GVV(JVZ,M)+VRAT*(GVV(JVZ+1,M)-GVV(JVZ,M))
 LOOP30 CONTINUE

      DO LOOP20 I=2,ITH
C.... ORBIT IS THRU V(J),TH(I) AT MIDPLANE****************
      ZB=ZBOUNCE(I,J)
      IF(ZB.LT.Z(L)) GO TO LOOP20
      VP=VCOS(I,J)
      IF(L.GT.1) GO TO JUMP40
      SINZ=SINN(I)  $  SINZ2=SINZ**2
      COSZ=COSS(I)  $  COSZ2=COSZ**2
      VOV=TOT=1.  $  CTOT=TOT*COSS(I)
      VOVV=TOV=TOVV=TOVT=0.
      CTOV=TOV*COSS(I)  $  CTOVV=TOVV*COSS(I)  $  CTOVT=TOVT*COSS(I)
      TOTT=CTOTT=0.
      GO TO JUMP45
 JUMP40 COSZ=VP*VZI  $  COSZ2=COSZ**2
      SINZ2=1.-COSZ2  $  SINZ=SQRT1(SINZ2)
      CUSE=AMAX1(COSS(I),EPSMU)
      TANO=SINN(I)/CUSE
      VOV=VZ*VI(J)  $  VOVV=-VPOT2*VI3(J)
      TOV=-TANO*VPOT2*VZI*VI2(J)
      TOVV=VZI*TOV*(VPOT2*VI2(J)*(TANO**2-1.)-3.)
      TOT=TANO*COSZ/SINZ  $  CTOT=CUSE*TOT
      TOVT=-TOT*VZI*VPOT2*VI2(J)/CUSE**2
      CTOV=TOV*CUSE  $  CTOVV=TOVV*CUSE  $  CTOVT=TOVT*CUSE
      TOTT=TANO*(TOT**2-1.)  $  CTOTT=CUSE*TOTT
 JUMP45 CONTINUE
      CALL GPLEG1(COSZ)    $$$  OBTAINS PLEG(M),DPLEG(M),DDPLEG(M)******
C.... GG, GGV ETC. ARE ROSENBLUTH POTENTIAL AND DERIVATIVES AT ORBIT VELOCITY
C.... AT Z(L)***************
      GGV=GGVV=GGM=GGMM=GGVM=0.

      DO LOOP45 M=0,MX
      GGV=GGV+GVZ(M)*PLEG(2*M)
      GGVV=GGVV+GVVZ(M)*PLEG(2*M)
      GGM=GGM+GZ(M)*DPLEG(2*M)
      GGMM=GGMM+GZ(M)*DDPLEG(2*M)
      GGVM=GGVM+GVZ(M)*DPLEG(2*M)
 LOOP45 CONTINUE
      GGT=-SINZ*GGM
      GGTT=-COSZ*GGM+SINZ2*GGMM
      GGVT=-SINZ*GGVM
C.... CALCULATE ION COEFFICIENTS FOR ORBIT VELOCITY AT Z ****
      AAVV=.5*GAMMA*GGVV
      AAVT=GAMMA*(VZI2*GGVT-VZI3*GGT)
```

```
      AATT=.5*GAMMA*(VZI4*GGTT+VZI3*GGV)
      AAV=GAMMA*(.5*VZI3*GGTT-.5*VZI3*COSZ*GGM+VZI2*GGV+CVZ)
      AAT=GAMMA*(-VZI4*.5*(2.-COSZ2)*GGM/SINZ-VZI3*GGVT+
     & .5*VZI3*COSZ*GGV/SINZ)
      FUSE=F(I,J)
      IF(KIDENT.EQ.0) GO TO JUMP46
      FUSE=0.

      DO LOOP46 M=0,MX
      AFM=ALEG(JVZ,M)+VRAT*(ALEG(JVZ+1,M)-ALEG(JVZ,M))
 LOOP46 FUSE=FUSE+AFM*PLEG(2*M)
 JUMP46 CONTINUE
      AA=4.*PI*GAMMA*(GAM4*ANUMBI2*CLOGIE*FFEZ+CLOGII*FUSE)
      BBVV=VOV**2*AAVV
      BBVT=2.*VOV*TOV*AAVV+TOT*VOV*AAVT
      BBTT=TOV**2*AAVV+TOT*TOV*AAVT+TOT**2*AATT
      BBV=VOV*AAV+VOVV*AAVV
      BBS=CTOV*AAV+CTOT*AAT+CTOVV*AAVV+CTOVT*AAVT+CTOTT*AATT
      BSMAG=.5*SINN(I)*(BDOT(L)*SINN(I)**2-BDOT(I))/BF(I)
      BB=AA
      CCVV(I,J)=CCVV(I,J)+DTAU(I,J)*BBVV
      CCVT(I,J)=CCVT(I,J)+DTAU(I,J)*BBVT
      CCTT(I,J)=CCTT(I,J)+DTAU(I,J)*BBTT
      CCV(I,J)=CCV(I,J)+DTAU(I,J)*BBV
      CCS(I,J)=CCS(I,J)+DTAU(I,J)*BBS
      CC(I,J)=CC(I,J)+DTAU(I,J)*BB
      TAU(I,J)=TAU(I,J)+DTAU(I,J)
      IF(ZB.GT.Z(L+1)) GO TO LOOP20
C....  EVALUATION AT BOUNCE POINT  ********
      IF(MIDPLANE.EQ.1) GO TO LOOP20
      COSZ=COSZ2=0.  $  SINZ=SINZ2=1.
      TOT=CTOT=TOVT=CTOVT=0.
      CTOTT=-SINN(I)
      BBVT=2.*VOV*TOV*AAVV+TOT*VOV*AAVT
      BBTT=TOV**2*AAVV+TOT*TOV*AAVT+TOT**2*AATT
      BBV=VOV*AAV+VOVV*AAVV
      BBS=CTOV*AAV+CTOT*AAT+CTOVV*AAVV+CTOVT*AAVT+CTOTT*AATT
      BB=AA
      CCVV(I,J)=CCVV(I,J)+EDTAU(I,J)*BBVV
      CCVT(I,J)=CCVT(I,J)+EDTAU(I,J)*BBVT
      CCTT(I,J)=CCTT(I,J)+EDTAU(I,J)*BBTT
      CCV(I,J)=CCV(I,J)+EDTAU(I,J)*BBV
      CCS(I,J)=CCS(I,J)+EDTAU(I,J)*BBS
      CC(I,J)=CC(I,J)+EDTAU(I,J)*BB
      TAU(I,J)=TAU(I,J)+EDTAU(I,J)
 LOOP20 CONTINUE

      DO LOOP52 J=1,JVI  $  DO LOOP52 I=1,ITH
      TAUI=1.  $  IF(TAU(I,J).GT.0.00) TAUI=1./TAU(I,J)
      CCVV(I,J)=CCVV(I,J)*TAUI
      CCVT(I,J)=CCVT(I,J)*TAUI
      CCTT(I,J)=CCTT(I,J)*TAUI
      CCV(I,J)=CCV(I,J)*TAUI
      CCS(I,J)=CCS(I,J)*TAUI
      CC(I,J)=CC(I,J)*TAUI
      IF(KFL.EQ.0) GO TO LOOP52
      IF(MIDPLANE.EQ.1) TAUI=V(J)*COSS(I)/Z(LZ)
      PARVM2=V2(J)*COSS(I)**2*PSI(LZ)+CONV*PHI(LZ)-V2(J)*PSII(LZ)
      PFAC=0.  $  ARG=-APAR2*PARVM2
      IF(ARG.LT.10.) PFAC=1/(1+EXP(ARG))
```

```
      CC(I,J)=CC(I,J)-PFAC*TAUI
LOOP52 CONTINUE
      IF(KBUG.EQ.0 .OR. N.NE.NOUT*(N/NOUT)) GO TO JUMP50
      CALL PEEK2(¤CCTTO(I,J)¤,CCTTO,ITH,JVI)
      CALL PEEK2(¤CCS(I,J)¤,CCS,ITH,JVI)
      CALL PEEK2(¤COEF TAU(I,J)¤,TAU,ITH,JVI)
      CALL PEEK2(¤COEF ZBOUNCE(I,J)¤,ZBOUNCE,ITH,JVI)
JUMP50 CONTINUE

      DO LOOP55 J=1,JVI
      CCTT(ITH,J)=CCTTO(ITH,J)-CCS(ITH,J)
      CCT(ITH,J)=0.

      DO LOOP55 I=1,ITH-1
LOOP55 CCT(I,J)=CCS(I,J)/COSS(I)
      CALL TIME2(4)
JUMP100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE COULOG(L)
C....ASSUMES UPDATED ALEG AND FEZ***  ENERGY IN ERGS*******
      USE BACOMM
      OPTIMIZE

      ZDEN=ENG=0.

      DO LOOP10 J=1,JVI
      ZDEN=ZDEN+DELV(J)*V2(J)*ALEG(J,0)
      ENG=ENG+DELV(J)*V4(J)*ALEG(J,0)
 LOOP10 CONTINUE
      IF(ZDEN.LE.0.00) GO TO JUMP10
      ENG=.5*AMASS*ENG/ZDEN
      ZDEN=4.*PI*ZDEN
 JUMP10 CONTINUE
      ZEDEN=EENG=0.

      DO LOOP20 J=1,JV
      ZEDEN=ZEDEN+DELV(J)*V2(J)*FEZ(J)
      EENG=EENG+DELV(J)*V4(J)*FEZ(J)
 LOOP20 CONTINUE
      IF(ZEDEN.LE.0.00) GO TO JUMP20
      EENG=.5*EMASS*EENG/ZEDEN
      ZEDEN=4.*PI*ZEDEN
      DEBYE=CDEB*SQRTF(EENG/ZEDEN)
      ENG=AMAX1(1.E-14*ENERGY,ENG)  $  EENG=AMAX1(1.E-14*EENERGY,EENG)
      SUPEE=ALOG(DEBYE*SQRTF(2.*EENG/EMASS))
      SUPII=ALOG(DEBYE*SQRTF(2.*ENG/AMASS))
      SUPIE=AMAX1(SUPEE,SUPII)
      CLOGEE=CONEE+SUPEE
      CLOGII=CONII+SUPII
      CLOGIE=CONIE+SUPIE
 JUMP20 CONTINUE
C.... STORE Z-DEP QUANTITIES*****
      ZDENI(L)=ZDEN   $   ZDENE(L)=ZEDEN
      ZNRGI(L)=ENG/ERGTKEV   $   ZNRGE(L)=EENG/ERGTKEV
      RETURN
      END
```

```
      SUBROUTINE DATAIN
      USE BACOMM
      DIMENSION SORSCA(NSOR)

      DATA VMAX,VMAXI,GMESH,BRATIO,ZLENGTH/2.E10,3.5E8,1.,2.,100./
      DATA DTIME/1.E-6/, DTSET/.02,.5,1.E-10,1.,1.1/
      DATA NSTOP,NSSMAX,NOUT,NPOT,NCHEC/0,500,9999,2(1)/
      DATA MAXWELL,MIDPLANE/2(0)/
      DATA KBUG,KSSPS,KTCON,KEEP,KELEC,KPOT,KIDENT,KFIXSS,KZGRID/9(0)/
      DATA KUF,KIAD,KEAD,KAMBI,KBDOT/5(0)/
      DATA POTENT,DPOTR,EENKEV/0.,.005,2./, ANLOSS,TEMRATIO/2(0.)/
      DATA PLTANG/225./, EPSSS/.01/, ZPARAB/1.E-10/, NCOEF,NELIO/2(1)/
      DATA AMASS,ANUMB/3.3433E-24,1./, DEN/1.E14/
      DATA ENKEV,FVS,FCOS,FTS/15.,1.,0.,10./
      DATA BVO,TSMESH,TMESH/7000.,2(1.)/
      DATA ZBEAM/100./, EPSMU/1.E-5/, TAUBEAM,SORSCA/1.E90,NSOR(1.00)/
      DATA  KFL/0/, APAR2/1./
      DATA NRUN/0/

      NAMELIST /NLIST/ AMASS,ANUMB,ANLOSS,APAR2, BRATIO,BRMAX,BVO,
     & DEN,DBSTOP,DPOTR,DTIME,DTSET, ENKEV,EENKEV,EPSSS,EXT,ETA,
     & FVS,FCOS,FTS, GMESH,
     & KBUG,KSSPS,KTCON,KEEP,KELEC,KPOT,KFIXSS,KIDENT,KZGRID,
     & KSOURCE,KFL,KEAD,KIAD,KUF,KAMBI,KBDOT,KFOUT,
     & MAXWELL,MIDPLANE, NSTOP,NSSMAX,NOUT,NPOT,NSET,
     & NCOEF,NCHEC,NSSMAX,NELIO,
     & POTENT,PLTANG,PMESH,PSI,
     & SENKEV,SVS,STS,SJCOS,SJCUR,SJI,SJCX,SORSCA,
     & TEMRATIO,TSMESH,TMESH,TAUBEAM,
     & VMAX,VMAXI,  ZLENGTH,ZBEAM,Z,ZPARAB

      NRUN=NRUN+1  $  IF(NRUN.GT.10) NRUN=1
      LUDATA=2
      IF(KTTYCON.EQ.1 .AND. KSSPS.EQ.1) LUDATA=59
      IF(LUDATA.EQ.59) CALL G08(1400B,I,3,MBELL(3))
      IF(LUDATA.EQ.59) PRINT FTTY
 FTTY FORMAT(¤NAMELIST INPUT?¤)
      INPUT DATA NLIST, LUDATA, 3
      IF(NOUT.GT.NMQ) NOUT=100*(NMQ/100)
      IF(NOUT.LE.0) NOUT=1
      IF(NCHEC.LE.0) NCHEC=1+NSTOP/NMQ
      IF(N.GT.0) GO TO JUMP1

C.... DEFAULTS*****
      IF(BRMAX.LE.BRATIO) BRMAX=BRATIO
      IF(MAXWELL.EQ.0) GO TO JUMP1
      IF(TEMRATIO.EQ.0.00) TEMRATIO=EENKEV/ENKEV
      EENKEV=ENKEV*TEMRATIO  $  ETEMP=EENKEV/1.5
      IF(ANLOSS.EQ.0.00) ANLOSS=EXP(-POTENT/ETEMP)
      POTRATIO=AMAX1(0.00,-ALOG(ANLOSS)/1.5)  $$$ SET POTENT/EENKEV
      POTENT=EENKEV*POTRATIO

 JUMP1 CONTINUE
      EENERGY=EENKEV*ERGTKEV   $$$  CONVERSION FROM KEV TO ERGS***
      ETEMP=2.*THIRD*EENKEV
      ENERGY=ENKEV*ERGTKEV

C.... WRITE OUT INITIAL DATA*****
      CALL FRAME
      CALL SETCH(1.,42.,0,0,1,0)
```

```

      DO LOOP10 KO=3,100,97
      CALL HEADER(KO)
      I=ITH  $  JI=JVI  $  J=JV  $  M=MX  $  L=LZ
      WRITE(KO,F01) I,JI,J
F01   FORMAT(//□COMPILED PARAMETERS□/□ITH=□,I3,5X,□JVI=□,I3,5X,□JV=□,I4)
      WRITE(KO,F02) M,L,NSOR,ITHS
F02   FORMAT(□MX=□,I2,5X,□LZ=□,I3,3X,□NSOR,ITHS=□,2I4)
      WRITE(KO,FAA) (MES(I),I=1,7)
FAA   FORMAT(/8A10)
      IF(N.EQ.0) WRITE(KO,□(//□□INITIAL DATA□□)□)
      IF(N.GT.0) WRITE(KO,□(//□□RESTART  N=□□,I6)□) N
      WRITE(KO,F05)   $$ F05 FORMAT(//□*** GRID PARAMETERS ***□)
      WRITE(KO,F06) VMAX,VMAXI,GMESH,TMESH
F06   FORMAT(/□VMAX=□,E12.5,2X,□VMAXI=□,E12.5,2X,□GMESH,TMESH=□,2F7.4)
      WRITE(KO,F07) BRATIO,BRMAX,BV0,ZLENGTH
F07   FORMAT(□BRATIO,BRMAX,BV0=□,2F10.5,E12.5/□ZLENGTH=□,F10.3)
      WRITE(KO,F08) KZGRID,ZPARAB,PMESH
F08   FORMAT(□KZGRID=□,I2,3X,□ZPARAB=□,E11.4,2X,□PMESH=□,F6.4)
      IF(KZGRID.EQ.0) GO TO JUMP5

      DO LOOP5 L=1,LZ
      WRITE(KO,F09) L,Z(L),PSI(L)
F09   FORMAT(□L=□,I3,5X,□Z=□,F10.4,3X,□PSI=□,F10.6)
      IF(PSI(L).LT.0.5) GO TO JUMP5
LOOP5 CONTINUE
JUMP5 CONTINUE
      WRITE(KO,F10)   $$ F10 FORMAT(□1*** TIME STEP CONTROL ***□)
      WRITE(KO,F11) DTIME,EPSSS,KSSPS,KTCON
F11   FORMAT(/□DTIME=□,E12.5,3X,□EPSSS=□,F7.5,3X,□KSSPS,KTCON=□,2I2)
      WRITE(KO,F12) (I,NSET(I),DTSET(I),I=1,5)
F12   FORMAT(/□  I   NSET□,5X,□DTSET□/(1X,I2,1X,I4,3X,E12.5))
      WRITE(KO,F20)   $$ F20 FORMAT(//□*** FLAGS AND PARAMETERS ***□)
      WRITE(KO,F21) NSTOP,NSSMAX,NOUT,NCHEC,NCOEF,NELIO,PLTANG
F21   FORMAT(/□NSTOP,NSSMAX=□,2I5,5X,□NOUT,NCHEC,NCOEF,NELIO=□,4I5/
     1 □PLTANG=□,F7.1)
      WRITE(KO,F22) KEEP,KBUG
F22   FORMAT(□KEEP,KBUG=□,4I2)
      WRITE(KO,F23) KELEC,KPOT,KFIXSS,KIDENT,KSOURCE
F23   FORMAT(□KELEC,KPOT,KFIXSS,KIDENT,KSOURCE=□,5I2)
      WRITE(KO,F23A) KUF,KBDOT,KIAD,KEAD,KAMBI
F23A  FORMAT(□KUF,KBDOT=□,2I3/□KIAD,KEAD,KAMBI=□,3I2)
      WRITE(KO,F25) MAXWELL
F25   FORMAT(□MAXWELL=□,I2)
      WRITE(KO,F26) MIDPLANE
F26   FORMAT(□MIDPLANE=□,I2)
      WRITE(KO,F27) NPOT,ANLOSS,TEMRATIO,POTENT,ETA
F27   FORMAT(□NPOT=□,I2/□ANLOSS=□,E12.5,2X,□TEMRATIO=□,E12.5,3X,
     1 □POTENT,ETA=□,2F8.4)
      WRITE(KO,F28) KFL,APAR2
F28   FORMAT(□KFL,APAR2=□,I2,E12.5)
      WRITE(KO,F29) DBSTOP
F29   FORMAT(□DBSTOP=□,F7.5,□ BETA CUTOFF RELATIVE TO BETAMAX□)
      WRITE(KO,F40)   $$ F40 FORMAT(□1*** INITIAL DISTRIBUTIONS ***□)
      WRITE(KO,F41) EENKEV
F41   FORMAT(/□ELECTRON ENERGY, EENKEV=□,E12.5,□ KEV□)
      WRITE(KO,F42) AMASS,ANUMB
F42   FORMAT(/□ION MASS, AMASS=□,E16.6,□ GRAMS□,
     1 5X,□ANUMB=□,F5.3)
      WRITE(KO,F43) DEN,ENKEV,FVS,FCOS,FTS
```

