EQUATIONS AND ASSOCIATED DEFINITIONS USED IN ONETWO

H.ST.JOHN

At this time this document is incomplete and may have errors in content,typos,etc. It is a preliminary working version only ! This document was last edited on November 29, 2005. This document describes features of the code current with The May '96 version of Onetwo. Some items described here may not be available in earlier versions. This file is called source-o12.* where the extension is ps forX1 the postscript file,tex for the latex file and dvi for the dvi file. If you use tex you may get a copy of the tex file and hack out parts that you might want to include in your own documents.

1 Brief Description

The Onetwo transport code solves a user selectable,arbitrary combination,of flux surface average parabolic equations representing the radial diffusion and convection of 1 or 2 thermal ion species and up to 2 neutral species,the electron energy,the ion energy,the poloidal magnetic field,and the toroidal momentum. The magnetic geometry may be self consistently maintained by solving the elliptic mhd equilibrium equation simultaneously with the diffusion equations. Both free and fixed plasma boundary equilibrium models are supported. Auxiliary heating models include Monte Carlo neutral beam deposition and electron and ion cyclotron and fast wave r.f. heating. Models for ohmic,bootstrap, beam and r.f. driven currents are included. Energy confinement may be simulated by selecting theoretical or empirical electron and/or ion thermal conductivity models,including Rebut-Lallia,Shay,and Waltz- Dominguez. A unique feature of the code is its ability to substitute measured profiles for any of the dependent variables and solving the diffusion equations only for the remaining unknown profiles. A combination of Crank-Nicholson,predictor and iterated corrector methods are used to converge the non linear diffusion equations. The elliptic equations are solved using variable finite difference schemes with single cyclic reduction.

In the next section the actual equations solved by Onetwo are given explicitly. The equations are similar to the original ones presented in GA-

A16178. However a new equation describing the evolution of the toroidal momentum has ben added as have bootstrap models and other source terms. Thus the original document is badly out of date. It is my intent, through online documentation, to fill in the gaps which have developed by making available "ONETWO PAGES" to the users. The equations appearing below have terms separated in a manner which makes them suitable for the matrix finite difference representation used in the code,to facilitate comparison with actual code fragments. Perhaps more importantly, the terms are given in such a manner that they can be associated with actual output quantities produced by the code. A consequence is that simple rearrangement of terms may be necessary in order to see the physical conservation process being described. Another consequence is that I use mixed math/fortran style in some places. This is justifiable since the main purpose of this document is to connect the output of Onetwo with the definitions presented here. The equations in Onetwo were not cast in non dimensional form so we specify the units of input and output terms as appropriate. The dependent variables which are either evolved in time (simulation mode) or are known a priori (analysis mode) are the ion densities, $\boldsymbol{n_i(\frac{1}{cm^3})}$, the electron temperature $\boldsymbol{T_e(kev)}$,the ion temperature $\boldsymbol{T(kev)}$,the poloidal magnetic field taken as the compound quantity $\boldsymbol{FGH\rho B_{p0}(Gauss-cm)}$ and the angular rotation speed $\boldsymbol{\omega(\frac{1}{sec})}$. In order to write some equations more concisely we use the notation $\boldsymbol{u^i}$ to refer to these dependent variables with the understanding that i=1 is a collective index that ranges over all ion densities,i=2 implies $\boldsymbol{T_e}$,i=3 implies $\boldsymbol{T}$, i=4 implies $\boldsymbol{FGH\rho B_{P0}}$,and i=5 implies $\boldsymbol{\omega}$.

2 Transport Equations

As will be seen below the grad rho factors that enter into the equations used in Onetwo (as a result of flux surface averaging the two dimensional transport equations) are treated in a particular way in the code. Thus one has to be careful in defining the fluxes and diffusion coefficients that are to be used. A detailed discussion is given in appendix A.

2.1 Particle Balance Equations

The equation that governs the evolution of the density of primary ion species i is

$$\frac{\partial n_i}{\partial t} + \frac{1}{H\rho}\frac{\partial}{\partial\rho}(H\rho\Gamma_i) = S_i + S_i^{2D} \tag{1}$$

The second term in this and other equations appearing below represents the azimuthally symmetric flux surface averaged divergence of the flux. Here $\boldsymbol{\Gamma_i}$ is the particle flux ($\frac{\#}{cm^2 sec}$) of ion species i. In simulation mode a pinch term can be added to this flux. The source terms appearing on the rhs of this and subsequent equations below are explained in section 4.

2.2 Electron Energy Equation

The equation for describing the evolution of the electron thermal energy is

$$\begin{aligned}\frac{3}{2}\left(T_e\sum_{i=1}^{nion}(n_i\frac{\partial Z_i}{\partial T_e}) + n_e\right)\frac{\partial T_e}{\partial t} + \frac{3}{2}T_e\sum_{i=1}^{nion} Z_i\frac{\partial n_i}{\partial t} \\ + \frac{1}{H\rho}\frac{\partial}{\partial\rho}\left(H\rho(q_e + \frac{5}{2}\Gamma_e T_e)\right) = Q_e - \omega L_e + S_{T_e}^{2D}\end{aligned} \tag{2}$$

The source term $\boldsymbol{\omega L_e}$ represents the (beam) energy that goes into spinning up the electron fluid. This energy is consequently not available for heating the electron distribution. The rotational kinetic energy of the electrons is assumed to be given to the ions (see below, this is a crude first approximation).

2.3 Ion Energy Equation

The equation for describing the evolution of the ion thermal and rotational kinetic energy is

$$\sum_{i=1}^{nion}\left\{\frac{3}{2}n_i\frac{\partial T}{\partial t}+\frac{\partial n_i}{\partial t}\left(\frac{3}{2}T+\frac{1}{2}m_i\omega^2 < R^2 >\right)\right\}+\sum_{i=1}^{nion} m_i n_i\omega < R^2 > \frac{\partial \omega}{\partial t}$$
$$+\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left\{H\rho\left(\sum_{i=1}^{nion}(q_i+\frac{5}{2}\Gamma_i T)+\Gamma_T^{\omega}+\Pi\omega\right)\right\}$$
$$= Q + S_T^{\omega} + S_T^{2D} + S_T^{2D\omega} \quad (3)$$

All ion species are assumed to have a common temperature T. The flux surface average kinetic energy per unit volume in rotation is

$$\sum_{i=1}^{nion}\frac{1}{2}m_i n_i\omega^2 < R^2 > \quad (4)$$

The kinetic energy is assumed to diffuse with the primary ions so we have a flux

$$\Gamma_T^{\omega}=\sum_{i=1}^{nion}\frac{1}{2}m_i < R^2 > \omega^2\Gamma_i \quad (5)$$

and a term associated with viscous heating of the plasma $\Pi\omega$ described further below.

2.4 Faraday's Law

The evolution of the poloidal B field is given by Faraday's Law. In Onetwo this equation takes the form

$$\frac{1}{FG(H\rho)^2\alpha}\frac{\partial(FGH\rho B_{p0})}{\partial t}-\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left(H\rho(d_{4,1}\frac{\partial n_i}{\partial\rho}+d_{4,2}\frac{\partial T_e}{\partial\rho}+d_{4,3}\frac{\partial T}{\partial\rho}\right.$$
$$\left.+d_{4,4}\frac{\partial FGH\rho B_{p0}}{\partial\rho})\right)-\frac{1}{H\rho}\frac{\partial(dB_{p0})}{\partial\rho}=-\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left(\eta_{\parallel}cH < \vec{J_{aux}}\cdot\frac{\vec{B}}{B_{t0}} > \right)$$
$$+\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left(H\rho(D_f^e+D_f^b)\frac{\partial n_f}{\partial\rho}\right)+\frac{B_{p0}}{H\rho}\frac{\partial}{\partial t}\left(lnFGH\rho\right)-\frac{B_{p0}}{H\rho}\frac{\partial d}{\partial\rho} \quad (6)$$

where the terms in the gradient of $n_i, T_e, T,$ and D_f^e, D_f^b are related to the bootstrap current ,B_{p0} is the poloidal magnetic field defined as $B_{p0} =$