```
F43  FORMAT(¤DEN=¤,E12.5,2X,¤ION ENERGY, ENKEV=¤,E12.5,¤ KEV¤/
    1 ¤FVS,FCOS,FTS=¤,3E12.5)
     WRITE(KO,F44) KFIN,KFOUT
F44  FORMAT(¤KFIN=¤,A10,3X,¤KFOUT=¤,A10)
     WRITE(KO,F60)  $$ F60 FORMAT(//¤*** SOURCE PARAMETERS ***¤)
     WRITE(KO,F60A) TSMESH,ZBEAM,TAUBEAM
F60A FORMAT(/¤TSMESH=¤,E12.5,¤ ZBEAM=¤,F8.3,¤CM¤,3X,¤TAUBEAM=¤,E12.5)
     WRITE(KO,F60B)
F60B  FORMAT(//¤SOURCE¤/¤NUMBER¤)
     WRITE(KO,F61)
F61  FORMAT(/¤  0¤,5X,¤SENKEV¤,10X,¤SJCOS¤,5X,¤SVS¤,12X,¤STS¤)

     DO LOOP62 NS=1,NSOR
     WRITE(KO,F62) NS,SENKEV(NS),SJCOS(NS),SVS(NS),STS(NS)
F62  FORMAT(1X,I2,F10.3,¤ KEV¤,2X,F10.6,E13 5,F12.3)
LOOP62 CONTINUE
     WRITE(KO,F63)
F63  FORMAT(/¤  0¤,4X,¤SJCUR¤,8X,¤SJI¤,11X,¤SJCX¤)

     DO LOOP64 NS=1,NSOR
     WRITE(KO,F64) NS,SJCUR(NS),SJI(NS),SJCX(NS)
F64  FORMAT(1X,I2,3E13.5)
LOOP64 CONTINUE
LOOP10 CONTINUE

     DO LOOP66 NS=1,NSOR
     SJCUR(NS)=SJCUR(NS)*SORSCA(NS)
     SJI(NS)=SJI(NS)*SORSCA(NS)  $  SJCX(NS)=SJCX(NS)*SORSCA(NS)
LOOP66 CONTINUE

     DO LOOP67 KO=3,100,97  $  DO LOOP67 NS=1,NSOR
     IF(SORSCA(NS).EQ.1.00) GO TO LOOP67
     WRITE(KO,F67) NS,SORSCA(NS)
F67  FORMAT(//¤SOURCE¤,I3,¤ RESCALED W SORSCA=¤,E12.5)
     WRITE(KO,F63)
     WRITE(KO,F64) NS,SJCUR(NS),SJI(NS),SJCX(NS)
LOOP67 CONTINUE

     DO LOOP69 NS=1,NSOR
LOOP69 SORSCA(NS)=1.00

C.... DATA CHECK***
     IF(VMAXI.GT.VMAX) CALL ERR(¤BAD INPUT1¤)
     IF(VMAXI.LT.1.E4) CALL ERR(¤BAD INPUT2¤)
     IF(DEN.LT.100.) CALL ERR(¤DEN NEAR ZERO INITIAL¤)
     IF(POTENT.LT.0.00) CALL ERR(¤POTENT.LT.0.00 INTITIAL¤)

     CALL EMPTY(3)
     RETURN
     END
```

```
      FUNCTION DENFPZ(PHII,L)
      USE BACOMM
      REAL MUR,LAM

      EVALI(MU,SQ)=.5*(MU*SQ+(1.-LAM)*ALOG(MU+SQ))
      HEVAL(MU,SQ)=EVALI(MU,SQ)-MU*SQ

      DENFPZ=0.  $  IF(L.GE.LZ) RETURN
      VPOT2=PHII*CONV
      SUM=0.

      DO LOOP22 J=2,JV1
      LAM=(V2(J)+VPOT2)/(PSI(L)*V2(J))
      IF(LAM.LT.1.00) GO TO JUMP22
C.... LAM GE TO ONE ...  NO IONS AT THIS VELOCITY ARE REFLECTED  *****
      SUMMU=-.5*F(ITH,J)*DCOSS(ITH-1)

      DO LOOP21 I=ILOSS(J),ITH-1
 LOOP21 SUMMU=SUMMU-.5*F(I,J)*(COSS(I+1)-COSS(I-1))
      SUMMU=SUMMU*SQRT(LAM)
      GO TO JUMP26
 JUMP22 CONTINUE
      MUR=0.
      IF(1.-LAM.GT.0.00) MUR=SQRTI(1.-LAM)
      IR=ITH
      IF(MUR.EQ.0.00) GO TO ENDSEARCH

      DO LOOP24 II=1,ITH-1
      I=ITH-II
      IF(COSS(I).GT.MUR) GO TO ENDSEARCH
      IR=I
 LOOP24 CONTINUE
      GO TO LOOP22
 ENDSEARCH CONTINUE
      IL=ILOSS(J)-1
      IF(IL.GE.IR) GO TO LOOP22
      SQ=0.
      IF(MUR**2-1.+LAM.GT.0.00) SQ=SQRTI(MUR**2-1.+LAM)
      SUMMU=0.
      IF(MUR+SQ.LT.1.E-10) GO TO JUMP24
      SUMMU=-F(IR-1,J)*SQ-((F(IR,J)-F(IR-1,J))*DCOSSI(IR-1))*
     & (EVALI(MUR,SQ)-COSS(IR-1)*SQ)
 JUMP24 CONTINUE
      SQ=SQRTI(COSS(IL)**2-1.+LAM)
      SUMMU=SUMMU+F(IL,J)*SQ+
     & ((F(IL+1,J)-F(IL,J))*DCOSSI(IL))*HEVAL(COSS(IL),SQ)

      DO LOOP26 I=IL+1,IR-1
      SQ=SQRTI(COSS(I)**2-1.+LAM)
      SUMMU=SUMMU+HEVAL(COSS(I),SQ)*((F(I+1,J)-F(I,J))*DCOSSI(I)-
     & (F(I,J)-F(I-1,J))*DCOSSI(I-1))
 LOOP26 CONTINUE
 JUMP26 CONTINUE
      SUM=SUM+DELV(J)*V2(J)*SUMMU
 LOOP22 CONTINUE
      DENFPZ=4.*PI*PSIH(L)*SUM
      RETURN

      ENTRY EDENFPZ(PHII,L)
      DENFPZ=0.  $  IF(PHII.GE.POTENT) RETURN
```

```
      VM2=ECONV*POTENT  $  VP2=ECONV*PHII  $  FACTI=PSI(L)/PSI(LZ)
      SUM=0.

      DO LOOP30 J=1,JV
      FACT=V2(J)-VP2
      IF(FACT.LE.0.00) GO TO LOOP30
      IF(V2(J).GT.VM2)  FACT=FACT+FACTI*(VM2-V2(J))
      SUM=SUM+DELV(J)*V(J)*FE(J)*SQRTI(FACT)
LOOP30 CONTINUE
      IF(L.EQ.1) ERAT=ANUMB*DEN/SUM
      DENFPZ=ERAT*SUM
      RETURN
      END
```

```
      SUBROUTINE DENSITY
C.... OBTAIN ENERGY AND DENSITY OF IONS AND ELECTRONS
C.... DEN,EDEN=ION,ELECTRON DENSITY AT MIDPLANE
C.... DENL,EDENL=ION,ELECTRON LINE DENSITY
C.... ENERGY,EENERGY=ION,ELECTRON ENERGY IN KEV
C.... ETEMP=ELECTRON TEMPERATURE IN KEV
      USE BACOMM
      OPTIMIZE

      DEN=ENERGY=0.
      DENL=0.

      DO LOOP10 J=1,JVI
      SUMU=.5*COSS(ITH-1)**2*F(ITH,J)*TAU(ITH,J)
      SUMI=.5*COSS(ITH-1)*F(ITH,J)

      DO LOOP11 I=2,ITH-1
      SUMU=SUMU+.5*(COSS(I-1)**2-COSS(I+1)**2)*F(I,J)*TAU(I,J)
LOOP11 SUMI=SUMI+.5*(COSS(I-1)-COSS(I+1))*F(I,J)
      DEN=DEN+DELV(J)*V2(J)*SUMI
      DENL=DENL+DELV(J)*V3(J)*.5*SUMU
      ENERGY=ENERGY+DELV(J)*V4(J)*SUMI
LOOP10 CONTINUE

      IF(DEN.LE.0.00) CALL ERR(¤DEN LT ZERO DENSITY¤)
      ENERGY=.5*AMASS*ENERGY/DEN
      ENKEV=ENERGY/ERGTKEV
      DEN=4.*PI*DEN
      DENL=8.*PI*DENL*8V0/80
      IF(MIDPLANE.EQ.1) DENL=DEN*Z(LZ)

      EDEN=EDENL=EENERGY=0.

      DO LOOP20 J=1,JV
      EDEN=EDEN+DELV(J)*V2(J)*FE(J)
      EDENL=EDENL+DELV(J)*FE(J)*V(J)*ETAU(J)
      EENERGY=EENERGY+DELV(J)*V4(J)*FE(J)
LOOP20 CONTINUE

      IF(EDEN.LE.0.00) CALL ERR(¤EDEN LT ZERO DENSITY¤)
      EENERGY=.5*EMASS*EENERGY/EDEN
      EENKEV=EENERGY/ERGTKEV  $  ETEMP=2.*THIRD*EENKEV
      EDEN=4.*PI*EDEN
      EDENL=EDENL*8*PI*8V0/80
      IF(MIDPLANE.EQ.1) EDENL=EDEN*Z(LZ)

      RETURN
      END
```

```
      SUBROUTINE DTCON
C.... CONTROL TIME STEP
C.... KTCON=0 SET DTIME=DTSET(I) WHEN N=NSET(I)
C....        =1 SET DTIME FROM CHARACTERISTIC TIME,TAUC
C.... IF KTCON IS SET TO 1 THEN DTSET IS DEFINED AS FOLLOWS:
C.... DTSET(1)=FRACTION OF TAUC, CHARACTERISTIC TIME, DTIME IS SET TO
C....           2  SUBSEQUENT REDUCTION FACTOR WHEN KSSPASS IS 2
C.... DTSET(3,4)=MINUNUM,MAXIMUM DTIME ALLOWED
C.... DTSET(5)=EXPANSION RATE FACTOR DTIME IS ALLOWED
      USE BACOMM

      S=ABS(BDOT(1))/BF(1)
      TAUMAG=1.E90  $  IF(S.GT.0.00) TAUMAG=1/S
C.... TAUII,DRAG COMPUTED IN SR. ROSEN  *******
C.... TAUSRC COMPUTED IN SOURCE

      TAUC=AMIN1(TAUII,TAUDRAG)
      IF(KTCON.GT.0) GO TO JUMP10
      IF(N.LE.0) GO TO JUMP100

      DO LOOP5 I=1,5
 LOOP5 IF(N.EQ.NSET(I)) DTIME=DTSET(I)
      GO TO JUMP100
 JUMP10 CONTINUE
      S=DTIME*DTSET(5)  $  DTIME=TAUC*DTSET(1)
      DTIME=AMAX1(DTIME,DTSET(3))  $  DTIME=AMIN1(DTIME,DTSET(4))
      DTIME=AMIN1(DTIME,.2*TAUSRC)
      IF(KSSPASS.GE.2) DTIME=DTIME*DTSET(2)  $  DTIME=AMIN1(DTIME,S)
 JUMP100 CONTINUE
      IF(DTIME.LE.0.) CALL ERR(¤ZERO TIME STEP DTCON¤)
      RETURN
      END
```

```
      SUBROUTINE EADVANCE
C.... ADVANCE ELECTRONS TO TIME+DTIME
C.... KEAD=0 FORMERLY FOR FP OF ELECTRONS; NOW DOES NOTHING
C.... KEAD=1 RATE EQUATION ON ELECTRON ENERGY
C.... KEAD=2 ELECTRON ENERGY SET TO RATIO OF ION ENERGY
C.... KEAD=3 FIXED ELECTRONS
C.... FOR KEAD NON ZERO ELECTRON DENSITY SET TO ION DENSITY
      USE BACOMM

      IF(KEAD.GE.3) GO TO JUMP64
      IF(KEAD.GT.0) GO TO JUMP50
      GO TO JUMP100

 JUMP50 CONTINUE
      GO TO (JUMP51,JUMP52) ,KEAD
 JUMP51 CONTINUE
C.... RATE EQN FOR ELECTRON TEMPERATURE   *****
      E1=EENKEV  $  D1=DEN  $  Y1=D1*E1
      EEX=3.0761E-37*CLOGIEO*ANUMB**2/AMASS
      EEX=EEX*D1*(ENKEV-E1)/(SQRT(E1+EMASS*ENKEV/AMASS)**3)
      CALL DENSITY
      D2=DEN
      ENDOT=SCUR-(D2-D1)/DTIME
      A=-ETA*ENDOT*2*THIRD/D1
      B=.5*(D1+D2)*EEX
      Y2=Y1*(1+1*.5*DTIME)+DTIME*B
      Y2=Y2/(1-A*.5*DTIME)
      EENKEV=Y2/D2  $  EENERGY=EENKEV*ERGTKEV
      GO TO JUMP60
C.... ELECTRON ENERGY AT FIXED RATIO TO ION ENERGY  ******
 JUMP52 CONTINUE
      CALL DENSITY
      EENERGY=ENERGY*TEMRATIO
 JUMP60 CONTINUE
C.... RECONTSRUCT FE FROM NEW ENERGY  *****
      EVMULT=.75*EMASS/EENERGY

      DO LOOP60 J=1,JV
 LOOP60 FE(J)=EXP(-EVMULT*V2(J))
 JUMP64 CONTINUE
C.... RESCALE SO ELECTRON DENSITY SATISFIES CHARGE NEUTRALITY  ***
      CALL DENSITY
      SCALE=ANUMB*DEN/EDEN

      DO LOOP61 J=1,JV
 LOOP61 FE(J)=FE(J)*SCALE
 JUMP100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ECOEF(L)
      RETURN
      END
```

```
      SUBROUTINE EMOMENTS(L)
C.... MOMENTS OF ELECTRON DIST. AT Z
      USE BACOMM
      OPTIMIZE
      DO LOOP10 J=1,JV
      FEZ(J)=EQMM(J)=EQNN(J)=EQEE(J)=0.
 LOOP10 CONTINUE
      IF(MAXWELL.LE.0) GO TO JUMP20
      SCALE=PRDEN(L)/PRDEN(1)
      DO LOOP15 J=1,JV
 LOOP15 FEZ(J)=SCALE*FE(J)
      GO TO JUMP30
 JUMP20 CONTINUE
C.... OBTAIN FEZ(J) FROM ELECTRON ORBIT EQUATIONS*******
      EVPOT2=PHI(L)*ECONV
      IF(EVPOT2.GT.0.00) GO TO JUMP25
      DO LOOP20 J=1,JV
 LOOP20 FEZ(J)=FE(J)
      GO TO JUMP30
 JUMP25 CONTINUE
      DO LOOP25 J=1,JV
      V0=SQRT(V2(J)+EVPOT2)
      IF(V0.GE.VMAX) GO TO JUMP30
      J0=INTERP(V0)
      FEZ(J)=FE(J0)+(FE(J0+1)-FE(J0))*(V0-V(J0))*DVI(J0)
 LOOP25 CONTINUE
 JUMP30 CONTINUE
      DO LOOP30 JJ=1,JV-1
      J=JV-JJ
      EQMM(J)=EQMM(J+1)+.25*(FEZ(J)+FEZ(J+1))*(V2(J+1)-V2(J))
 LOOP30 CONTINUE
      DO LOOP40 J=1,JV-1
      FBAR=.5*(FEZ(J)+FEZ(J+1))
      EQNN(J+1)=EQNN(J)+FBAR*THIRD*(V3(J+1)-V3(J))
      EQEE(J+1)=EQEE(J)+FBAR*.2*(V5(J+1)-V5(J))
 LOOP40 CONTINUE
      DO LOOP50 J=2,JV
      EQMM(J)=VI2(J)*EQMM(J)
      EQNN(J)=VI3(J)*EQNN(J)
      EQEE(J)=VI5(J)*EQEE(J)
 LOOP50 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE EORBIT(L)
      RETURN
      END
```

```
      SUBROUTINE GETR(GMESH,N,R)
C.... THIS ROUTINE DETERMINES GEOM. GRID RATIO, R, FROM INPUT GMESH,N
C.... N=TOTAL NUMBER OF GRID POINTS
C.... GMESH=LEFTMOST MESH SPACING/EVEN SPACING
C.... R IS SUCH THAT DX(J+1)=R*DX(J) WILL GENERATE DESIRED GRID
C.... GMESH MAYBE GREATER THAN 1.00
      IF(N.LE.2) RETURN
      T=1./GMESH
      WRITE(3,F10) GMESH,T,N
 F10  FORMAT(¤1  GETR OUTPUT¤/¤GMESH,T=¤,2E22.13,¤  N=¤,I5,//5X)
      RB1=2.*(T-1.)/(N-2)
      R=R1=1.+RB1
      IF(N.EQ.3) RETURN
      IF(GMESH.GT.1.00) GO TO JUMP1
      IF((N-3)*RB1/3. .LT. 0.1 ) GO TO JUMP1
      R2=((N-1)*(T-1.)+1.)**(1./(N-1))
      R2=AMIN1(R1,R2)
      R3=T**(1./(N-2))
      T2=(R2**(N-1)-1.)/((N-1)*(R2-1.))
      T3=(R3**(N-1)-1.)/((N-1)*(R3-1.))
      R=R3+(R2-R3)*(T-T3)/(T2-T3)
 JUMP1 CONTINUE
C
      KOUNT=KPASS=0
C
 JUMP5 CONTINUE
      KOUNT=KOUNT+1
      GI=1.  $  GIP=GIPP=0.
      F=FP=FPP=0.
C
      DO LOOP10 I=0,N-2
      F=F+GI
      FP=FP+GIP
      FPP=FPP+GIPP
      GIPP=2.*GIP+R*GIPP
      GIP=GI+R*GIP
      GI=R*GI
 LOOP10 CONTINUE
      F=F/(N-1)  $  FP=FP/(N-1)  $  FPP=FPP/(N-1)
      F=F-T
      SQ=AMAX1(0.,FP**2-2.*F*FPP)
      DR=-2.*F/(FP+SQRTF(SQ))
      K=KOUNT
      R=R+DR
      EPS=T*N*1.E-14
      IF(ABS(F).LT.EPS) KPASS=KPASS+1
      IF(KPASS.GE.3) GO TO JUMP10
      IF(KOUNT.LT. MIN0(50+KPASS,99)) GO TO JUMP5
      R=GMESH=0.
      CALL ERR(¤GETR FAILED¤)
 JUMP10 CONTINUE
      WRITE(3,F15) R,F,KOUNT
 F15  FORMAT(¤GETR SUCCEEDED  R=¤,E22.13,3X,¤F=¤,E22.13,3X,¤ITERATIONS=¤,I3)
      RETURN
      END
```

```
      SUBROUTINE GPLEGO(X)
C.... RETURNS VALUES OF LEGENDRE POLYNOMIALS ORDER M IN PLEG(M) FOR
C.... ARGUEMENT X**************
      USE BACOMM
      OPTIMIZE

      PLEG(0)=1.  $  PLEG(1)=X

      DO LOOP2 M=1,2*MX-1
      PLEG(M+1)=((2*M+1)*X*PLEG(M)-M*PLEG(M-1))/(M+1)
 LOOP2 CONTINUE
      RETURN GPLEGO

      ENTRY GPLEG1(X)
C.... 1ST AND 2ND DERIVATIVES OF LEGENDRE POLYNOMIALS*******
      CALL GPLEGO(X)
      DPLEG(0)=DDPLEG(0)=0.
      DPLEG(1)=1.  $  DDPLEG(1)=0.

      DO LOOP10 M=1,2*MX-1
      DPLEG(M+1)=(2*M+1)*PLEG(M)+DPLEG(M-1)
      DDPLEG(M+1)=(2*M+1)*DPLEG(M)+DDPLEG(M-1)
 LOOP10 CONTINUE
      RETURN GPLEG1
      END
```

```
      SUBROUTINE GRID
C.... GENERATES GRID FROM INPUT VARIABLES: GMESH,VMAXI,VMAX,BRATIO******
      USE BACOMM

      IF(VMAXI.GE.VMAX) CALL ERR(¤VMAXI.GE.VMAX¤)
      V(1)=0.
      EVEN=VMAXI/(JVI-1)
      IF(GMESH.GT.1.0) GMESH=1.
      V(2)=GMESH*EVEN
      V(JVI)=VMAXI
      AIMESH=(V(JVI)-(JVI-1)*V(2))/((JVI-1)*(JVI-2))
      BIMESH=((JVI-1)**2*V(2)-V(JVI))/((JVI-1)*(JVI-2))

      DO LOOP10 J=3,JVI+1
 LOOP10 V(J)=(J-1)*(AIMESH*(J-1)+BIMESH)
      V(JVI)=VMAXI
      IF(V(JVI-1).GE.V(JVI)) CALL ERR(¤BAD GRID1¤)
      DVI=V(JVI+1)-V(JVI)
      V(JV)=VMAX
      EVEN=(V(JV)-V(JVI))/(JV-JVI)
      GEMESH=DVI/EVEN  $  RAT=1.
      CALL GETR(GEMESH,JV-JVI+1,RAT)

      DO LOOP20 J=JVI+1,JV-2
 LOOP20 V(J+1)=V(J)+RAT*(V(J)-V(J-1))

C.... TEST GRID*******

      DO LOOP15 J=2,JV
      IF(V(J).LE.V(J-1)) CALL ERR(¤BAD GRID2 ¤)
 LOOP15 CONTINUE
C.... CALC OF POWERS OF V(J), DIFERENCES AND RECIPROCALS, AND QRAT***

      DO LOOP30 J=1,JV
      V2(J)=V(J)**2
      V3(J)=V(J)**3
      V4(J)=V2(J)**2
      V5(J)=V2(J)*V3(J)
 LOOP30 CONTINUE

      DO LOOP32 J=2,JV
      VI(J)=1./V(J)
      VI2(J)=VI(J)**2
      VI3(J)=VI(J)**3
      VI4(J)=VI2(J)**2
      VI5(J)=VI2(J)*VI3(J)
 LOOP32 CONTINUE
      VI(1)=VI2(1)=VI3(1)=VI4(1)=1.E90

      DO LOOP34 J=2,JV-1
      DV(J)=V(J+1)-V(J)
      DVI(J)=1./DV(J)
      DELV(J)=.5*(V(J+1)-V(J-1))
 LOOP34 CONTINUE
      DV(1)=V(2)-V(1)
      DVI(1)=1./DV(1)
      DELV(1)=.5*(V(2)-V(1))
      DELV(JV)=V(JV)-V(JV-1)

      DO LOOP35 J=1,JV
```

```
 LOOP35 DELVI(J)=1./DELV(J)

        DO LOOP36 M=0,2*MX+5
        QRAT(1,M)=0.
 LOOP36 CONTINUE

        DO LOOP38 J=2,JVI-1

        DO LOOP37 M=1,4
 LOOP37 QRAT(J,-M)=(V(J+1)/V(J))**M
        QRAT(J,0)=ALOG(V(J+1)/V(J))

        DO LOOP38 M=1,2*MX+5
        QRAT(J,M)=(V(J)/V(J+1))**M
 LOOP38 CONTINUE
        IF(KBUG.GT.1) CALL PEEK2(¤QRAT(J,M)¤,QRAT,JVI,2*MX+10)

C.... THETA GRID*****

        THLOSS=ASINF(SQRTF(1./BRMAX))
        TH(1)=THLOSS
        TH(ITH)=.5*PI
        CALL GETR(TMESH,ITH,RAT)  $  S=TMESH*(TH(ITH)-TH(1))/(ITH-1)
        I=ITH  $  TH(I+1)=TH(I)+S  $  TH(I-1)=TH(I)-S

        DO LOOP40 II=2,ITH-2
        I=ITH-II
 LOOP40 TH(I)=TH(I+1)-RAT*(TH(I+2)-TH(I+1))

        DO LOOP41 I=1,ITH
        DTH(I)=TH(I+1)-TH(I)  $  DTI(I)=1/DTH(I)
 LOOP41 CONTINUE
        DELT=DTH/2  $  DELTI=1/DELT

        DO LOOP41A I=2,ITH
        DELT(I)=.5*(DTH(I-1)+DTH(I))  $  DELTI(I)=1./DELT(I)
 LOOP41A CONTINUE

        DO LOOP42 I=1,ITH
        THDEG(I)=TH(I)*57.2957795
        COSS(I)=COSF(TH(I))
        SINN(I)=SINF(TH(I))  $  XL(I)=SINN(I)**2
        CTNN(I)=COSS(I)/SINN(I)
 LOOP42 CONTINUE
        THDEG(ITH)=90.
        COSS(ITH)=0.
        SINN(ITH)=1.
        CTNN(ITH)=0.

        DO LOOP44 I=1,ITH-1
        DCOSS(I)=COSS(I+1)-COSS(I)
        DCOSSI(I)=1./DCOSS(I)
 LOOP44 CONTINUE
        TINT(1)=COSS(1)-COSS(2)  $  TINT(ITH)=COSS(ITH-1)
        TINL(1)=TINT(1)*COSS(1)  $  TINL(ITH)=0.
        DXL(1)=.5*(XL(2)-XL(1))  $  DXL(ITH)=.5*(XL(ITH)-XL(ITH-1))

        DO LOOP50 I=2,ITH-1
        TINT(I)=COSS(I-1)-COSS(I+1)
        TINL(I)=COSS(I)*TINT(I)
```

```
        DXL(I)=(XL(I+1)-XL(I-1))/2
  LOOP50 CONTINUE
        VINT(1)=VINT(JVI)=VINL(1)=VINL(JVI)=0.

        DO LOOP55 J=2,JVI-1
        VINT(J)=2.*PI*V2(J)*DELV(J)
        VINL(J)=2.*VINT(J)*V(J)
  LOOP55 CONTINUE

C.... Z AND PSI GRID  *********
        CALL ZGRID

        WRITE(3,F01) BRMAX
  F01   FORMAT(///¤BRMAX=¤,F10.4)
        WRITE(3,F02)
  F02   FORMAT(/3X,¤I¤,9X,¤TH(I)¤,11X,¤THDEG¤,12X,¤DTH¤,10X,¤CTNN¤)
        WRITE(3,F03)(I,TH(I),THDEG(I),DTH(I),CTNN(I),I=1,ITH)
  F03   FORMAT(1X,I3,4E16.6)

        WRITE(3,F04) GMESH
  F04   FORMAT(///¤GMESH=¤,F10.6/3X,¤J¤,9X,¤V¤,18X,¤DV¤,18X,¤DELV¤)
        WRITE(3,F05)(J,V(J),DV(J),DELV(J),J=1,JV)
  F05   FORMAT(1X,I3,3E17.6)
        CALL PEEKI(¤PSI(L)¤,PSI,LZ)
        CALL PEEKI(¤PSII(L)¤,PSII,LZ)
        CALL PEEKI(¤Z(L)¤,Z,LZ)
        CALL PEEKI(¤BVAC(L)¤,BVAC,LZ)

        CALL EMPTY(3)
        RETURN
        END
```

```
      SUBROUTINE GTDIAG
      USE BACOMM
      DATA LDIAG/10H       BETA,10H    DPPERP/

      EDIAG(1)=BETA
      EDIAG(2)=-8*PI*(PPERP(2)-PPERP(1))/(B0**2*(PSI(2)**2-1))
C.... SOURCE DIAGNOSTIC
      SP=SH=0

      DO LOOP10 J=2,JVI-1  $  DO LOOP10 I=1,ITH
      IF(I.LT.ITH) GO TO JUMP8
      TERM=2*(SS(I,J)-SS(I-1,J))/DTH(I-1)**2
      GO TO JUMP9
 JUMP8 TERM=(SS(I+1,J)-SS(I-1,J))/(2*DTH(I)*COSS(I))
 JUMP9 CONTINUE
      SH=SH+VINT(J)*TINT(I)*V2(J)*SINN(I)**3*TERM
      SP=SP+VINT(J)*TINT(I)*V2(J)*SINN(I)**2*SS(I,J)
 LOOP10 CONTINUE
      SH=SH*AMASS
      SP=SP*AMASS
      EK=4*PI*SP/BV0**2
      WRITE(3,FMT1) SP,SH,EK,BETA,BETAMAX
 FMT1 FORMAT(□SP,H=□,2E11.3,2X,□K=□,E11.3,2X,□BETA,MAX=□,2F8.5)
      RETURN
      END
```

```
      SUBROUTINE HEADER(LIO)
C.... OUTPUT IDENTIFICATION ROUTINE.
C.... LIO=I-O UNIT NUMBER TO WHICH WRITES ARE DIRECTED.
C.... INPUT FILE ASSUMED HOOKED UP TO I-O UNIT 2.
      COMMON /GOBCOM/ IGOB(40)
      DATA KSET/0/, IT,ID,ITP,INFN/4(0)/

      IF(KSET.EQ.1) GO TO JUMP1
      KSET=1
      CALL CLOCK(IT,ID)
      CALL OOOHWD(2,ITP,INFN)
 JUMP1 CONTINUE
      WRITE(LIO,FMT1) IGOB(25),IGOB(31),IGOB(32)
      WRITE(LIO,FMT2) INFN
      WRITE(LIO,FMT3) IT,ID
 FMT1 FORMAT(¤CONTROLLEE¤,A10,2X,¤LOADED AT¤,2A10)
 FMT2 FORMAT(¤INPUT FILENAME=¤,A10,¤ IO-UNIT 2¤)
 FMT3 FORMAT(¤EXECUTION STARTED ¤,2A10)
      RETURN
      END
```

```
      SUBROUTINE IADVANCE
C.... ADVANCE IONS FROM TIME AT N TO TIME+DTIME AT N+1
C.... KBDOT NONZERO INCLUDES BDOT TERMS
C.... IF KBDOT IS NONZERO, KIAD CAUSES ADVANCEMENT OF F/B0**KIAD IN
C.... PLACE OF F.  KIAD=0,1,2 ALLOWED.  THIS OPTION REQUIRED FOR
C.... NUMERICAL STABILITY AT HIGH BETA WITH BDOT TERMS IN EFFECT.

C.... AT HIGH BETA.
      USE BACOMM
      DIMENSION C(JVI),XSI(JVI),FI(ITPI,JVI)
      EQUIVALENCE(C,QEE),(XSI,QMM),(FI,VCOSP)
      OPTIMIZE

      IF(KBDOT.EQ.0) GO TO JUMP1
      S=S2=1  $  S1=0
      IF(KIAD.EQ.0) GO TO JUMP0
      S=1/BF  $  S1=-BDOT/BFE  $  S2=1/BFE
      IF(KIAD.EQ.1) GO TO JUMP0
      S=1/BF**2  $  S1=-2*BDOT/BFE  $  S2=1/BFE**2
 JUMP0 CONTINUE

      DO LOOP0 J=1,JVI  $  DO LOOP0 I=1,ITH
      F(I,J)=F(I,J)*S
      CC(I,J)=CC(I,J)+S1
      SS(I,J)=SS(I,J)*S2
      CCT(I,J)=CCT(I,J)-TDOT(I,J)
      CCV(I,J)=CCV(I,J)-VDOT(I,J)
 LOOP0 CONTINUE
 JUMP1 CONTINUE

      DO LOOP1 J=1,JVI  $  DO LOOP1 I=1,ITH
 LOOP1 FI(I,J)=0.

      DO LOOP2 J=1,JVI
 LOOP2 F(ITH+1,J)=F(ITH-1,J)

      DO LOOP10 I=2,ITH
      C(JLOSS(I)-1)=XSI(JLOSS(I)-1)=0.

      DO LOOP12 J=JLOSS(I),JVI-1
      A1=-DTIME*(DVI(J)*CCVV(I,J)+.5*CCV(I,J))
      A0=DELV(J)-DTIME*(-(DVI(J)+DVI(J-1))*CCVV(I,J)+
     & DELV(J)*(.5*CC(I,J)-.25*CCX(I,J)))
      AM=-DTIME*(DVI(J-1)*CCVV(I,J)-.5*CCV(I,J))
      R=DELV(J)*F(I,J)+DTIME*(.125*CCVT(I,J)*DELTI(I)*
     & (F(I+1,J+1)-F(I-1,J+1)-F(I+1,J-1)+F(I-1,J-1))+
     & DELV(J)*(.5*SS(I,J)-.25*CCX(I,J)*F(I,J)))
      C(J)=A1/(A0-AM*C(J-1))
      XSI(J)=(R-AM*XSI(J-1))/(A0-AM*C(J-1))
 LOOP12 CONTINUE
      FI(I,JVI)=F(I,JVI)
      FI(I,JVI-1)=XSI(JVI-1)

      DO LOOP14 JJ=2,JVI-JLOSS(I)
      J=JVI-JJ
      FI(I,J)=XSI(J)-C(J)*FI(I,J+1)
 LOOP14 CONTINUE
 LOOP10 CONTINUE

      DO LOOP15 J=1,JVI
```

```
      FI(ITH+1,J)=FI(ITH-1,J)
LOOP15 CONTINUE

      DO LOOP20 J=2,JVI-1
      C(ILOSS(J)-1)=XSI(ILOSS(J)-1)=0.

      DO LOOP22 I=ILOSS(J),ITH
      AI=-DTIME*(DTI(I)*CCTT(I,J)+.5*CCT(I,J))
      AO=DELT(I)-DTIME*(-(DTI(I)+DTI(I-1))*CCTT(I,J)+
     1 DELT(I)*(.5*CC(I,J)-.25*CCX(I,J)))
      AM=-DTIME*(DTI(I-1)*CCTT(I,J)-.5*CCT(I,J))
      R=DELT(I)*FI(I,J)+DTIME*(.125*CCVT(I,J)*DELVI(J)*
     & (FI(I+1,J+1)-FI(I-1,J+1)-FI(I+1,J-1)+FI(I-1,J-1))+
     & DELT(I)*(.5*SS(I,J)-.25*CCX(I,J)*FI(I,J)))
      IF(I.EQ.ITH) AM=AM+AI    $$$  SINCE F(ITH-1,J)=F(ITH+1,J) AT BOUNDARY*****
      C(I)=AI/(AO-AM*C(I-1))
      XSI(I)=(R-AM*XSI(I-1))/(AO-AM*C(I-1))
LOOP22 CONTINUE
      F(ITH,J)=XSI(ITH)

      DO LOOP24 II=1,ITH-ILOSS(J)
      I=ITH-II
      F(I,J)=XSI(I)-C(I)*F(I+1,J)
LOOP24 CONTINUE
      F(ITH+1,J)=F(ITH-1,J)
LOOP20 CONTINUE

      DO LOOP32 J=2,JVI-1
      I1=ILOSS(J)  $  C(I1-1)=XSI(I1-1)=0.