$\frac{1}{R_0}\frac{\partial \psi}{\partial \rho}$. d is the speed of a flux surface moving relative to the magnetic axis. For non evolving equilibria we have $d \equiv 0$ and the profiles F,G,H are time independent. The factor α appearing in this equation is used to speed up evolvement toward the steady state in certain instances ($\alpha$is set to a large number when the input switch iffar=1 is set in inone). The bootstrap selection is done with the $d_{4,j}$ parameters . A number of different bootstrap models are available in Onetwo. Each model consists of a subroutine which provides the fourth row of the matrix d as indicated above,plus an explicit source term for fast ions (since,at present,a fast ion equation is not part of our coupled set of diffusion equations). The total bootstrap current density takes the particular form

$$\left\langle \frac{\vec{J_{boot}} \cdot \vec{B}}{B_{T0}} \right\rangle = -\frac{\rho}{c\eta_{\|}} \left(\sum_{i=1}^{3} d_{4,i} \frac{\partial u^i}{\partial \rho} + (D_f^e + D_f^b) \frac{\partial n_f}{\partial \rho} \right) \tag{7}$$

The terms in D_f^e and D_f^b represents and ad-hock approximation to account for the fast ions and may be included or excluded as the user desires.

The total parallel current density consists of ohmic,bootstrap and auxiliary contributions:

$$\left\langle \frac{\vec{J_t} \cdot \vec{B}}{B_{T0}} \right\rangle = \left\langle \frac{\vec{J_{ohm}} \cdot \vec{B}}{\vec{B_{T0}}} \right\rangle + \left\langle \frac{\vec{J_{aux}} \cdot \vec{B}}{\vec{B_{T0}}} \right\rangle + \left\langle \frac{\vec{J_{boot}} \cdot \vec{B}}{\vec{B_{T0}}} \right\rangle \tag{8}$$

Here $\vec{J_{aux}}$ is the current driven by neutral beam injection and rf current drive,$\vec{J_{boot}}$ is the bootstrap current,and the total magnetic field is $\vec{B} = \vec{B_T} + \vec{B_P}$ (ie the sum of toroidal and poloidal components). B_{T0} is a reference magnetic field in the absence of any plasma current at the particular location $R_0 = 169.5\ cm$ (for DIIID,must be defined consistently with $f(\psi)$ see mhd section below).

The parallel component of Ampere's Law is given by

$$\left\langle \frac{\vec{J_t} \cdot \vec{B}}{B_{T0}} \right\rangle = \frac{\rho}{c\eta_{\|}} d_{4,4} \frac{\partial (FGH\rho B_{p0})}{\partial \rho} \tag{9}$$

where the "diffusion" coefficient is

$$d_{4,4} = \frac{c^2 \eta_{\|}}{4\pi F^2 H \rho^2} \tag{10}$$

The form of Eq.[228] is a consequence of the assumed relationship between the parallel inductive electric field and the ohmic current:

$$E_0 \equiv H \left\langle \frac{\vec{E} \cdot \vec{B}}{B_{T_0}} \right\rangle = \eta_{\|} H \left\langle \frac{\vec{J_{ohm}} \cdot \vec{B}}{B_{T_0}} \right\rangle \tag{11}$$

$\eta_{\parallel}$ is the parallel resistivity (corrected for trapped particles) and $\vec{J_{ohm}}$ is the inductively driven current. At any time t the loop voltage calculated by the code is given in terms of this electric field as $V_{loop} = 2\pi R_0 E_0(\rho = a)$ where a is the plasma radius. Combining these results we find that the equation for the parallel ohmic electric field is given by

$$-\frac{cE_0}{H\rho} = -\left(\sum_{i=1}^{4} d_{4,i}\frac{\partial u^i}{\partial \rho} + \left(D_f^e + D_f^b\right)\frac{\partial n_f}{\partial \rho}\right) + \frac{\eta_{\parallel} c}{\rho}\left\langle \frac{\vec{J_{aux}}\cdot \vec{B}}{B_{T0}} \right\rangle \quad (12)$$

Note that when i=4 in the above equation we are in fact referring to the total current. Hence the rhs of Eq.[12] is just the ohmic current expressed using Eqs.[7,8,9].

The Grad-Shafranov equation is flux surface averaged to yield an expression for the toroidal current density J_ϕ:

$$\frac{1}{H\rho}\frac{\partial}{\partial \rho}\left(\frac{u^4}{F}\right) = \frac{4\pi}{c}\left\langle \frac{J_\phi R_0}{R} \right\rangle \quad (13)$$

$$= -\frac{1}{B_{P0}}\left(4\pi\frac{\partial P}{\partial \rho} + \frac{1}{2}\left\langle \frac{1}{R^2} \right\rangle \frac{\partial f^2}{\partial \rho}\right) \quad (14)$$

The calculations are done as follows. At any time t the dependent variable $u^4 \equiv FGH\rho B_{P0}$ is known either from the initial condition or from having solved (possibly some subset of) the set of diffusion equations. The total parallel current density,Eq.[9],is thus determined. The bootstrap current is evaluated using Eq.[7] and independent models are used to determine the auxiliary beam and rf driven currents. We are thus able to solve Eq.[8] for the ohmic term. The original version of Onetwo substitutes $\left\langle \frac{J_\phi R_0}{R} \right\rangle$ for $\left\langle \frac{\vec{J_t}\cdot\vec{B}}{B_{T0}} \right\rangle$ in this calculation. The relationship between these quantities is

$$\left\langle \frac{J_\phi R_0}{R} \right\rangle = F\left\langle \frac{\vec{J_t}\cdot \vec{B}}{B_{T0}} \right\rangle - \frac{c}{4\pi}\frac{GB_{p0}}{F}\frac{\partial F}{\partial \rho} \quad (15)$$

and hence the replacement is equivalent to assuming that $F \equiv 1$. The new version of Onetwo allows the user to eliminate this approximation!

2.4.1 Faraday's Law In Tdem Mode

The time dependent eqdsk mode of operation (see section 3.5 for a description), introduces a new way of treating Faraday's Law in Onetwo. By writing

Eq.[228] in the form

$$c\frac{\partial}{\partial\rho}\left(H\eta_{\parallel}\left\langle\frac{\vec{J_{Ohmic}}\cdot\vec{B}}{B_{T0}}\right\rangle\right)=\frac{1}{FGH\rho}\frac{\partial(FGH\rho B_{P0})}{\partial t}-\frac{\partial}{\partial\rho}dB_{P0}$$
$$-B_{P0}\frac{\partial}{\partial t}(\ln FGH\rho)+B_{P0}\frac{\partial d}{\partial\rho} \quad (16)$$

we obtain a simple differential equation for the product of the parallel resistivity and the ohmic current. Internal definititons used in Onetwo which are relevant to Eq[16] are

$$-B_{P0}\frac{\partial}{\partial t}(\ln FGH\rho)=-H\rho\,\mathit{fday2d1} \quad (17)$$
$$B_{P0}\frac{\partial d}{\partial\rho}=-H\rho\,\mathit{fday2d2} \quad (18)$$
$$-\frac{\partial}{\partial\rho}dB_{P0}=-H\rho\,\mathit{fday2d3} \quad (19)$$
$$\quad (20)$$

Note that Eq[16] can be reduced to the simpler form

$$\frac{\partial}{\partial t}\Psi=HR_0\eta_{\parallel}\left\langle\frac{\vec{J_{Ohmic}}\cdot\vec{B}}{B_{T0}}\right\rangle \quad (21)$$

using the relationship

$$\left.\frac{\partial\Psi}{\partial t}\right|_{\rho}=\left.\frac{\partial\Psi}{\partial t}\right|_{\zeta}-\left(\frac{\zeta}{\rho_a}\right)\left(\left.\frac{\partial\Psi}{\partial\zeta}\right|_{t}\right)\left(\left.\frac{\partial\rho_a}{\partial t}\right|_{\zeta}\right) \quad (22)$$

and the definitions

$$B_{P0}\equiv\frac{1}{R_0}\left.\frac{\partial\Psi}{\partial\rho}\right|_{t} \quad (23)$$
$$d\equiv-\left.\frac{\partial\zeta}{\partial t}\right|_{\rho}\left.\frac{\partial\rho}{\partial\zeta}\right|_{t}=\left.\frac{\partial\rho}{\partial t}\right|_{\zeta} \quad (24)$$
$$\zeta\equiv\frac{\rho}{\rho_{a(t)}} \quad (25)$$