      DO LOOP30 I=I1,ITH
      AI=0.
      IF(I.LT.ITH) AI=(XL(I+1)**2*DMU(I+1,J)+XL(I)**2*DMU(I,J))/
     1 (XL(I+1)-XL(I))
      AM=(XL(I)**2*DMU(I,J)+XL(I-1)**2*DMU(I-1,J))/(XL(I)-XL(I-1))
      R=.5*DTIME/(TAU(I,J)*DXL(I))
      AO=1+R*(AI+AM)
      AI=-R*AI  $  AM=-R*AM
      R=F(I,J)
      C(I)=AI/(AO-AM*C(I-1))
      XSI(I)=(R-AM*XSI(I-1))/(AO-AM*C(I-1))
LOOP30 CONTINUE
      F(ITH,J)=XSI(ITH)

      DO LOOP31 II=1,ITH-I1
      I=ITH-II
LOOP31 F(I,J)=XSI(I)-C(I)*F(I+1,J)
LOOP32 CONTINUE

      DO LOOP35 J=1,JVI  $  DO LOOP35 I=1,ITH
      IF(F(I,J).LT.0.00) F(I,J)=0.
      IF(J.LT.JLOSS(I)) F(I,J)=0.
LOOP35 CONTINUE
      IF(K8DOT.EQ.0) GO TO JUMP100
      S=1
      IF(KIAD.EQ.0) GO TO JUMP45
C.... DETERMINE NEW PPERP AT MIDPLANE TO RESTORE STORAGE W
C.... BO AT ADVANCED TIME.
      CALL PRESSO
      H=PPERP
```

```
      IF(KIAD.EQ.1) S=-4*PI*H+SQRT(BVAC**2+16*PI**2*H**2)
      IF(KIAD.EQ.2) S=BVAC**2/(1+8*PI*H)
JUMP45 CONTINUE

      DO LOOP45 J=1,JVI  $  DO LOOP45 I=1,ITH
      F(I,J)=F(I,J)*S
      SS(I,J)=SS(I,J)/S2
      CC(I,J)=CC(I,J)-S1
      CCT(I,J)=CCT(I,J)+TDOT(I,J)
      CCV(I,J)=CCV(I,J)+VDOT(I,J)
LOOP45 CONTINUE
JUMP100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INDIST
C.... KFIN IS 0 FOR NO INPUT FILE FOR IONS
C.... OTHERWISE KFIN SUPPLIES NAME OF DISK FILE TO BE READ
C.... SIMILARLY FOR KFOUT WRT OUTPUT FILE FOR IONS
      USE BACOMM
      DIMENSION FF(1),FT(1),FV(1)
      EQUIVALENCE  (FF,CC), (FT,CCVT), (FV,CCVV)
      DATA IMAX,JMAX/2(1)/

      IF(KFIN.EQ.0) RETURN
      CALL ASSIGN(4,KFIN)
      WRITE(3,FMT1) KFIN
 FMT1 FORMAT(¤HEADER INFO FROM INPUT FILE ¤,A10)
      DO LOOP1 KK=1,3
      READ(4,FMTA) (MES(I),I=1,7)
      WRITE(3,FMTA)  (MES(I),I=1,7)
 FMTA FORMAT(8A10)
 LOOP1 CONTINUE
      READ(4,F01) IMAX,JMAX
 F01  FORMAT(10I5)
      IF(IMAX*JMAX.GT.6*ITH*JVI) CALL ERR(¤OVERWITE HAZARD INDIST¤)
      READ(4,F02) (FT(I),I=1,IMAX)
 F02  FORMAT(5E16.8)
      READ(4,F02) (FV(J),J=1,JMAX)
      READ(4,F02) ((FF(I+(J-1)*IMAX),I=1,IMAX),J=1,JMAX)
C.... INTERPOLATE TO V,TH GRID

      DO LOOP20 J=1,JVI  $  DO LOOP20 I=1,ITH
      DO LOOP10 II=1,IMAX-1
 LOOP10 IF(FT(II).LE.TH(I)) IL=II
      DO LOOP11 JJ=1,JMAX-1
 LOOP11 IF(FV(JJ).LE.V(J)) JL=JJ
      IUSE=IL+(JL-1)*IMAX
      F(I,J)=FF(IUSE)*(FV(JL+1)-V(J.)*(FT(IL+1)-TH(I))
      IUSE=IL+1+(JL-1)*IMAX
      F(I,J)=F(I,J)+FF(IUSE)*(FV(JL+1)-V(J))*(TH(I)-FT(IL))
      IUSE=IL+JL*IMAX
      F(I,J)=F(I,J)+FF(IUSE)*(V(J)-FV(JL))*(FT(IL+1)-TH(I))
      IUSE=IL+1+JL*IMAX
      F(I,J)=F(I,J)+FF(IUSE)*(V(J)-FV(JL))*(TH(I)-FT(IL))
      F(I,J)=F(I,J)/( (FV(JL+1)-FV(JL))*(FT(IL+1)-FT(IL)) )
 LOOP20 CONTINUE
      CALL ASSIGN(4,0,KFIN,-1)
      RETURN

      ENTRY OUTDIST
      IF(KFOUT.EQ.0) RETURN
      I=4+ITH/5+JVI/5+(ITH*JVI)/5
      I=I*12+500
      I=I*3
      CALL ASSIGN(4,KFOUT)
      CALL HEADER(4)
      WRITE(4,F01) ITH,JVI,LZ
      WRITE(4,F02) (TH(I),I=1,ITH)
      WRITE(4,F02) (V(J),J=1,JVI)
      WRITE(4,F02) ( (F(I,J), I=1,ITH), J=1,JVI)
      CALL PEEK1(¤SOURCE¤,SS,ITH*JVI)
      CALL PEEK1(¤Z¤,Z,LZ)
      CALL PEEK1(¤PSI¤,PSI,LZ)
      CALL PEEK1(¤PHI¤,PHI,LZ)
```

```
      CALL PEEK1(□BVAC□,BVAC,LZ)
      CALL PEEK1(□BF□,BF,LZ)
      CALL PEEK1(□PPERP□,PPERP,LZ)
      CALL PEEK1(□PPAR□,PPAR,LZ)
      CALL PEEK1(□PRDEN□,PRDEN,LZ)
      J=4
      CALL SUMMARY(J)
      CALL EMPTY(J)
      KFOUT=0
      RETURN
      END
```

```
      SUBROUTINE INITIAL
      USE BACOMM
      DATA (PI=3.1415926535), (EMASS=9.1066E-28), (CHARGE=4.803E-10)
      DATA (CLIGHT=3.E10), (ERGTKEV=1.602E-9)

C.... CONSTANTS****
      THIRD=1./3.
      FINES=1./137.
      CONV=2.*ERGTKEV*ANUMB/AMASS
      ECONV=2.*ERGTKEV/EMASS
      CDE8=SQRTF(1./(6.*PI*CHARGE**2))
      CONEE=ALOG(EMASS*FINES*CLIGHT/CHARGE**2)-.5
      CONII=ALOG(AMASS*FINES*CLIGHT/CHARGE**2)-.5
      CONIE=ALOG((EMASS*AMASS/(EMASS+AMASS))*2.*FINES*CLIGHT/CHARGE**2)-.5
      GAMMA=4.*PI*(ANUMB*CHARGE)**4/AMASS**2
      EGAMMA=4.*PI*CHARGE**4/EMASS**2
      ANUMB2=ANUMB**2
      ANUMBI=1./ANUMB
      ANUMBI2=ANUMBI**2
      GAM1=EMASS/AMASS
      GAM2=1.-GAM1
      GAM3=1.-AMASS/EMASS
      GAM4=AMASS/EMASS

      CALL GRID
C.... INITIAL PHI OF PSI ARRAY   **********

      DO LOOP1 L=1,LZ
 LOOP1 PHI(L)=POTENT*PSI1(L)/PSI1(LZ)
C.... ELECTRON DISTRIBUTION SHAPE***
      EVMULT=.75*EMASS/EENERGY

      DO LOOP10 J=1,JV
 LOOP10 FE(J)=EXP(-EVMULT*V2(J))
C.... ION SHAPE***
      VPEAK=SQRTF(2.*ENERGY/AMASS) $ IF(FVS.GT.1.E-4) FVS=1/(CONV*FVS)

      DO LOOP20 J=2,JVI-1
 LOOP20 EXV(J)=EXP(-FVS*(VPEAK-V(J))**2)
      EXV(1)=EXV(JVI)=0.
      IF(EXT(ITH).GT.0.00) GO TO JUMP25

      DO LOOP25 I=2,ITH
 LOOP25 EXT(I)=EXP(-FTS*(COSS(I)-FCOS)**2)
      EXT(1)=0.
 JUMP25 CONTINUE

      DO LOOP30 J=1,JVI
      DO LOOP30 I=1,ITH
 LOOP30 F(I,J)=EXV(J)*EXT(I)
C.... CHECK FOR DISK FILE INPUT FOR IONS
      CALL INDIST
C.... RESCALE DISTRIBUTIONS TO INITIAL DENSITY***
      DENSAVE=DEN
      CALL BOUNDARY   $$$ USES PHI AND PSI SUPPLIED ABOVE***
      CALL DENSITY
      SCALE=DENSAVE/DEN

      DO LOOP40 J=1,JVI
      DO LOOP40 I=1,ITH
```

```
LOOP40 F(I,J)=SCALE*F(I,J)
       SCALE=DENSAVE/EDEN

       DO LOOP45 J=1,JV
LOOP45 FE(J)=SCALE*FE(J)
       CALL DENSITY
       RETURN
       END
```

```
      FUNCTION INTERP(VV)
C.... BINARY SEARCH FOR J SUCH THAT V(J).LE.VV.LT.V(J+1)
      USE BACOMM

      IF(VV.LT.V(1).OR.VV.GT.V(JV)) CALL ERR(¤ARG OUT OF BOUNDS INTERP¤)
      JL=1  $  JR=JV

      DO LOOP10 KOUNT=1,JV
      J=(JL+JR)/2
      IF(J.EQ.JL) GO TO JUMP10
      IF(VV-V(J)) JUMP1,,JUMP2
      INTERP=J
      RETURN
 JUMP1 JR=J
      GO TO LOOP10
 JUMP2 JL=J
 LOOP10 CONTINUE
      CALL ERR(¤INTERP FAILURE¤)
 JUMP10 INTERP=J
      RETURN
      END
```

```
      SUBROUTINE IONPROJ(L)
C.... OBTAINS LEGENDRE PROJECTIONS OF IONS DISTRIBUTIONS AT Z(L)*****
      USE BACOMM
      REAL F0(ITH),FZ(ITH),MU(ITH),SUM((0,MX)),LAM,LAMH,LAMHI,CMU(ITH),COS0(ITH)
      OPTIMIZE
      CALL TIMEI(8)

      DO LOOP10 M=0,MX
      DO LOOP10 J=1,JVI
      ALEG(J,M)=0.
 LOOP10 CONTINUE
      IF(L.EQ.LZ) GO TO JUMP40
      VPOT2=PHI(L)*CONV
      J0=1

      DO LOOP20 J=2,JVI
      V02=V2(J)-VPOT2
      IF(V02.LE.0.00) GO TO LOOP20
      V0=SQRTI(V02)

      DO LOOP21 JJ=J0,JVI-1
      J0=JJ
 LOOP21 IF(V0.LT.V(J0+1)) GO TO JUMP21
 JUMP21 CONTINUE
      VRAT=(V0-V(J0))*DVI(J0)

      DO LOOP22 I=1,ITH
      F0(I)=F(I,J0)+VRAT*(F(I,J0+1)-F(I,J0))
 LOOP22 CONTINUE
      LAM=V2(J)/(PSI(L)*V02)
      LAMH=SQRTF(LAM)
      LAMHI=1./LAMH
      COSR=0.
      IF(1.-LAM.GT.0.00) COSR=SQRTF(1.-LAM)
      COS0(ITH)=COSR
      MU(ITH)=0.
      IF(COSR.GT.0.00) GO TO JUMP22
      IF(LAM-1..GT.0.00) MU(ITH)=LAMHI*SQRTI(LAM-1.)
 JUMP22 CONTINUE
      IL=ILOSS(J0+1)-1
      IF(IL.EQ.ITH-1) GO TO LOOP20
      COS0(1)=COSS(IL)
      IF(COS0(1)**2-1.+LAM.LE.0.00) GO TO LOOP20  $$$ ZERO INTERVAL OF INTEG.
      MU(1)=LAMHI*SQRTI(COS0(1)**2-1.+LAM)
      DMU=(MU(ITH)-MU(1))/(ITH-1)

      DO LOOP24 I=2,ITH-1
      MU(I)=MU(1)+(I-1)*DMU
      COS0(I)=SQRTI(1.-LAM*(1.-MU(I)**2))
 LOOP24 CONTINUE
      II=IL
      FZ(1)=F0(IL)
      FRAT=(F0(II+1)-F0(II))*DCOSSI(II)

      DO LOOP26 I=2,ITH
 JUMP25 IF(COS0(I).GE.COSS(II+1)) GO TO JUMP26
      II=II+1
      IF(II.EQ.ITH) GO TO JUMP26
      FRAT=(F0(II+1)-F0(II))*DCOSSI(II)
      GO TO JUMP25
```

```
JUMP26 FZ(I)=FO(I)+FRAT*(COSO(I)-COSS(I))
LOOP26 CONTINUE
       CMU(1)=.5*(MU(1)-MU(2))
       CMU(ITH)=.5*(MU(ITH-1)-MU(ITH))

       DO LOOP30 I=2,ITH-1
LOOP30 CMU(I)=.5*(MU(I-1)-MU(I+1))

       DO LOOP32 M=0,MX
LOOP32 SUM(M)=0.

       DO LOOP35 I=1,ITH
       CALL GPLEGO(MU(I))

       DO LOOP35 M=0,MX
LOOP35 SUM(M)=SUM(M)+CMU(I)*FZ(I)*PLEG(2*M)

       DO LOOP40 M=0,MX
LOOP40 ALEG(J,M)=(4*M+1)*SUM(M)
LOOP20 CONTINUE
JUMP40 CONTINUE
       CALL TIME2(8)
       RETURN
       END
```

```
      SUBROUTINE IORBIT(OZ,OPS,OPH)
      USE BACOMM
      DIMENSION OZ(LZ),OPH(LZ),OPS(LZ)
      DIMENSION OPS1(LZ)
      DATA EPS/0.05/

      IF(ZPARAB.GT.0.00) ZPARAB=AMAX1(ZPARAB,OZ(2))
      LREFF=1

      DO LOOP0 LL=1,LZ
      OPS1(LL)=OPS(LL)-1
      IF(OZ(LL).GT.ZPARAB) GO TO LOOP0
      LREFF=LL
LOOP0 CONTINUE
      ZR=OZ(LREFF)  $  PSIR=OPS(LREFF)  $  PSI1R=OPS1(LREFF)
      VPOTR2=PHI(LREFF)*CONV
      DO LOOP0A LL=1,LZ
LOOP0A OVPOT2(LL)=PHI(LL)*CONV
      DO LOOP0B LL=2,LREFF-1
LOOP0B OVPOT2(LL)=VPOTR2*OPS1(LL)/PSI1R

      DO LOOP1 II=1,ITH  $  DO LOOP1 JJ=1,JV1
      VUSE=AMAX1(V(JJ),1.E-5*V(2))  $  CUSE=AMAX1(COSS(II),EPSMU)
      VP0=VCOSP(II,JJ)=VCOS(II,JJ)=VUSE*CUSE
      VP02=VP0**2
      VPR2=VP02*PSIR+VPOTR2-PSI1R*VUSE**2
      IF(VPR2.GT.0.00) GO TO JUMP1
      ZB=ZR*VP0/SQRT(VP02-VPR2)
      GO TO LOOP1
JUMP1 VPM2=VPR2  $  ZB=OZ(LZ)

      DO LOOP0C LL=LREFF+1,LZ
      VPOT2=OVPOT2(LL)
      VP2=VP02*OPS(LL)+VPOT2-OPS1(LL)*VUSE**2
      IF(VP2.GT.0.00) GO TO LOOP0C
      ZB=OZ(LL-1)+(OZ(LL)-OZ(LL-1))*VPM2/(VPM2-VP2)
      GO TO LOOP1
LOOP0C VPM2=VP2
LOOP1 ZBOUNCE(II,JJ)=ZB
      RETURN

      ENTRY ORBIT(L)
      WEVAL(V1,V2)=2.*THIRD*(2.*V1+V2)/(V1+V2)**2

      DO LOOP40 J=1,JV1  $  DO LOOP40 I=1,ITH
      ZB=ZBOUNCE(I,J)  $  IF(ZB.LE.OZ(L)) GO TO LOOP40
      VPM=VCOS(I,J)  $  VP=VCOSP(I,J)
      VUSE=AMAX1(V(J),1.E-5*V(2))  $  CUSE=AMAX1(COSS(I),EPSMU)
      VP0=VUSE*CUSE  $  VP02=VP0**2
      VPP2=VP**2
      IF(L.LT.LZ) VPOT2=OVPOT2(L+1)
      IF(L.LT.LZ) VPP2=VP02*OPS(L+1)+VPOT2-OPS1(L+1)*V2(J)
      VPP=0.  $  IF(VPP2.GT.0.00) VPP=SQRT1(VPP2)
      Z1=Z2=Z3=OZ(L)
      IF(L.GT.1) Z1=OZ(L-1)  $  IF(L.LT.LZ) Z3=AMIN (OZ(L+1),ZB)
      IF(L.GT.LREFF .OR. LREFF.LE.1) GO TO JUMP11
      VPR2=VP02*PSIR+VPOTR2-PSI1R*V2(J)
      US=1.  $  IF(VPR2-VP02.LT.0.00) US=-1.
      EZ=SQRT1(ABS(VPR2-VP02))/(ZR*VP0)
      E1=EZ*Z1  $  E2=EZ*Z2  $  E3=EZ*Z3
```

```
      ASE1=1-.5*THIRD*US*E1**2  $  SQE1=THIRD-.1375*US*E1**2
      ASE2=1-.5*THIRD*US*E2**2  $  SQE2=THIRD-.1375*US*E2**2
      ASE3=1-.5*THIRD*US*E3**2  $  SQE3=THIRD-.1375*US*E3**2
      IF(US.GT.0.00) GO TO JUMP5
      IF(E2.LE.EPS) GO TO JUMP3
      ASE2=.5*PI  $  SQE2=.25*PI  $  IF(E2.GE.1.00) GO TO JUMP3
      ASE2=ASIN(E2)  $  SQE2=ASE2-E2*SQRT(1.-E2**2)
      SQE2=SQE2*.5/E2**3  $  ASE2=ASE2/E2
 JUMP3 IF(E1.LE.EPS) GO TO JUMP4
      ASE1=.5*PI  $  SQE1=.25*PI  $  IF(E1.GE.1.00) GO TO JUMP4
      ASE1=ASIN(E1)  $  SQE1=ASE1-E1*SQRT(1.-E1**2)
      SQE1=SQE1*.5/E1**3  $  ASE1=ASE1/E1
 JUMP4 IF(E3.LE.EPS) GO TO JUMP11
      ASE3=.5*PI  $  SQE3=.25*PI  $  IF(E3.GE.1.00) GO TO JUMP11
      ASE3=ASIN(E3)  $  SQE3=ASE3-E3*SQRT(1.-E3**2)
      SQE3=SQE3*.5/E3**3  $  ASE3=ASE3/E3
      GO TO JUMP11
 JUMP5 CONTINUE
      IF(E2.LE.EPS) GO TO JUMP6
      SQE2=SQRT(1.+E2**2)  $  ASE2=ALOG(E2+SQE2)
      SQE2=ASE2-E2*SQE2  $  SQE2=-SQE2*.5/E2**3
      ASE2=ASE2/E2
 JUMP6 IF(E1.LE.EPS) GO TO JUMP7
      SQE1=SQRT(1.+E1**2)  $  ASE1=ALOG(E1+SQE1)
      SQE1=ASE1-E1*SQE1  $  SQE1=-SQE1*.5/E1**3
      ASE1=ASE1/E1
 JUMP7 IF(E3.LE.EPS) GO TO JUMP11
      SQE3=SQRT(1.+E3**2)  $  ASE3=ALOG(E3+SQE3)
      SQE3=ASE3-E3*SQE3  $  SQE3=-SQE3*.5/E3**3
      ASE3=ASE3/E3
 JUMP11 CONTINUE
      WM=0.  $  IF(Z1.GE.Z2) GO TO JUMP20
      WM=Z2**3*SQE2-Z1**3*SQE1-Z1**2*Z2*ASE2+Z1**3*ASE1
      WM=WM/(VP0*(Z2**2-Z1**2))
      IF(Z1.GE.ZR) WM=(Z2-Z1)*WEVAL(VPM,VP)
 JUMP20 WP=0.  $  IF(Z2.GE.Z3) GO TO JUMP30
      WP=Z3**3*ASE3-Z3**2*Z2*ASE2-Z3**3*SQE3+Z2**3*SQE2
      WP=WP/(VP0*(Z3**2-Z2**2))
      IF(Z2.GE.ZR) WP=(Z3-Z2)*WEVAL(VPP,VP)
 JUMP30 DTAU(I,J)=WP+WM
      VCOS(I,J)=VP  $  VCOSP(I,J)=VPP
      IF(L.GE.LZ) GO TO LOOP40
      IF(Z8.GT.Z(L+1)) GO TO LOOP40
      Z1=Z2 $ Z2=Z3 $ VPM=VP $ VP=VPP
      ASE1=ASE2 $ ASE2=ASE3 $ SQE1=SQE2 $ SQE2=SQE3
      WM=0.  $  IF(Z1.GE.Z2) GO TO JUMP36
      WM=Z2**3*SQE2-Z1**3*SQE1-Z1**2*Z2*ASE2+Z1**3*ASE1
      WM=WM/(VP0*(Z2**2-Z1**2))
      IF(Z1.GE.ZR) WM=(Z2-Z1)*WEVAL(VPM,VP)
 JUMP36 EDTAU(I,J)=WM  $$$  THIS OVERWRITES VCOSP
 LOOP40 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MOMENTS(M)
C.... OBTAINS MOMENTS OF ION DISTRIBUTION'S 2*M-TH
C.... LEGENDRE POLYNOMIAL PROJECTION, ALEG.J.M).
      USE BACOMM
      DIMENSION ABAR(JVI)
      OPTIMIZE

      DO LOOP10 J=1,JVI-1
      ABAR(J)=.5*(ALEG(J+1,M)+ALEG(J,M))
 LOOP10 CONTINUE
      TFACT=1./(2*M+3)   $   FFACT=1./(2*M+5)
      QNN(1)=TFACT*ALEG(1,M)
      QEE(1)=FFACT*ALEG(1,M)

      DO LOOP20 J=1,JVI-1
      QNN(J+1)=QRAT(J,2*M+3)*(QNN(J)-TFACT*ABAR(J))+TFACT*ABAR(J)
      QEE(J+1)=QRAT(J,2*M+5)*(QEE(J)-FFACT*ABAR(J))+FFACT*ABAR(J)
 LOOP20 CONTINUE

      QMM(JVI)=QRR(JVI)=0.
      IF(M.NE.1) GO TO MNE1

      DO LOOPM1 JJ=1,JVI-2
      J=JVI-JJ
      QMM(J)=QMM(J+1)+QRAT(J,0)*ABAR(J)
      QRR(J)=QRAT(J,-2)*(QRR(J+1)+.5*ABAR(J))-.5*ABAR(J)
 LOOPM1 CONTINUE
      RETURN

 MNE1 CONTINUE
      IF(M.NE.2) GO TO JUMP30

      DO LOOPM2 JJ=1,JVI-2
      J=JVI-JJ
      QMM(J)=QRAT(J,2)*(QMM(J+1)-.5*ABAR(J))+.5*ABAR(J)
      QRR(J)=QRR(J+1)+QRAT(J,0)*ABAR(J)
 LOOPM2 CONTINUE
      RETURN

 JUMP30 CONTINUE
      TFACT=1./(2*M-2)  $  FFACT=1./(2*M-4)

      DO LOOP30 JJ=1,JVI-2
      J=JVI-JJ
      QMM(J)=QRAT(J,2*M-2)*(QMM(J+1)-TFACT*ABAR(J))+TFACT*ABAR(J)
      QRR(J)=QRAT(J,2*M-4)*(QRR(J+1)-FFACT*ABAR(J))+FFACT*ABAR(J)
 LOOP30 CONTINUE
      IF(M.EQ.0) QMM(1)=QMM(2)*V2(2)+ABAR(1)*.5*V2(2)  $$$..USED IN ECOEF ONLY
      RETURN
      END
```

```
      SUBROUTINE PCONTOUR(MES,KOPT,ARR,IMAX,X,IDIM,Y,JDIM)
C.... MES(1) AND MES(2) CONTAIN LABLES FOR X AND Y AXIS
C.... KOPT IS NUMBER OF CONTOURS ON ONE SIDE OF ZERO
C.... MES(3) ETC. IS FOR PLOT ID.  TERMINATED WITH MES(N)=77B
C.... ARR(IMAX,JMAX) DETERMINES LEVEL CURVES FOR CONTOURS
C.... IMAX MUST BE ARRAY'S CORRECT NUMBER OF ROWS
C.... X(IDIM) AND Y(JDIM) ARE PLANE IN WHICH CONTOURS ARE DRAWN
      DIMENSION MES(10),ARR(IDIM,JDIM),X(IDIM),Y(JDIM),C(25)

      CALL FRAME
C.... DETERMINE CONTOUR INTERVAL
      A1=A2=RMIN=RMAX=ARR(1,1)

      DO LOOP40 J=1,JDIM
      CALL AMINMX(ARR,1+IMAX*(J-1),IMAX*J,1,A1,A2)
      RMIN=AMIN1(A1,RMIN)  $  RMAX=AMAX1(A2,RMAX)
 LOOP40 CONTINUE
      KOPT=MIN0(KOPT,21)  $  KOPT=MAX0(KOPT,11)  $$$ KOPT BETWEEN 11 AND 21
      C(2)=AMAX1(ABS(RMIN),RMAX)/KOPT
      IF(C(2).GT.1.E-90) GO TO JUMP40
      CALL SETCH(10.,20.,1,0,2,0)
      WRITE(100,F40) C(2)
 F40  FORMAT(□CONTOUR INTERVAL=□,E12.3)
      CALL CRTBCD(MES(3),5)
      RETURN PCONTOUR

C.... PLOTING
 JUMP40 C(1)=1.E-10*C(2)
      CALL MAPS(X(1),X(IDIM),Y(1),Y(JDIM),.059,.999,.25,.95)
      CALL RCONTR(0,C,0,ARR,IMAX,X,1,IDIM,1,Y,1,JDIM,1)
      CALL SETCH(42.,5.,1,0,3,0)  $  CALL CRTBCD(MES(1),1)
      CALL SETCH(1.,20.5,1,0,3,0)  $  CALL CRTBCD(MES(2),1)
      CALL SETCH(10.,8.,1,0,1)  $  CALL CRTBCD(MES(3),8)
      WRITE(100,F41) C(2),RMIN,RMAX
 F41  FORMAT(□CONTOUR INTERVAL=□,E12.3,5X,□MIN,MAX=□,2E13.4)
      RETURN PCONTOUR

      ENTRY TRANSPOSE(ARR1,ARR2,IDIM,JDIM)
C.... INTERCHANGE ROW AND COLUMNS OF ARR1(IDIM,JDIM) AND
C.... PUT RESULT IN ARR2
C.... BEWARE-- INCORRECT USE OF THIS ROUTINE CAN OVERWRITE   ***********
      DIMENSION ARR1(IDIM,JDIM),ARR2(JDIM,IDIM)

      DO LOOP50 I=1,IDIM  $  DO LOOP50 J=1,JDIM
 LOOP50 ARR2(J,I)=ARR1(I,J)
      RETURN TRANSPOSE
      END
```

```
      SUBROUTINE PEEK(MES,KOPT,ARR,IDIM,JDIM)
      DIMENSION MES(1),X(1),ARR(IDIM,JDIM),N(1)

      ENTRY PEEK1(MES,X,IDIM)
      CALL MESSQ(MES)

      WRITE(3,F02) IDIM,(X(I),I=1,IDIM)
F02   FORMAT(¤LOCATION 1 TO ¤,I3/(10E12.4))
      RETURN PEEK1

      ENTRY PEEKN(MES,N,IDIM)
      CALL MESSQ(MES)

      WRITE(3,F01) IDIM,(N(I),I=1,IDIM)
F01   FORMAT(¤LOCATION 1 TO¤,I3/(10I10))
      RETURN PEEKN

      ENTRY PEEK2(MES,ARR,IDIM,JDIM)
      CALL MESSQ(MES)
      JL=1
JUMP11 CONTINUE
      JR=MIN0(JL+9,JDIM)
      WRITE(3,F10) (J,J=JL,JR)
F10   FORMAT(/3X,¤COL¤,10(I3,8X))
      WRITE(3,F11)
F11   FORMAT(¤ ROW¤/)

      DO LOOP11 I=1,IDIM
      WRITE(3,F12) I,(ARR(I,J),J=JL,JR)
F12   FORMAT(1X,I2,10E11.3)
LOOP11 CONTINUE
      JL=JR+1
      IF(JL.LE.JDIM) GO TO JUMP11
      RETURN PEEK2

      ENTRY MESSQ(MES)
      IEND=1

      DO LOOP30 I=1,8
      IF(MES(I).EQ.77B) GO TO JUMP30   $$$  TEST FOR END OF MESSAGE  ****
LOOP30 IEND=I
JUMP30 WRITE(3,FAA) (MES(I),I=1,IEND)
FAA   FORMAT(/8A10)
      RETURN MESSQ
      END
```

```
      SUBROUTINE PLOTLH(X,Y,N,LAB)
C.... N=NUMBER OF POINTS. LAB IS A LABLE
      DIMENSION X(1),Y(1)
      LCM Y
      DATA YMIN,YMAX/2(0.)/

      X1=.15  $  X2=.99
      Y1=.15  $  Y2=.5
      GO TO JUMP10

      ENTRY PLOTUH(X,Y,N,LAB)
      X1=.15  $  X2=.99
      Y1=.65  $  Y2=.99
 JUMP10 CONTINUE
      CALL AMINMX(Y,1,N,1,YMIN,YMAX)
      YMAX=AMAX1(YMAX,YMIN+.01*YMAX)
      CALL MAPS(X(1),X(N),YMIN,YMAX,X1,X2,Y1,Y2)
      CALL TRACE(X,Y,N)
      S1=85.
      S1=S1*(Y1+Y2)/2-5.
      CALL SETCH(.9*X1,S1,0,0,1,1,0)
      WRITE(100,FMT10) LAB
 FMT10 FORMAT(A10)
      RETURN
      END
```

```
      SUBROUTINE POTSHAPE
C.... KPOT=0 COMPUTE PHI(PSI)
C.... KPOT=1 PHI(PSI) FIXED
C.... KPOT=2 PHI LINEAR IN PSI
C.... MAXWELL=0 FP-ELECTRONS ASSUMED
C.... MAXWELL=1 ELECTRONS MAXWELLIAN W RESIDUAL WARM PLASMA ASSUMED
C.... MAXWELL=2 SKIPS PHI OF PSI CALC
      USE BACOMM

      IF(MOD(N,NPOT).NE.0) RETURN
      PHI(1)=0.  $  PHI(LZ)=POTENT
      IF(KPOT.EQ.0) GO TO JUMP1
      IF(KPOT.EQ.1) GO TO JUMP30

      DO LOOP2 L=2,LZ-1
 LOOP2 PHI(L)=POTENT*PSII(L)/PSII(LZ)
      GO TO JUMP30
 JUMP1 CONTINUE
      IF(MAXWELL.EQ.2) GO TO JUMP30
      IF(POTENT.EQ.0.00) GO TO JUMP30
      CALL TIME1(3)
      QCOLD=0.
      IF(MAXWELL.EQ.1) QCOLD=DEN/(EXP(POTENT/ETEMP)-1.)
      IF(MAXWELL.EQ.0) QE=EDENFPZ(PHI,1)  $$$ INITIALIZE FN

      DO LOOP20 L=2,LZ-1
      PHII=AMAX1(PHI(L-1),PHI(L))
      DPHI=.01*ETEMP

      DO LOOP19 KOUNT=1,50
      IF(MAXWELL.EQ.0) QE=EDENFPZ(PHII,L)
      IF(MAXWELL.EQ.1) QE=(DEN+QCOLD)*EXP(-PHII/ETEMP)
      Q=ANUMB*DENFPZ(PHII,L)-QE+QCOLD
      IF(KOUNT.GT.1) GO TO JUMP10
      QP=QN=Q  $  PP=PN=PHII
      DPHI=-SIGN(DPHI,Q)
      GO TO JUMP16
 JUMP10 CONTINUE
      IF(QN.EQ.QP) GO TO JUMP14
C.... REPLACE ACCORDING TO SIZE OF Q
      IF(ABS(Q).GT.0.01*(ABS(QN)+ABS(QP))) GO TO JUMP14
      IF(ABS(QP).GT.ABS(QN)) GO TO JUMP15
      QN=Q  $  PN=PHII
      GO TO JUMP16
 JUMP14 CONTINUE
C.... REPLACE ACCORDING TO SIGN  **********
      IF(Q) ,,JUMP15
      QN=Q  $  PN=PHII
      GO TO JUMP16
 JUMP15 CONTINUE
      QP=Q  $  PP=PHII
 JUMP16 CONTINUE
      IF(QP.NE.QN) DPHI=(QP*PN-QN*PP)/(QP-QN)-PHII
      IF(QN*QP.LE.0.00) GO TO JUMP18
      DPHI=AMIN1(DPHI,POTENT-PHII) $ DPHI=AMAX1(DPHI,PHI(L-1)-PHII)
 JUMP18 PHII=PHII+DPHI
      IF(KOUNT.LE.2) GO TO LOOP19
      IF(ABS(Q).GT.5.E-2*EDEN) GO TO LOOP19
      IF(ABS(DPHI).GT.1.E-3*EENKEV) GO TO LOOP19
      GO TO JUMP19
```

```
LOOP19 KKOUNT(L)=KOUNT
      CALL ERR(¤TOO MANY PASSES POTSHAPE¤)

JUMP19 PHI(L)=AMAX1(PHI1,PHI(L-1))
LOOP20 CONTINUE
      CALL TIME2(3)
JUMP30 CONTINUE
      QCOLD=0.
      IF(MAXWELL.EQ.1) QCOLD=DEN/(EXP(POTENT/ETEMP)-1.)

      DO LOOP30 L=1,LZ
LOOP30 PDEN(L)=ANUMB*DENFPZ(PHI(L),L)+QCOLD
      RETURN
      END
```

```
      SUBROUTINE PRESSURE
C.... COMPUTE PPERP,PPAR OF PSI  ************
      USE BACOMM
      DIMENSION FO(ITH),XMU(ITH)

      CALL TIME1(7)
      IPRESS=0  $  GO TO JUMP1

      ENTRY PRESS0
C.... ENTER HERE FOR PRESSURES AT Z=0 ONLY
      IPRESS=1
 JUMP1 CONTINUE

      DO LOOP50 L=1,LZ-1
      PP=PV=PR=0.
      VPOT2=CONV*PHI(L)

      DO LOOP40 J=2,JV1-1
      IL=1  $  IR=ITH
      VZ2=V2(J)+VPOT2
      XA=(PSI(L)*V2(J)-VZ2)/(PSI(L)*V2(J))
      FO(1)=F(1,J)  $  XMU(1)=COSS(1)

      DO LOOP10 I=2,ITH
      FO(I)=F(I,J)  $  XMU(I)=AMAX1(COSS(I),1.E-5)
      IF(FO(I).LE.0.00 .AND. IL+1.EQ.I) IL=I
      IF(COSS(I)**2-XA.GT.0.00) GO TO LOOP10
      IR=I  $  XMU(I)=SQRT(AMAX1(XA*(1.+1.E-10),1.E-10))
      FO(I)=FO(I-1)*(COSS(I)-XMU(I))+FO(I)*(XMU(I)-COSS(I-1))
      FO(I)=FO(I)/(COSS(I)-COSS(I-1))
      GO TO JUMP10
 LOOP10 CONTINUE
 JUMP10 CONTINUE
      IF(IL.GE.IR) GO TO LOOP40
      SQ=SQRT1(XMU(IL)**2-XA)  $  XLOG=LOG1(XMU(IL)+SQ)
      E1=SQ  $  H1=.5*(XMU(IL)*SQ-XA*XLOG)
      E2=SQ**3/3.  $  H2=XMU(IL)*SQ**3/12.-.125*XA*(XMU(IL)*SQ-XA*XLOG)
      SUM1=-FO(IL)*E1+(FO(IL+1)-FO(IL))*H1/(XMU(IL+1)-XMU(IL))
      SUM2=-FO(IL)*E2+(FO(IL+1)-FO(IL))*H2/(XMU(IL+1)-XMU(IL))
      SQ=SQRT1(XMU(IR)**2-XA)  $  XLOG=LOG1(XMU(IR)+SQ)
      E1=SQ  $  H1=.5*(XMU(IR)*SQ-XA*XLOG)
      E2=SQ**3/3.  $  H2=XMU(IR)*SQ**3/12.-.125*XA*(XMU(IR)*SQ-XA*XLOG)
      SUM1=SUM1+FO(IR)*E1-
     1 (FO(IR)-FO(IR-1))*H1/(XMU(IR)-XMU(IR-1))
      SUM2=SUM2+FO(IR)*E2-
     1 (FO(IR)-FO(IR-1))*H2/(XMU(IR)-XMU(IR-1))

      DO LOOP20 I=IL+1,IR-1
      SQ=SQRT1(XMU(I)**2-XA)  $  XLOG=LOG1(XMU(I)+SQ)
      E1=SQ  $  H1=.5*(XMU(I)*SQ-XA*XLOG)
      E2=SQ**3/3.  $  H2=XMU(I)*SQ**3/12.-.125*XA*(XMU(I)*SQ-XA*XLOG)
      SUM1=SUM1+H1*( (FO(I+1)-FO(I))/(XMU(I+1)-XMU(I))-
     1 (FO(I)-FO(I-1))/(XMU(I)-XMU(I-1)) )
      SUM2=SUM2+H2*( (FO(I+1)-FO(I))/(XMU(I+1)-XMU(I))-
     1 (FO(I)-FO(I-1))/(XMU(I)-XMU(I-1)) )
 LOOP20 CONTINUE
      SUM1=-SUM1  $  SUM2=-SUM2
      PP=PP+DELV(J)*V2(J)*VZ2*SUM2
      PV=PV+DELV(J)*V2(J)*VZ2*SUM1
      PR=PR+DELV(J)*V2(J)*SUM1
```

```
 LOOP40 CONTINUE
       PPAR(L)=PP*AMASS*4.*PI*PSI(L)*PSIH(L)
       PV=PV*AMASS*4.*PI*PSIH(L)
       PPERP(L)=.5*(PV-PPAR(L))
       PRDEN(L)=4.*PI*PSIH(L)*PR
C.... ELECTRON CONTRIBUTION TO PRESSURES NOT PRESENTLY INCLUDED  *****
       IF(IPRESS.EQ.1) RETURN  $$$***  L=1 ONLY FROM PRESS0 ENTRY  ******
 LOOP50 CONTINUE
C.... DETERMINE B0MIN, MINIMUM FIELD FOR MIRROR-MODE STABILITY
C.... BETAMAX IS BETA-PERP FOR THIS FIELD
       B0MIN=0.