In tdem mode the space and time dependent behavior of the rhs of Eq.[16] is approximated a priori using time dependent solutions of the Grad-Shafranov equation. The constant of integration in Eq.[16] is related to the plasma surface voltage, $V_s\equiv2\pi\frac{\partial}{\partial t}\Psi_{lim}$,

$$\frac{V_s}{2\pi R_0}=\eta_{parallel}H\left.\left\langle\frac{\vec{J_{Ohmic}}\cdot\vec{B}}{B_{T0}}\right\rangle\right|_{\rho_a} \quad (26)$$

It is thus possible to determine the resistivity and/or the non inductive current. In the latter case we can determine either the bootstrap or the auxiliary current provided we are willing to approximate the other current profiles with an appropriate model.

2.5 Toroidal Momentum Equation

The equation for toroidal momentum and rotation used in Onetwo assumes that all of the momentum and energy is carried by the ions. All ions have the same temperature and rotation speed, the associated momentum of each ion fluid depends on the mass of the ion however. The actual equation solved by Onetwo is

$$\sum_{i=1}^{nprim} m_i n_i < R^2 > \frac{\partial \omega}{\partial t} + \omega \sum_{i=1}^{nprim} m_i < R^2 > \frac{\partial n_i}{\partial t} + \frac{1}{H\rho}\frac{\partial}{\partial \rho}\left(H\rho\Gamma_\omega\right) = S_\omega + S_\omega^{2D} \quad (27)$$

The flux surface average toroidal angular momentum density for ion species i is given by

$$m_i n_i \omega < R^2 > \quad (28)$$

The total momentum flux($\frac{g}{sec^2}$) is made up of two parts as usual

$$\Gamma_\omega = \Gamma_\omega^{cond} + \Gamma_\omega^{conv} \quad (29)$$

The "convective" flux is

$$\Gamma_\omega^{conv} = \sum_{i=1}^{nprim} m_i < R^2 > \omega \Gamma_i \quad (30)$$

The "conductive" momentum flux is

$$\Gamma_\omega^{cond} \equiv \Pi = \sum_{k=1} \Pi_k \quad (31)$$

With each Π_k defined as

$$\Pi_k = -d_k(\rho)\frac{\partial}{\partial \rho} u_k \quad (32)$$

At the present time $d_k(\frac{gcm}{sec}$ is assumed to be diagonal with a contribution only form $k = \omega$ Several simple models for d_ω exist in the code.

All of the above equations can be run in analysis or simulation mode. Analysis mode means that the appropriate dependent variable is known a priori as a function of space (and possibly time). Consequently the associated diffusion equation can be inverted to yield information on the transport coefficients. The success of this inverse method depends on how well the sources are known. For example if the momentum equation is run in analysis mode, Eq.[229] is first solved for $\boldsymbol{\Gamma_\omega}$ (assuming all other terms in the equation are known a priori):

$$\boldsymbol{\Gamma_\omega = \frac{1}{H\rho}\int_0^\rho H\rho\Big(S_\omega + S_\omega^{2D} - \omega\sum_{i=1}^{nprim} m_i < R^2 > \frac{\partial n_i}{\partial t} - \frac{\partial \omega}{\partial t}\sum_{i=1}^{nprim} m_i n_i < R^2 > \Big)d\rho} \tag{33}$$

and then $\boldsymbol{\Gamma_\omega^{cond}}$ is obtained from Eq.[29] (the convective part is known at this point from particle balance,Eq.[220]). The diffusion coefficient can then be obtained using Eq.[32].

The equations that are run in simulation must have transport coefficients and appropriate initial and boundary conditions specified a priori. The profiles are then evolved from the initial condition profile as usual.