       DO LOOP60 L=2,LZ
       PDERIV=PSI(L)**2-PSI(L-1)**2
       PDERIV=(PPERP(L-1)-PPERP(L))/PDERIV
       B0MIN=AMAX1(B0MIN,PDERIV)
 LOOP60 CONTINUE
       B0MIN=SQRT(8*PI*B0MIN)
       BETAMAX=PPERP/(PPERP+B0MIN**2/(8*PI))
       CALL TIME2(7)
       RETURN
       END
```

```
      SUBROUTINE RECORD
C.... THIS ROUTINE STORES TIME DEP QUANTITIES AND
C.... MAKES CALL TO RECOUT WHEN KRECOUT IS SET FROM 0 TO 1
C.... THE FLAG KRECOUT IS SET IN MAIN
      USE BACOMM
      DATA NN/0/

      IF(N.EQ.NCHEC*(N/NCHEC)) GO TO JUMP1
      IF(KRECOUT.EQ.1) GO TO JUMP1
      RETURN

 JUMP1 CONTINUE
C.... STORE VALUES AT CURRENT TIME
      IF(NN.EQ.0 .OR. TTIME(NN).NE.TIME) NN=NN+1
      TTIME(NN)=TIME
      TDENI(NN)=DEN
      TDENL(NN)=DENL
      TEDENL(NN)=EDENL
      TDENE(NN)=EDEN
      TNRGI(NN)=ENKEV
      TNRGE(NN)=EENKEV
C.... EDIAG DEFINED IN SR. GTDIAG
      TDIAG(NN,1)=EDIAG(1)
      TDIAG(NN,2)=EDIAG(2)
      TPHIM(NN)=POTENT
      IF(KRECOUT.EQ.0) RETURN

      KRECOUT=0
      CALL RECOUT
      IF(NMQ-NN.LT.NOUT/NCHEC+1) NN=0
      RETURN
      END
```

```
      SUBROUTINE RECOUT
      USE BACOMM
      DIMENSION M(20)
      DIMENSION DCONT(30),FCONT(JVI,ITH), PSICOM(LZ)
      DIMENSION FPA(ITH,JVI),FPE(JV)
      EQUIVALENCE (FPA,VCOSP), (FESAVE,FPE)
      EQUIVALENCE (FCONT,VCOSP), (PSICOMM,VCOSP)
      DATA EMIN,FMIN,FMAX/3(0.)/

      IF(NN.LE.1 .OR. NOUT.LE.1) GO TO JUMP19  $$$  NO TIME DEPENDANT OUTPUT
C.... OUTPUT TIME DEP ARRAYS***************

C.... PLOT ELECTRON ENERGY AND DENSITY IN TIME******
      CALL FRAME
      CALL SETCH(35.,41.,0,0,1)
      WRITE(100,F41) N,TIME,DTIME
      CALL AMINMX(TNRGE,1,NN,1,EMIN,EMAX)
      EMAX=AMAX1(EMAX,EMIN+.01*EMAX)
      CALL AMINMX(TDENE,1,NN,1,DMIN,DMAX)
      DMAX=AMAX1(DMAX,DMIN+.01*DMAX)
      CALL MAPS(TTIME(1),TTIME(NN),DMIN,DMAX,.15,.99,.65,.99)
      CALL TRACE(TTIME,TDENE,NN)
      CALL MAPS(TTIME(1),TTIME(NN),EMIN,EMAX,.15,.99,.15,.5)
      CALL TRACE(TTIME,TNRGE,NN)
      CALL SETCH(17.,3.,1,0,1,0)
      WRITE(100,F01) TDENE(NN),TNRGE(NN),TTIME(NN)
 F01  FORMAT(□DENSITY=□,E16.6,5X,□ENERGY=□,E16.6,□KEV□//□TIME=□,E16.6)
      CALL SETCH(1.,17.,1,0,1,1)
      WRITE(100,F02)
 F02  FORMAT(□ELECTRON  ENERGY□,30X,□ DENSITY□)

C.... PLOT ION DENSITY AND ENERGY IN TIME********
      CALL FRAME
      CALL AMINMX(TDENI,1,NN,1,DMIN,DMAX)
      DMAX=AMAX1(DMAX,DMIN+.01*DMAX)
      CALL AMINMX(TNRGI,1,NN,1,EMIN,EMAX)
      EMAX=AMAX1(EMAX,EMIN+.01*EMAX)
      CALL MAPS(TTIME(1),TTIME(NN),DMIN,DMAX,.15,.99,.65,.99)
      CALL TRACE(TTIME,TDENI,NN)
      CALL MAPS(TTIME(1),TTIME(NN),EMIN,EMAX,.15,.99,.15,.5)
      CALL TRACE(TTIME,TNRGI,NN)
      CALL SETCH(17.,3.,1,0,1,0)
      WRITE(100,F05) DEN,TNRGI(NN),TTIME(NN)
 F05  FORMAT(□DENSITY=□,E16.6,5X,□ENERGY=□,E16.6,□KEV□//□TIME=□,E16.6)
      CALL SETCH(1.,17.,1,0,1,1)
      WRITE(100,F06)
 F06  FORMAT(□ION  ENERGY□,30X,□DENSITY□)
      CALL FRAME

C.... PLOT LINE DENSITY OF IONS AND POTENT IN TIME  **********
      CALL AMINMX(TDENL,2,NN,1,DMIN,DMAX)
      DMAX=AMAX1(DMAX,AMAXAF(TEDENL,1,NN))
      DMAX=AMAX1(DMAX,DMIN+.01*DMAX)
      CALL MAPS(TTIME(1),TTIME(NN),DMIN,DMAX,.15,.99,.65,.99)
      CALL SETPCH(0,0,1,0,150)
      CALL TRACEC(1H1,TTIME,TDENL,NN)

      CALL AMINMX(TPHIM,1,NN,1,DMIN,DMAX)
      DMAX=AMAX1(DMAX,DMIN+.01*DMAX)
      CALL MAPS(TTIME(1),TTIME(NN),DMIN,DMAX,.15,.99,.15,.5
```

```
      CALL TRACE(TTIME,TPHIM,NN)
      CALL SETCH(1.,17.,1,0,1,1)
      WRITE(100,F07)
F07   FORMAT(□AMBIPOLAR POTENTIAL□,30X □LINE DENSITY□/□UNITS ARE KEV□)
      CALL SETCH(15.,3.,1,0,1,0)
      WRITE(100,F08) POTENT,TIME
F08   FORMAT(□POTENT=□,E12.4,□ KEV□/□TIME=□,E11.3,□ SEC□)
C.... PLOT TIME DEPENDENT DIAGNOSTCS FROM SR. GTDIAG
      CALL FRAME
      CALL QACOPY(YPLOT,TDIAG(1,2),NN)
      CALL PLOTUH(TTIME,YPLOT,NN,LDIAG(2))
      CALL QACOPY(YPLOT,TDIAG(1,1),NN)
      CALL PLOTLH(TTIME,YPLOT,NN,LDIAG(1))
      CALL SETCH(17.,  ,1,0,1,0)
      WRITE(100,F08A) EDIAG(1),EDIAG(2)
F08A FORMAT(□EPAR0,L=□,2E12.5,□KEV□)
JUMP19 CONTINUE
C.... BOUNCE AVERAGED AND Z-DEPENDANT QUANTITIES OUTPUTTED BELOW  *******

C.... SOURCES
      M(1)=4H'066  $  M(2)=6H'L'017
      M(3)=□SS(I,J)□  $  M(4)=□IONS□  $  M(5)=778
      CALL TRANSPOSE(SS,FCONT,ITH,JVI)
      CALL PCONTOUR(M,2),FCONT,JVI,V,JVI,THDEG,ITH)
      WRITE(100,F04) SCUR,SENK
F04   FORMAT(□MIDPLANE SOURCE=□,E13.5,□PART/SEC-CM3□,5X,E12.5,□KEV□)
      WRITE(100,F09) SCX,SCUR-SCX
F09   FORMAT(□CX=□,E13.5,3X,□IONIZATION=□,E13.5)
      WRITE(100,F10) SCURL,SCURL2,SCXL,SCXL2,SCURL-SCXL
F10   FORMAT(□LINE INTEG. OF SOURCE=□,2E13.5,□PER CM2□/□CX=□,2E13.5,
     & 2X,□IONIZATION=□,E13.5)
      WRITE(100,F41) N,TIME,DTIME

C.... PLOT ION & ELECTRON ENERGY IN Z***********
      CALL FRAME
C.... ELECTRON ENERGY PLOT IN UPPER HALF FRAME****
      CALL AMINMX(ZNRGE,1,LZ,1,EMIN,EMAX)
      EMAX=AMAX1(EMAX,EMIN+.01*EMAX)
      CALL MAPS(Z(1),Z(LZ),EMIN,EMAX,.15,.99,.65,.99)
      CALL TRACE(Z,ZNRGE,LZ)
      CALL SETCH(1.,17.,1,0,1,1)
      WRITE(100,F11)
F11   FORMAT(2X,□ION ENERGY□,30X,□ELECTRON ENERGY□)
      WRITE(100,F12) TIME
F12   FORMAT(□PLOTS IN Z□/□TIME=□,E11.3)
      CALL AMINMX(ZNRGI,1,LZ,1,EMIN,EMAX)
      EMAX=AMAX1(EMAX,EMIN+.01*EMAX)
      CALL MAPS(Z(1),Z(LZ),EMIN,EMAX,.15,.99,.15,.5)
      CALL TRACE(Z,ZNRGI,LZ)

C.... PLOT DENSITIES IN Z*********************
      CALL FRAME
C.... ION DENSITY VS PSI PLOTTED IN LOWER HALF FRAME  ********
      DMAX=1.05*AMAX1(ZDENI(1),PDEN(1))
      DMIN=0.
      CALL MAPS(PSI(1),PSI(LZ),DMIN,DMAX,.15,.99,.15,.5)
      CALL TRACE(PSI,PRDEN,LZ)
C.... COMPARISION OF ELECTRON, ION & DENSITY FROM POTSHAPE CALC...
      CALL MAPS(Z(1),Z(LZ),DMIN,DMAX,.15,.99,.65,.99)
      CALL SETPCH(0,0,1,0,150)
```

```
      CALL TRACEC(IHE,Z,ZDENE,LZ)
      CALL SETPCH(0,0,1,0,165)
      CALL TRACEC(IHI,Z,ZDENI,LZ)
      CALL SETPCH(0,0,1,0,180)
      CALL TRACEC(IHP,Z,PDEN,LZ)
      CALL SETCH(1.,17.,1,0,1,1)
      WRITE(100,F15)
 F15  FORMAT(□ION DENSITY VS PSI□,23X,□COMPARISON OF Z-DEP DENSITIES□)
      WRITE(100,F16) TIME
 F16  FORMAT(□TIME=□,E11.3,25X,□ELECTRONS(E), IONS(I) AND□)
      WRITE(100,F17)
 F17  FORMAT(41X,□FROM CALC OF POTSHAPE(P)□)
      WRITE(100,F18)
 F18  FORMAT(41X,□PLOTS IN Z□)
C.... Z-DEP PLOT OF PHI AND PSI  ******
      CALL FRAME
      CALL MAPS(Z(1),Z(LZ),PHI(1),PHI(LZ),.15,.99,.65,.99)
      CALL TRACE(Z,PHI,LZ)
      CALL MAP(PSI(1),PSI(LZ),PHI(1),PHI(LZ),.15,.99,.65,.99)
      CALL TRACEP(PSI,PHI,LZ)
      CALL SETCH(1.,17.,1,0,1,1)
      WRITE(100,F25)
 F25  FORMAT(□PSI VS Z□,30X,□PHI VS Z AND PSI(POINTS)□)
      WRITE(100,F41) N,TIME,DTIME
      CALL MAPS(Z(1),Z(LZ),PSI(1),PSI(LZ),.15,.99,.15,.5)
      CALL TRACEP(Z,PSIVAC,LZ)
      CALL TRACE(Z,PSI,LZ)

      DO LOOP16 L=1,LZ
      CALL LINE(Z(L),PSI(1),Z(L),PSI(1)+.02*PSI(LZ))
      CALL LINE(Z(1),PSI(L),.01*Z(LZ),PSI(L))
 LOOP16 CONTINUE

C.... PERPENDICULAR AND PARALLEL PRESURE PROFILES  ********
      CALL FRAME
      CALL MAPS(PSI(1),PSI(LZ),PPAR(1),PPAR(LZ),.15,.99,.65,.99)
      CALL TRACE(PSI,PPAR,LZ)
      DMAX=AMAXAF(PPERP,1,LZ)
      CALL MAPS(PSI(1),PSI(LZ),DMAX,PPERP(LZ),.15,.99,.15,.5)
      CALL TRACE(PSI,PPERP,LZ)
      CALL SETCH(1.,17.,1,0,1,1)
      WRITE(100,F26)
 F26  FORMAT(6X,□PPERP VS PSI□,30X,□PPAR VS PSI□)
      CALL SETCH(10.,4.,1,0,1,0)
      WRITE(100,F27) PPERP(1),PPAR(1),BF(1),BV0,B0MIN,BETA,BETAMAX,BRATIO
 F27  FORMAT(□PPERP,PAR=□,2E12.5/
     1 □B0,BV0=□,2E12.5,2X,□B0MIN=□,E12.5,□ GAUSS□/
     2 □BETA,BETAMAX=□,2F7.4,2X,□MIRROR RATIO=□,F7.5)

      WRITE(3,F41) N,TIME,DTIME
      CALL PEEKI(□PDEN(L)□,PDEN,LZ)
      CALL PEEKI(□ZDENE(L)□,ZDENE,LZ)
      CALL PEEKI(□ZDENI(L)□,ZDENI,LZ)
      CALL PEEKI(□ZNRGE(L)□,ZNRGE,LZ)
      CALL PEEKI(□ZNRGI(L)□,ZNRGI,LZ)
      CALL PEEKI(□PRDEN(L)□,PRDEN,LZ)
      CALL PEEKI(□PPAR(L)□,PPAR,LZ)
      CALL PEEKI(□PPERP(L)□,PPERP,LZ)
      CALL PEEKI(□BVAC(L)□,BVAC,LZ)
      CALL PEEKI(□BF(L)□,BF,LZ)
```

```
      CALL PEEK1(¤PSI(L)¤,PSI,LZ)
      CALL PEEK1(¤PSIVAC(L)¤,PSIVAC,LZ)
      CALL PEEK1(¤Z(L)¤,Z,LZ)
      WRITE(3,¤(/¤¤ELECTRON¤¤)¤)
      WRITE(3,F04) ESCUR,ESENK
      WRITE(3,¤(/¤¤ION¤¤)¤)
      WRITE(3,F04) SCUR,SENK
      IF(SCUR.GT.0.00) CALL PEEK2(¤SS(I,J)¤,SS,ITH,JVI)
      CALL PEEK2(¤CCX(I,J)¤,CCX,ITH,JVI)
      CALL PEEK1(¤ES(J)¤,ES,JV)
      IF(KBUG.EQ.0) GO TO JUMP20
      WRITE(3,F41) N,TIME,DTIME
      CALL PEEK2(¤ZBOUNCE(I,J)¤,ZBOUNCE,ITH,JVI)
      CALL PEEK2(¤TAU(I,J)¤,TAU,ITH,JVI)
      CALL PEEK2(¤CCVV(I,J)¤,CCVV,ITH,JVI)
      CALL PEEK2(¤CCVT(I,J)¤,CCVT,ITH,JVI)
      CALL PEEK2(¤CCTT(I,J)¤,CCTT,ITH,JVI)
      CALL PEEK2(¤CCV(I,J)¤,CCV,ITH,JVI)
      CALL PEEK2(¤CCT(I,J)¤,CCT,ITH,JVI)
      CALL PEEK2(¤CC(I,J)¤,CC,ITH,JVI)
      CALL PEEK1(¤ECVV¤,ECVV,JV)
      CALL PEEK1(¤ECV¤,ECV,JV)
      CALL PEEK1(¤EC¤,EC,JV)
      CALL PEEK1(¤EL¤,EL,JV)
      CALL PEEK1(¤ETAU¤,ETAU,JV)

C.... MIDPLANE OUTPUT  *****
 JUMP20 CONTINUE
C.... PLOT ELECTRON DISTRIBUTION AT MIDPLANE******

      CALL FRAME
      CALL AMINMX(FE,1,JV,1,FMIN,FMAX)
      CALL MAPS(0.,VMAX,FMIN,FMAX,.15,.9,.15,.9)
      CALL TRACE(V,FE,JV)
      CALL SETCH(1.,20.,0,0,1,1)
      WRITE(100,F20) TIME
 F20  FORMAT(¤ELECTON DISTRIBUTION AT TIME=¤,E11.3,¤ SECONDS¤)
      CALL SETCH(15.,3.,1,0,1,0)
      WRITE(100,F21) EDEN,EDENL,EENKEV
 F21  FORMAT(¤DENSITY=¤,2E13.5,5X,¤ENERGY=¤,E12.4,¤KEV¤)
C.... 3-D PLOT OF MIDPLANE ION DISTRIBUTION  ********

      PTANG=PLTANG

      DO LOOP08 I=1,ITH  $  DO LOOP08 J=1,JVI
      QEE(J)=V(J)
 LOOP08 VCOS(I,J)=F(I,J)
      IF(PLTANG.LE.180.) GO TO JUMP11
      PTANG=PLTANG-180.

      DO LOOP09 I=1,ITH  $  DO LOOP09 J=1,JVI
      JJ=JVI-J+1
      QEE(J)=V(JJ)
 LOOP09 VCOS(I,J)=F(I,JJ)
 JUMP11 CONTINUE
      CALL FRAME
      CALL PLOT3D(PTANG,QEE,THDEG,VCOS,1.,1.,1.,JVI,ITH,ITH)
C.... CONTOUR PLOT OF ION DISTRIBUTION*****

      DO LOOP10 J=1,JVI
```

```
      QEE(J)=F(ITH,J)
      DO LOOP10 I=1,ITH
LOOP10 FCONT(J,I)=F(I,J)
      DMAX=AMAXAF(QEE,1,JVI)
      DCONT(1)=1.E-10*DMAX
      DCONT(2)=.9999999*DMAX
      CALL FRAME
      CALL MAPS(V(1),V(JVI),THDEG(1),THDEG(ITH),.059,.999,.25,.95)
      CALL RCONTR(-21,DCONT,1,FCONT,JVI,V,1,JVI,1,THDEG,1,ITH,1)
      CALL SETCH(17.,8.,1,0,1,0)
      WRITE(100,F22)
F22   FORMAT(□ION DISTRIBUTION□/□21 CONTOURS EQUALLY SPACED FROM ZERO TO FMAX□)
      WRITE(100,F23) DMAX,TIME
F23   FORMAT(□FMAX=□,E12.4,□ PART/(CM-CM/SEC)3□,5X,□TIME=□,E11.3,□ SEC□)
      WRITE(100,F23A) DEN,ENKEV
F23A FORMAT(□DENSITY=□,E13.5,5X,□ENERGY=□,E12.4,□KEV□)
      DEN2=.5*Z(2)*PRDEN(1)
      DENQ=.5*Z(2)*PRDEN(1)**2

      DO LOOP11 L=2,LZ-1
      DENQ=DENQ+.5*(Z(L+1)-Z(L-1))*PRDEN(L)**2/PSI(L)
LOOP11 DEN2=DEN2+.5*(Z(L+1)-Z(L-1))*PRDEN(L)/PSI(L)
      DENQ=2.*DENQ*BV0/B0
      DEN2=2.*DEN2*BV0/B0
      WRITE(100,F24) DENL,DEN2,DENQ,PQT,PQJ
F24   FORMAT(□LINE DENSITY=□,2E13.5,□PER CM2□,3X,□N SQ=□,E13.5/
     3 □PQT,J=□,2E12.5)
      WRITE (100,F25A) TAUSRC,TAUII,TAUDRAG,TAUMAG
F25A FORMAT(□TAUSRC,II,DRAG,MAG□,4E10.3)
      CALL SETCH(1.,20.5,0,1,3,0)
      CALL CRTBCD(4H'017)
      CALL SETCH(42.,3.,0,1,3,0)
      CALL CRTBCD(4H'066)