2.6 Neutral Density Equation

The neutral density in Onetwo is determined by instantaneous global rate balance conditions. Given the electron,and fast and thermal ion densities we can calculate the volume source of neutrals from the recombination and beam neutral charge exchange processes. Using this volumetric source we can determine the resulting neutral density by solving Eq.[**??**] below. This neutral density then allows us to determine the total, volume integrated,rate at which neutrals are consumed due to electron impact ionization. At any given time the total volume integrated, neutral loss rate must equal the source rate. This determines the neutral density due to volumetric sources and sinks (subroutine neuden carries out these calculations).

To complete the problem we must also consider wall sources of neutrals. In analysis mode additional information,in the form of the particle confinement time,***taupin***,is required. Given ***taupin*** we can determine the total rate of ions leaving the plasma through outward diffusion (by the definition of particle confinement time):

$$\boldsymbol{\Gamma_i * S_A = \frac{\int n_i dv}{taupin}} \tag{34}$$

The inward neutral flux due to this ion species is the above value plus any additional specified gas puff. This inward flux of neutrals yields,by way of Eq.[**??**] an associated density in the plasma. Again, this neutral density has a volume integrated loss rate due to electron impact ionization which must be balanced with the global source rate and hence the neutral density due to wall sources is determined. This part of the neutral density is thus inversely proportional to ***taupin***. The total neutral density is just the sum of the wall and volume source problems (Eq.[**??**] is linear).

Onetwo may determine fast ion charge exchange losses that are too large when ***taupin*** is set to a value that is too small (eg much less than the usual 200-400 ms) which results in a large neutral density. Because ***taupin*** controls the neutral density and the neutral density in turn controls the fast ion density a small value of ***taupin*** is often used to enforce charge balance. Particularly in high performance,low density cases, the fast ion density may locally (near the magnetic axis) become greater than or equal to the measured electron density. Decreasing ***taupin*** would decrease this fast ion density so that charge balance could be ensured. However a large fast ion charge exchange loss is thus incurred.

Assuming that ***taupin*** is set to approximately the correct value and assuming that the measured input profiles of $\boldsymbol{n_e}$ and $\boldsymbol{z_{eff}}$ are reasonably accurate, one possible explanation of this local lack of charge balance is an incorrect (ie too peaked) fast ion density profile. Improper prompt orbit averaging and possibly (lack of) fast ion diffusion are thus likely to be the cause . An option in Onetwo allows spreading of the fast ion density to achieve charge balance with the specified $\boldsymbol{zeff}$ and electron density profiles(see the input switch hdepsmth). This is only a temporary stop-gap measure until more appropriate action can be taken.

NOTE: As discussed above ***taupin*** controls the neutral density from (part of) the wall source. It may happen that the neutral density due to the volume source of neutrals is much larger (high density cases) so that ***taupin*** becomes irrelevant.

The neutral density is determined by the Boltzmann equation specialized to circular cylindrical geometry,as developed by Burrell[[1]]. Put neutral density equation and defns here!

3 MHD

The flux surface average geometry dependent factors appearing in the above equations are defined as

$$F = R_0 B_{t0}/f(\psi(\rho)), \qquad G =< (\nabla\rho)^2 R_0^2/R^2 >, \qquad H = F/ < R_0^2/R^2 >$$

where the angle brackets denote the usual flux surface averaging, $R_0$is the major radius at which the vacuum magnetic toroidal field is given as B_{t0}, R is the major radius coordinate to a a point on a flux surface (in the output of Onetwo these quantities are labeled fcap,gcap and hcap respectively). The independent radial coordinate used in the diffusion equations is defined as

$$\rho = \sqrt{\frac{\Phi}{\pi B_{t0}}} \tag{35}$$

Where Φ is the toroidal flux inside a given flux surface.

There are currently three methods of running mhd calculations in Onetwo: single cyclic reduction (used primarily in the free boundary equilibrium cases), successive over relaxation (used primarily in the fixed boundary cases), and the Green's function solution (which is too slow for routine usage but provides a check on other methods). Two other methods, the time dependent eqdsk method, and the static equilibrium method, do not solve the Grad-Shafranov equation at all.