      WRITE(3,F29) N,TIME
F29   FORMAT(□1□/□N=□,I5,/□TIME=□,E11.3,□ SECONDS□/)
      WRITE(3,F30) EDEN,EENKEV
F30   FORMAT(///□ELECTRON DEN=□,E13.4/□ENERGY=□,E11.4,□KEV□)
      CALL PEEK1(□FE(J)□,FE,JV)
      WRITE(3,F41) N,TIME,DTIME
F41   FORMAT(□N=□,I5,5X,□TIME,DTIME=□,E16.9,E9.2,,□ SEC□)
      WRITE(3,F24) DENL,DEN2
      WRITE(3,F31) DEN,ENKEV
F31   FORMAT(□ ION DEN=□,E13.4/□ENERGY=□,E11.4,□KEV□//□ION DIST □)

      DO LOOP40 J=1,JVI
LOOP40 F(ITH+1,J)=F(ITH-1,J)
      CALL PEEK2(□F(I,J)□,F,ITH+1,JVI)
      WRITE(3,F41) N,TIME,DTIME
      WRITE(3,F40) POTENT
F40   FORMAT(///□POTENT=□,F13.3,□KEV□)
      CALL PEEK1(□PHI(L)□,PHI,LZ)
      CALL PEEK1(□PSI(L)□,PSI,LZ)
      WRITE(3,F41) N,TIME,DTIME
      WRITE(3,F42)(ILOSS(J),J=1,JVI)
F42   FORMAT(/□ILOSS(J),J=1,JVI□/(30I4))
      WRITE(3,F43)(JLOSS(I),I=1,ITH)
F43   FORMAT(/□JLOSS(I),I=1,ITH□/(30I4))
      CALL PEEKN(□POTSHAPE ITERATIONS KKOUNT(L)□,KKOUNT,LZ)
C.... SUMMARY PAGE TO FICHE AND DD80 FILES
```

```
      CALL FRAME
      CALL SETCH(1.,42.,0,0,1,0,0)
      CALL SUMMARY(100)
      CALL SUMMARY(3)
      CALL PLOTEA
      CALL TIMEOUT
      CALL EMPTY(3)
      RETURN
      END
```

```
      SUBROUTINE ROSEN(L)
      USE BACOMM
      REAL NO
      DIMENSION GSUM(ITH,JVI)
      DIMENSION TG(JVI,(0,MX)),TGV(JVI,(0,MX)),TGVV(JVI,(0,MX))
      EQUIVALENCE (GSUM,CCX)
      DIMENSION GROSS(JVI,(0,MX),3)
      EQUIVALENCE (GROSS,G)
      EQUIVALENCE (TG,G),(TGV,GV),(TGVV,GVV)

      CALL IONPROJ(L)     $$$** OBTAINS ALEG(J,M)
      CALL EMOMENTS(L)    $$$** OBTAINS EQMM(J), ETC. AND FEZ(J) ****
      CALL COULOG(L)
      IF(L.GT.1) GO TO JUMP5
      TAUDRAG=.5*2.15E13*SQRT(EENKEV**3)/(CLOGIE*EDEN)
      CLOGIE0=CLOGIE  $$$ SAVE MIDPLANE COULOMB LOG FOR ELECTRON RATE EQN
      TAUII=ENKEV*AMASS/1.67E-24
      TAUII=2.44E11*SQRT(TAUII)*ENKEV/(CLOGII*ANUMB**4*DEN)
 JUMP5 CONTINUE
      IF(KBUG.LE.1) GO TO JUMP10
      CALL PEEK2(□ALEG□,ALEG,JVI,MX+1)
      CALL PEEK1(□FEZ□,FEZ,JV)
      CALL PEEK1(□EQEE□,EQEE,JV)
      CALL PEEK1(□EQMM□,EQMM,JV)
      CALL PEEK1(□EQNN□,EQNN,JV)
      WRITE(3,F01) L,CLOGEE,CLOGIE,CLOGII
 F01  FORMAT(/□L=□,I3/□CLOGEE,IE,II=□,3E16.8)
 JUMP10 CONTINUE

      DO LOOP20 M=0,MX
      CALL MOMENTS(M)    $$$**  OBTAINS QMM(J), ETC FOR IONS****
      MM=2*M
      IF(KBUG.LE.1) GO TO JUMP20
      WRITE(3,F02) M,MM
 F02  FORMAT(/□M=□,I4,4X,□2*M=□,I4)
      CALL PEEK1(□QEE□,QEE,JVI)
      CALL PEEK1(□QMM□,QMM,JVI)
      CALL PEEK1(□QNN□,QNN,JVI)
      CALL PEEK1(□QRR□,QRR,JVI)
 JUMP20 CONTINUE
      CFACT=4.*PI/(2*MM+1)
      GFACT1=1./(2*MM+3)
      GFACT2=1./(2*MM-1)

      DO LOOP25 J=2,JVI
      TG(J,M)=CFACT*V4(J)*(GFACT1*(QEE(J)+QMM(J))-GFACT2*(QNN(J)+QRR(J)))
      TGV(J,M)=CFACT*V3(J)*(GFACT1*((MM+2)*QMM(J)-(MM+1)*QEE(J))-
     & GFACT2*(MM*QRR(J)-(MM-1)*QNN(J)))
      TGVV(J,M)=CFACT*V2(J)*(GFACT1*((MM+1)*(MM+2))*(QEE(J)+QMM(J))-
     & GFACT2*(MM*(MM-1))*(QNN(J)+QRR(J)))
 LOOP25 CONTINUE
      IF(M.EQ.0) CALL ECOEF(L.    $$$  ECOEF OBTAINS ELECTRON COEFFICIENTS
 LOOP20 CONTINUE
 JUMP25 CONTINUE
C.... HAVE COMPLETE ELECTRON COEFFICIENTS AT Z, AND HAVE ION PART OF
C     G-ROSENBLUTH POTENTIAL STORED IN TG(J,M)*******

      DO LOOP40 J=2,JVI
      EHV=-4.*PI*V(J)*EQNN(J)
      EGV=4.*PI*V3(J)*(THIRD*(2.*EQMM(J)-EQEE(J))+EQNN(J))
```

```
      EGVV=8.*PI*THIRD*V2(J)*(EQEE(J)+EQMM(J))
      CV(J)=GAM3*CLOGIE*ANUMBI2*EHV
      G(J,0)=0.  $$$  NOT NEEDED IN CALC. OF DERIVATVES OF G
      GV(J,0)=CLOGII*TGV(J,0)+CLOGIE*ANUMBI2*EGV
      GVV(J,0)=CLOGII*TGV\ (J,0)+CLOGIE*ANUMBI2*EGVV

      DO LOOP45 M=1,MX
      G(J,M)=CLOGII*TG(J,M)
      GV(J,M)=CLOGII*TGV(J,M)
      GVV(J,M)=CLOGII*TGVV(J,M)
LOOP45 CONTINUE
LOOP40 CONTINUE
JUMP30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SOURCE
C.... EVALUATION AND ORBIT INTEGRAL OF SOURCE
C.... FLAG KSOURCE=0,1 TO FIX OR RESCALE SOURCE BETWEEN EVALUTIONS
C.... SOURCE EVALUATED WHEN N=MULTIPLE OF NCOEF
C.... FLAG KFIXSS=1 CAUSES SOURCE TO BE EVALUATED AT N=0 AND THEN
C.... REMAIN FIXED
C.... SENKEV,SJCOS,SVS,STS,SJCUR,SJI,SJCX,ZBEAM,TSMESH,TAUBEAM ARE
C.... INPUT.
C.... SVS=GAUSSIAN FACTOR.  IF LESS THAN 1.E-4 THEN SVS IS TAKEN TO BE
C.... EFOLDING ENERGY WIDTH IN KEV.
      USE BACOMM

      DATA NRUNS/0/, RAT,DENSAVE/2(1.)/

      IF(N*KFIXSS.GT.0) GO TO JUMP80
      IF(NRUNS.EQ.NRUN) GO TO JUMP20
      NRUNS=NRUN
      CALL GETR(TSMESH,ITHS,RAT)
      SCOS(ITHS)=0.  $  SCOS(1)=COSS(1)
      SCOS(ITHS-1)=TSMESH*COSS(1)/(ITHS-1)

      DO LOOP2 II=2,ITHS-2
      I=ITHS-II
 LOOP2 SCOS(I)=SCOS(I+1)+RAT*(SCOS(I+1)-SCOS(I+2))
      CALL PEEKI(¤SCOS(I)¤,SCOS,ITHS)
      IF(SCOS(2).GE.SCOS(1)) CALL ERR(¤BAD SOURCE GRID¤)
      TCX=TJI=TCUR=0.

      DO LOOP5 NS=1,NSOR
      TCX=TCX+SJCX(NS)  $   TJI=TJI+SJI(NS)
      TCUR=TCUR+SJCUR(NS)
 LOOP5 CONTINUE

      DO LOOP20 NS=1,NSOR

      DO LOOP10 I=1,ITHS
      EXT(I,NS)=EXP(-STS(NS)*(SCOS(I)-SJCOS(NS))**2)
      EXT(I,NS)=EXT(I,NS)+EXP(-STS(NS)*(SCOS(I)+SJCOS(NS))**2)
 LOOP10 CONTINUE
      SUM=EXT(ITHS,NS)*SCOS(ITHS-1)
      DO LOOP11 I=2,ITHS-1
 LOOP11 SUM=SUM+(SCOS(I-1)-SCOS(I+1))*EXT(I,NS)
      IF(SUM.LE.0.00) CALL ERR(¤BAD SOURCE 1¤)
      DO LOOP12 I=1,ITHS
 LOOP12 EXT(I,NS)=EXT(I,NS)/SUM
      SENERGY=SENKEV(NS)*ERGTKEV
      VPEAK=0.  $  IF(SENERGY.GT.0.00) VPEAK=SQRT(2.*SENERGY/AMASS)
      IF(SVS(NS).GT.1.E-4) SVS(NS)=1/(CONV*SVS(NS))
      DO LOOP15 J=1,JVI-1
 LOOP15 EXV(J,NS)=EXP(-SVS(NS)*(V(J)-VPEAK)**2)
      SUM=0.
      DO LOOP16 J=2,JVI-1
 LOOP16 SUM=SUM+DELV(J)*V2(J)*EXV(J,NS)
      SUM=SUM*2.*PI
      IF(SUM.LE.0.00) CALL ERR(¤BAD SOURCE2¤)

      DO LOOP17 J=1,JVI
 LOOP17 EXV(J,NS)=EXV(J,NS)/SUM
 LOOP20 CONTINUE
      TSTART=TIME
```

```
       ZBSTART=ZBEAM
 JUMP20 CONTINUE
       ZBEAM=ZBSTART+(TIME-TSTART)*ZLENGTH/TAUBEAM
       IF(NCOEF*(N/NCOEF).EQ.N) GO TO JUMP21
       IF(KSOURCE.EQ.0) GO TO JUMP80
       SCALE=DEN/DENSAVE
       DENSAVE=DEN
       IF(ABS(SCALE-1.00).GT.0.05) GO TO JUMP21

       DO LOOP21 J=1,JVI  $  DO LOOP21 I=1,ITH
 LOOP21 SS(I,J)=SS(I,J)*SCALE
       GO TO JUMP80

 JUMP21 DENSAVE=DEN

       DO LOOP30 J=1,JVI  $  DO LOOP30 I=1,ITH
       SS(I,J)=CCX(I,J)=0.
 LOOP30 CONTINUE
       IF(TCUR+TJI+ABS(TCX).LE.0.00) GO TO JUMP80  $$$  NO SOURCE
       CALL TIME1(5)
       LB=1

       DO LOOP31 L=1,LZ
       SZ(L)=Z(L) $ SPSI(L)=PSI(L) $ SPH(L)=PHI(L) $ SZDEN(L)=PRDEN(L)
 LOOP31 IF(Z(L).LT.ZBEAM) LB=L
       IF(LB.GT.LZ-2 .OR. LB.LE.1) GO TO JUMP35
       SZ(LB+1)=.001*Z(LB)+.999*ZBEAM $ SZ(LB+2)=.001*Z(LB+2)+.999*ZBEAM
       DPS=PSI(LB+1)-PSI(LB)  $  DPH=PHI(LB+1)-PHI(LB)
       DNZ=PRDEN(LB+1)-PRDEN(LB)
       DZUSE=(SZ(LB+1)-Z(LB))/(Z(LB+1)-Z(LB))
       SPSI(LB+1)=PSI(LB)+DZUSE*DPS  $  SPH(LB+1)=PHI(LB)+DZUSE*DPH
       SZDEN(LB+1)=PRDEN(LB)+DZUSE*DNZ
       DZUSE=(SZ(LB+2)-Z(LB))/(Z(LB+1)-Z(LB))
       SPSI(LB+2)=PSI(LB)+DZUSE*DPS  $  SPH(LB+2)=PHI(LB)+DZUSE*DPH
       SZDEN(LB+1)=PRDEN(LB)+DZUSE*DNZ
 JUMP35 CONTINUE
       CALL IORBIT(SZ,SPSI,SPH)

       DO LOOP70 L=1,LZ
       IF(L.GT.LB+2) GO TO LOOP70
       IF(MIDPLANE.EQ.1 .AND. L.GT.1) GO TO LOOP70
       CALL ORBIT(L)

       DO LOOP70 J=2,JVI-1
       VZ2=V2(J)+OVPOT2(L)
       IF(VZ2.GE.VMAXI**2) GO TO LOOP70
       VZ=SQRTI(VZ2)

       DO LOOP40 JJ=J,JVI
       JZ=JJ
 LOOP40 IF(V(JJ).GT.VZ) GO TO JUMP40
 JUMP40 VRAT=(VZ-V(JZ-1))/(V(JZ)-V(JZ-1))

       DO LOOP70 I=1,ITH
       ZB=ZBOUNCE(I,J)  $  IF(ZB.GE.Z(LZ)) GO TO LOOP70
       IF(ZB.LE.SZ(L)) GO TO LOOP70
       COSZ=VCOS(I,J)/VZ

       DO LOOP45 II=2,ITHS
       IZ=II
```

```
 LOOP45 IF(SCOS(II).LT.COSZ) GO TO JUMP45
 JUMP45 CRAT=(COSZ-SCOS(IZ-1))/(SCOS(IZ)-SCOS(IZ-1))
        SRC=CCEX=0.
        IF(ZBEAM.LT.SZ(L)) GO TO JUMP50

        DO LOOP50 NS=1,NSOR
        E1=EXV(JZ-1,NS)+VRAT*(EXV(JZ,NS)-EXV(JZ-1,NS))
        E2=EXT(IZ-1,NS)+CRAT*(EXT(IZ,NS)-EXT(IZ-1,NS))
        SJ=SJCUR(NS)+(SJI(NS)+SJCX(NS))*SZDEN(L)
        SRC=SRC+SJ*E1*E2
        CCEX=CCEX+SJCX(NS)
 LOOP50 CONTINUE
        SRC=AMAX1(SRC,0.)
 JUMP50 CONTINUE
        SS(I,J)=SS(I,J)+DTAU(I,J)*SRC
        CCX(I,J)=CCX(I,J)+DTAU(I,J)*CCEX
        IF(ZB.GE.SZ(LZ)) GO TO LOOP70
        IF(ZB.GE.SZ(L+1)) GO TO LOOP70
        IF(MIDPLANE.EQ.1) GO TO LOOP70
        SRC=CCEX=0.
        IF(ZB.GT.ZBEAM) GO TO JUMP55

        DO LOOP55 NS=1,NSOR
        E1=EXV(JZ-1,NS)+VRAT*(EXV(JZ,NS)-EXV(JZ-1,NS))
        E2=EXT(ITHS,NS)
        SJ=SJCUR(NS)+(SJI(NS)+SJCX(NS))*SZDEN(L)
        SRC=SRC+SJ*E1*E2
        CCEX=CCEX+SJCX(NS)
 LOOP55 CONTINUE
 JUMP55 CONTINUE
        SS(I,J)=SS(I,J)+EDTAU(I,J)*SRC
        CCX(I,J)=CCX(I,J)+EDTAU(I,J)*CCEX
 LOOP70 CONTINUE

        DO LOOP80 J=1,JVI  $  DO LOOP80 I=1,ITH
        IF(TAU(I,J).LE.0.00) GO TO LOOP80
        SS(I,J)=SS(I,J)/TAU(I,J)
        CCX(I,J)=CCX(I,J)/TAU(I,J)
 LOOP80 CONTINUE
        CALL TIME2(5)
 JUMP80 CONTINUE
C.... ELECTRON SOURCE FROM ION SOURCE  **********
        CALL QVSET(0.,ES,JV)
        DO LOOP81 J=1,JVI  $  DO LOOP81 I=1,ITH
 LOOP81 ES(J)=ES(J)+.5*TINT(I)*SS(I,J)
C.... SOURCE DIAGNOSTICS  *********
        ESENK=ESCUR=0.

        DO LOOP82 J=1,JV
        ESCUR=ESCUR+2*VINT(J)*ES(J)
        ESENK=ESENK+2*VINT(J)*ES(J)*V2(J)
 LOOP82 CONTINUE
        IF(ESCUR.GT.0.00) ESENK=ESENK*.5*EMASS/(ESCUR*ERGTKEV)
        SCXL2=Z(2)*SZDEN(1)*TCX  $  SCURL2=Z(2)*(TCUR+(TJI+TCX)*SZDEN(1))

        DO LOOP85 L=2,LZ-1
        IF(SZ(L).GT.ZBEAM) GO TO LOOP85
        DZUSE=SZ(L+1)-SZ(L-1)
        SCXL2=SCXL2+DZUSE*SZDEN(L)*TCX*PSI(L)
        SCURL2=SCURL2+DZUSE*(TCUR+(TJI+TCX)*SZDEN(L))*PSI(L)
```

```
LOOP85 CONTINUE
       SCX=SCXL=SCUR=SCURL=SENK=0.

       DO LOOP90 J=1,JVI  $  DO LOOP90 I=1,ITH
       SCX=SCX+TINT(I)*VINT(J)*CCX(I,J)*F(I,J)
       SCXL=SCXL+TINL(I)*VINL(J)*TAU(I,J)*F(I,J)*CCX(I,J)
       SCUR=SCUR+TINT(I)*VINT(J)*SS(I,J)
       SENK=SENK+TINT(I)*VINT(J)*SS(I,J)*V2(J)
       SCURL=SCURL+TINL(I)*VINL(J)*TAU(I,J)*SS(I,J)
LOOP90 CONTINUE
       SCXL=SCXL*BV0/B0  $  SCURL=SCURL*BV0/B0
       SCXL2=SCXL2*BV0/B0  $  SCURL2=SCURL2*BV0/B0
       IF(SCUR.GT.0.00) SENK=SENK*.5*AMASS/(SCUR*ERGTKEV)
       TAUSRC=DEN/(ABS(SCUR)+ABS(SCX)+1.E-90)
       IF(KBUG.EQ.0) RETURN
       CALL PEEK2(□SOURCE ZBOUNCE(I,J)□,ZBOUNCE,ITH,JVI)
       RETURN
       END
```

```
      SUBROUTINE SSTEST
C.... TEST FOR STEADY-STATE.  IF KSSPS=1 OR LARGER,
C.... KSSPASS IS INCREMENTED EACH TIME SS TEST IS PASSED
C.... IF N REACHES NSSMAX, KSSPASS IS SET TO 3 TO FORCE NEW INPUT.
C.... KTTY SET TO 1 WHEN SS TEST PASSED TO GET
C.... TTY OUTPUT FROM SR. TTYINT
      USE BACOMM
      DATA NRUNS/0/

      IF(NRUNS.EQ.NRUN) GO TO JUMP10
      NRUNS=NRUN
      KSSPASS=KSS=0
      TSTART=TIME  $  NSTART=N
      NTEST=N+20
 JUMP10 CONTINUE
      TO=TN  $  DO=DN  $  EO=EN  $  DLO=DLN
      TN=TIME  $  DN=DEN  $  EN=ENKEV  $  DLN=DENL
      IF(N.LT.NTEST) GO TO JUMP20
      ESLOPE=AMAX1(ABS(DN-DO)/DN,ABS(EN-EO)/EN)
      IF(MIDPLANE.EQ.0) ESLOPE=AMAX1(ESLOPE,ABS(DLN-DLO)/DLN)
      ESLOPE=ESLOPE*AMAX1(TAUSRC,TAUII,TAUDRAG)/(TN-TO)
      IF(ESLOPE.LT.EPSSS .AND. KSSPS.GT.0) KSSPASS=KSSPASS+1
      IF(N.GE.NSSMAX) KSSPASS=3
      IF(KSS.EQ.KSSPASS) GO TO JUMP20
      KSS=KSSPASS  $  KTTY=1
      IF(KSSPASS.EQ.1) NTEST=N+.1*(TIME-TSTART)/DTIME
      IF(KSSPASS.EQ.2) NTEST=N+20
 JUMP20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SUMMARY(KO)
C.... OUTPUT SUMMARY PAGE TO IO-UNIT SPECIFIED BY KO
      USE BACOMM

      WRITE(KO,FMT1) N,TIME
 FMT1 FORMAT(1H1/¤SUMMARY PAGE¤,5X,¤N,TIME=¤,I5,E16.8,¤SEC¤///)
      WRITE(KO,FMT2) POTENT
 FMT2 FORMAT(¤  DENSITY¤,6X,¤ENERGY¤,10X,¤POTENTIAL=¤,E12.5,¤KEV¤)
      WRITE(KO,FMT3) EDEN,EENKEV
 FMT3 FORMAT(2E12.5,¤ ELECTRONS¤)
      WRITE(KO,FMT4) DEN,ENKEV
 FMT4 FORMAT(2E12.5,¤ IONS¤)
      FRAC=0.  $  IF(DENL2.GT.0.) FRAC=100*(DENL2-DENL)/DENL2
      WRITE(KO,FMT5) DENL,FRAC
 FMT5 FORMAT(¤LINE DENSITY(1D)=¤,E12.5,3X,E10.2,¤% ERROR W 2D¤)
      DENLSQ=Z(2)*PRDEN(1)**2  $  DO LOOP10 L=2,LZ-1
 LOOP10 DENLSQ=DENLSQ+(Z(L+1)-Z(L-1))*PRDEN(L)**2/PSI(L)
      WRITE(KO,FMT6) DENLSQ
 FMT6 FORMAT(¤LINE DENSITY SQUARED=¤,E12.5)

C.... BETA ETC.
      WRITE(KO,FMT6A) BETA,BETAMAX,BF(1),BVAC(1),PSI(LZ)
 FMT6A FORMAT(/¤BETA,MAX=¤,2F9.6/
     1 ¤B0,BV0=¤,2E12.5,¤GAUSS¤/
     2 ¤MIRROR RATIO=¤,E12.5)

C.... SOURCE QUANTITIES
      WRITE(KO,FMT7)
 FMT7 FORMAT(/¤SOURCE DIAGNOSTICS¤)
      WRITE(KO,FMT7A) ZBEAM,Z(LZ)
 FMT7A FORMAT(¤ZBEAM=¤,E12.5,¤CM¤,5X,¤MIRROR LENGTH=¤,E12.5,¤CM¤)
      WRITE(KO,FMT8) SCUR,SCX,SCUR-SCX
 FMT8 FORMAT(¤MIDPLANE CURRENT=¤,E12.5,2X,¤CX=¤,E12.5,¤NET CUR=¤,E12.5)
      FRAC=0.  $  IF(SCURL2.GT.0.) FRAC=100*(SCURL2-SCURL)/SCURL2
      WRITE(KO,FMT9) SCURL,SCURL2,FRAC
 FMT9 FORMAT(¤TOTAL LINE CURRENT=¤,E12.5,¤(2D)¤,1X,E12.5,¤(1D)¤,2X,
     1 E10.2,¤% ERR¤)
      FRAC=0.  $  IF(ABS(SCXL2).GT.0.) FRAC=100*(SCXL2-SCXL)/SCXL2
      WRITE(KO,FMT10) SCXL,SCXL2,FRAC
 FMT10 FORMAT(¤TOTAL LINE CX=¤,E12.5,¤(2D)¤,1X,E12.5,¤(1D)¤,2X,
     1 E10.2,¤% ERR¤)
      EIL=SCURL-SCXL  $  EIL2=SCURL2-SCXL2
      FRAC=0.  $  IF(ABS(EIL2).GT.0.) FRAC=100*(EIL2-EIL)/EIL2
      WRITE(KO,FMT11) EIL,EIL2,FRAC
 FMT11 FORMAT(¤NET LINE CURRENT=¤,E12.5,¤(2D)¤,1X,E12.5,¤(1D)¤,2X,
     1 E10.2,¤% ERR¤)
      ENTAU1=DEN**2/(1.+ABS(SCUR-SCX))
      ENTAU2=DENLSQ/(1.+ABS(EIL))
      WRITE(KO,FMT12) ENTAU1,ENTAU2
 FMT12 FORMAT(/¤NTAU=¤,2E10.2,¤ MIDPLANE,BOUNCE-AVERAGE¤)
      WRITE(KO,FMT13)
 FMT13 FORMAT(/¤REACTOR DESIGN QUANTITIES¤)
      EL0=Z(2)  $  DO LOOP15 L=2,LZ-1
 LOOP15 EL0=EL0+(Z(L+1)-Z(L-1))/PSI(L)
      EL0=EL0+(Z(LZ)-Z(LZ-1))/PSI(LZ)
      EL1=DENL2/PRDEN  $  EL2=DENLSQ/PRDEN**2
      WRITE(KO,FMT15) EL0,EL1,EL2
 FMT15 FORMAT(¤L0,1,2=¤,3E12.5,¤CM¤)
      EL0=EL0/Z(LZ)  $  EL1=EL1/Z(LZ)  $  EL2=EL2/Z(LZ)
      WRITE(KO,FMT16) EL0,EL1,EL2
```

```
FMT16 FORMAT(¤L0,1,2=¤,3F12.6,¤ UNIT S OF ZM¤)
      QMF=0.
      WRITE(KO,FMT20) QMF
FMT20 FORMAT(/¤Q=¤,E12.5)
      RETURN
      END
```

```
      SUBROUTINE TIMEIN(LBIN,NBIN,IOUT)
C.... LBIN STORES BIN LABLE ADDRESS, NBIN IS NUMBER OF BINS
C.... IOUT IS OUTPUT UNIT NUMBER FOR WRITE STATEMENTS
C.... NCALL IS TOTAL NUMBER OF CALLS REFERING TO BIN
C.... T IS TOTAL CPU CHARGE AND TAV IS AVERAGE IN SECONDS AND SEC/CALL
C.... A CALL TO TIME1 STARTS TIMING ON BIN □NB□ WHICH ENDS WITH A
C...  CALL TO TIME2(NB).   TIMEOUT OUTPUTS TIMINGS TO I-O UNIT IOUT
      DIMENSION LBIN(10),NCALL(10),T(10),TAV(10)

      DATA CONV/1.E-6/, I,J,K,L,M/5(0)/

      NOBIN=AMINO(NBIN,10)
      CALL TICHEK(I,J)

      DO LOOP10 NN=1,10
      NCALL(NN)=0
      T(NN)=TAV(NN)=0.
 LOOP10 CONTINUE
      RETURN

      ENTRY TIME1(NB)
      CALL TICHEK(I,J,K,L,M)
      TAV(NB)=K  $$$ STORE CURRENT CPU TIME IN MICROSEC
      RETURN

      ENTRY TIME2(NB)
      CALL TICHEK(I,J,K,L,M)
      T(NB)=T(NB)+(K-TAV(NB))*CONV  $$$ UPDATE CPU CHARGE IN SECONDS
      NCALL(NB)=NCALL(NB)+1  $  TAV(NB)=T(NB)/NCALL(NB)
      RETURN

      ENTRY TIMEOUT
      CALL TICHEK(I,J,K,L,M)  $  TOTALT=CONV*K
      WRITE(IOUT,F01) TOTALT,CONV*L,CONV*M
 F01  FORMAT(//□TOTAL CPU,IO,SYS=□,3E12.5,□ SECONDS□/
     1 □ BIN   LABLE□,4X,□NO. REFFS□,5X,□TOTAL CPU□,6X,□AVG CPU/REFF□,
     2 4X,□CPU ACCOUNTED FOR□)
      S=0.

      DO LOOP30 NN=1,NOBIN
      S=S+T(NN)
      WRITE(IOUT,F02) NN,LBIN(NN),NCALL(NN),T(NN),TAV(NN),100*T(NN)/TOTALT
 F02  FORMAT(1X,I2,3X,A10,1X,I7,5X,E12.5,3X,E12.5,□ SEC□,10X,F6.2,□%□)
 LOOP30 CONTINUE
      WRITE(IOUT,F03) 100*S/TOTALT
 F03  FORMAT(18X,□IF BINS ARE MUTUALLY EXCLUSIVE, TOTAL ACCOUNTED FOR=□,
     1 F6.2,□%□)
      RETURN
      END
```

```
      SUBROUTINE TTYINT
C.... HANDLES TTY INTERACTION
C.... KTTYCON=0,1 IS NOT,IS TTY CONTROLLED
C.... KTTY IS FLAG TO ACTIVATE TTY OUTPUT WHEN KTTYCON=1
      USE BACOMM
      DATA NRUNS,KTTYCON/2(0)/

      IF(NRUNS.EQ.NRUN) GO TO JUMP0
      NRUNS=NRUN
      CALL GOB(2425B,KTTYCON,1,M)  $$$  SEE IF TTY CONTROLLED
      KTTY=0
 JUMP0 CONTINUE
      IF(KTTYCON.EQ.0) GO TO JUMP30
      IF(KCHUG.EQ.0) GO TO JUMP29
      PRINT F00A, N,DEN,DENL,ENKEV,BETA,BETAMAX
 F00A FORMAT(1X,I4,2E12.5,2X,E12.5,¤KEV  ¤,2F7.4)
 JUMP29 CONTINUE
      M=I=0  $  CALL GOB(2001B,I,100B,M)  $$$ GET MESSAGE
      IF(I.EQ.0) KTTY=1  $$$ A MESSAGE WAS WAITING
      IF(KTTY.EQ.0) GO TO JUMP30
      KTTY=0
      IF(M.EQ.6RENDNOW) NSTOP=N
      IF(M.EQ.3REND) NSTOP=NOUT*(1+N/NOUT)
      IF(M.EQ.4RFR80) KEEP=-1
      IF(M.EQ.4RGIVE) KEEP=0
      IF(M.EQ.4RKEEP) KEEP=1
      IF(M.EQ.4RCHUG) KCHUG=1
      IF(M.EQ.6RUNCHUG) KCHUG=0
      IF(M.EQ.4RPASS) KSSPASS=3

      KO=59
      WRITE(KO,F01) N,NTEST,KSSPASS,ESLOPE,TIME-TSTART,DTIME
 F01  FORMAT(/¤N,NTEST=¤,2I5,¤  PASS,SS=¤,I1,E9.2,2X,¤T,DT=¤,2E9.2,¤SEC¤)
      IF(M.EQ.3RHOW) GO TO JUMP30
      WRITE(KO,F02) EDEN,EENKEV,POTENT,TIME
 F02  FORMAT(¤ELEC=¤,2E11.4,¤KEV¤,2X,¤POTENT=¤,E10.3,¤KEV¤,2X,¤TIME=¤,E10.3)
      WRITE(KO,F03) DEN,DENL,ENKEV,BETA,BETAMAX
 F03  FORMAT(¤IONS=¤,3E11.4,¤KEV  BETA,MAX=¤,2F7.4)
      WRITE(KO,F04) TAUSRC,TAUII,TAUDRAG,TAUMAG
 F04  FORMAT(¤TAUSRC,II,DRAG,MAG=¤,4E9.2)
      WRITE(KO,F05) NSTOP,KEEP
 F05  FORMAT(¤NSTOP=¤,I5,2X,¤KEEP=¤,I2)
      IF(KTCON.EQ.0) GO TO JUMP30
      IF(M.EQ.3RIDT) DTSET=2.*DTSET
      IF(M.EQ.3RRDT) DTSET=.5*DTSET
 JUMP30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE UFIELD
C.... UFIELD UPDATES BF, MAGNETIC FIELD
C.... UFIELD CALLS POTSHAPE,UPDATE PHI OF PSI, AND PRESSURE FOR PPERP,PAR
C.... KUF IS INPUT FLAG FOR THIS ROUTINE
C.... KUF=0  NO UPDATE OF FIELD, BUT BVAC IS DIAGNOSED W BETA=BETAMAX
C....     1  BF AND BVAC RELATED VIA LONG-THIN APPROX
 ....     2  BF HAS SAME SHAPE AS BVAC BUT DEPRESSED AT CENTER
      USE BACOMM

      IF(KUF.GT.0) GO TO JUMP20
C   .. DETERMINE OPTIMAL BVAC FOR BETA=BETAMAX
      CALL POTSHAPE
      CALL PRESSURE
      BETA=BETAMAX

      DO LOOP10 L=1,LZ
      BF(L)=B0MIN*PSI(L)
      BVAC(L)=SQRT(BF(L)**2+8*PI*PPERP(L))
 LOOP10 CONTINUE
      GO TO JUMP100
 JUMP20 CONTINUE
C.... DETERMIN BVAC OF B
      CALL PRESS0  $$$ GET PPERP AT MIDPLANE
      BF=BVAC**2-8*PI*PPERP
      IF(BF.LE.0.) CALL ERR(¤FIELD REVERSAL??¤)
      BF=B0=SQRT(BF)
      BRATIO=BVAC*BRVAC/BF
      SCALE=(BRATIO-1)/(PSI(LZ)-1)

      DO LOOP20 L=1,LZ
      PSI1(L)=PSI1(L)*SCALE
      PSI(L)=PSI1(L)+1
      PSIH(L)=SQRT(PSI(L))
      BF(L)=BF*PSI(L)
 LOOP20 CONTINUE
      CALL POTSHAPE
      CALL PRESSURE
      BETA=PPERP/(PPERP+BF**2/(8*PI))
      IF(KUF.EQ.2) GO TO JUMP21
      IF(BETA.LE.0.1) GO TO JUMP21
      IF(BETAMAX-BETA.GT.DBSTOP*BETAMAX) GO TO JUMP21
      NSTOP=N
      WRITE(3,F00)  $$F00 FORMAT(¤BETA TOO CLOSE TO BETAMAX¤)
      CALL SETCH(10.,32.,0,0,2,0,0)
      WRITE(100,F00)
      IF(KTTY.EQ.1) PRINT F00
 JUMP21 CONTINUE

      DO LOOP25 L=2,LZ-1
      BVAC(L)=SQRT( BF(L)**2+8*PI*PPERP(L) )
      IF(KUF.LE.1) GO TO JUMP24
C.... FORCE BVAC AND BF TO HAVE SAME SHAPE
      BVAC(L)=(BF(L)-BF(1))/(BF(LZ)-BF(1))*(BRVAC-1)
      BVAC(L)=BVAC(1)*(1+BVAC(L))
 JUMP24 CONTINUE
      Z(L)=Z(L-1)
      IF(BVAC(L).LE.BVAC) GO TO LOOP25
      Z(L)=ZOFBV(BVAC(L))
 LOOP25 CONTINUE
C.... TEST FOR MIRROR MODE STABILITY
```

```
      DO LOOP30 L=2,LZ
LOOP30 IF(BVAC(L-1).GE.BVAC(L)) CALL ERR(¤MIRROR MODE UNSTABLE UFIELD¤)
      CALL BTERMS
JUMP100 CONTINUE
      DO LOOP100 L=1,LZ
LOOP100 PSIVAC(L)=BVAC(L)/BF
      RETURN
      END
```

```
      SUBROUTINE ZGRID
C.... KZGRID=0 FOR PARABOLIC WELL
C....        1 FOR LINEAR WELL, USUALLY FOR DEBUG PURPOSES
      USE BACOMM

      EVEN=(BRATIO-1.)/(LZ-1)
      IF(PMESH.LE.0.00) PMESH=.67*(COSS(ITH-1)/SINN(ITH-1))**2/EVEN
      PMESH=AMIN1(PMESH,1.00)
      RAT=1.  $  CALL GETR(PMESH,LZ,RAT)
      PSI1(1)=0.  $  PSI1(2)=EVEN*PMESH  $  PSI1(LZ)=BRATIO-1.
      DO LOOP10 L=3,LZ-1
 LOOP10 PSI1(L)=PSI1(L-1)+RAT*(PSI1(L-1)-PSI1(L-2))
      IF(PSI1(LZ-1).GE.PSI1(LZ)) CALL ERR(□BAD GRID ZGRID1□)
      Z(1)=0.  $  Z(LZ)=ZLENGTH
      IF(KZGRID.GT.0) GO TO JUMP20
C.... KZGRID ZERO CAUSES PARABOLIC WELL TO BE CONSTRUCTED  ****
      DO LOOP15 L=2,LZ-1
 LOOP15 Z(L)=Z(LZ)*SQRT(PSI1(L)/PSI1(LZ))
      GO TO JUMP50
 JUMP20 CONTINUE
C.... CONSTRUCT LINEAR MAGNETIC WELL  ********
      DO LOOP22 L=2,LZ-1
 LOOP22 Z(L)=Z(LZ)*PSI1(L)/PSI1(LZ)
 JUMP50 CONTINUE
      PSI(1)=PSIH(1)=1.00  $  PSI(LZ)=BRVAC=BRATIO  $  PSIH(LZ)=SQRT(BRATIO)

      DO LOOP50 L=2,LZ-1
      PSI(L)=PSI1(L)+1.00
      PSIH(L)=SQRT(PSI(L))
 LOOP50 CONTINUE

      DO LOOP55 L=1,LZ-1
      DZ(L)=Z(L+1)-Z(L)
      IF(DZ(L).LE.0.00) CALL ERR(□BAD ZGRID3□)
 LOOP55 CONTINUE
      DZ(LZ)=DZ(LZ-1)  $  B0=BV0

      DO LOOP60 L=1,LZ
      BVAC(L)=BF(L)=BV0*PSI(L)
 LOOP60 PSIVAC(L)=PSI(L)
      RETURN
      END
```

```
      FUNCTION ZOFBV(BV)
C.... GET Z FROM VACUUM FIELD, BV
C.... BVAC ASSUMED QUADRATIC IN Z BELOW
      USE BACOMM

      ZOFBV=Z(LZ)*SQRT( (BV-BVAC)/(BVAC(LZ)-BVAC) )
      RETURN
      END
```