3.1 CYCRED

CYCRED refers to the single cyclic reduction scheme used in Onetwo to solve the Grad-Shafranov equation. I designed the cycred subroutine especially with the idea that FACR(l) method (ie Fourier Analysis combined with cyclic reduction to level l, which is the fastest known method if the grid is fine enough) would eventually be implemented. The method is usually used to determine the contribution to the poloidal flux due to the plasma current only. Contributions from external coils are added in using the Greens function method. Note that using single (as opposed to double) cyclic reduction eliminates the restriction on the grid size in the non reduced dimension (in this case R). Hence 90 by 129 eqdsk have been generated on occasion. Additionally the solution is somewhat faster than in the double cyclic reduction case. This is due to the fact that after a reduction in the Z direction the resulting set of equations is tridiagonal and can be solved very efficiently with a standard (pivoting) tridiagonal solver.

3.2 SORPICRD

SORPICRD refers to successive over relaxation and Picard iteration. I developed this method to handle cases where the plasma boundary is known and fixed for all times. The method uses variable finite difference expressions to accommodate the plasma boundary. We may add the faster (algebraic) multi-grid method at some point in the future since subroutines are now available. Inverse methods could also be used advantageously here.

3.3 GREEN

GREEN refers to the Greens function solution method. A fundamental solution of the Grad-Shafranov equation is obtained by solving

$$\nabla^* G = \frac{\delta(R-R_0)\delta(Z-Z_0)}{2\Pi R} \tag{36}$$

$$G(R=0,Z)=0, \quad G(R,Z)@>> R,Z \Rightarrow \infty > 0$$

The substitution $G = RA(R,Z)$ yields the equation

$$\overbrace{\frac{\partial^2 A}{\partial R^2} + \frac{1}{R}\frac{\partial A}{\partial R} - \frac{A}{R^2}}^{B_1} + \frac{\partial^2 A}{\partial Z^2} = \frac{\delta(R-R_0)\delta(Z-Z_0)}{2\Pi R^2}$$

Here B_1 is Bessel's operator which can be Hankel transformed $(R \Rightarrow \zeta)$. A subsequent Fourier transform $(Z \Rightarrow s)$ leads to the expression

$$A(\zeta, s) = -\frac{J_1(R_0\zeta)}{R_0(2\pi)^{\frac{3}{2}}}\frac{e^{isZ_0}}{(s^2+\zeta^2)}$$

Inversion of the Fourier and Hankel transforms then yields the fundamental solution

$$G^* = \frac{1}{\pi}\sqrt{\frac{R}{R_0}}\frac{1}{k}\left\{E(k) - (1-\frac{1}{2}k^2)K(k)\right\} \tag{37}$$

$$x = \frac{(Z-Z_0)^2 + R_0^2 + R^2}{2R_0 R}, \qquad k^2 = \frac{2}{x+1}, \qquad G^* = 2\pi R_0 G$$

Here $E(k)$ and $K(k)$ are the incomplete elliptic functions which are evaluated numerically using the rational fits in Abramowitz[[?]]. For any toroidal current density the poloidal flux (divided by 2π) is then found by evaluating the integral

$$\Psi(R,Z) = -\mu_0 \int R_0 J_\phi(R_0,Z_0) G^*(R,Z;R_0,Z_0) dR_0 dZ_0 \tag{38}$$

The finite cross sectional area of each toroidal field coil can not be neglected. Consequently the current density $\boldsymbol{J_\phi}$ is modeled as a sum of filaments,each with its own $\boldsymbol{(R_0, Z_0)}$. For any given machine the contribution of the coils can be calculated a priori and this information is stored in the Greens Table. The method can be used with the plasma current as well although in practice it is too slow. Instead we use a finite difference form of the Grad Shafranov equation to determine that part of the poloidal flux which is due to the plasma current.

3.4 Current Hole Equilibria

The Greens function solution method, Eq.[**??**], lends some insight into the possibility of generating a current hole equilibrium. Any solution for $\boldsymbol{\Psi(R, Z)}$, must satisfy the boundary condition that $\boldsymbol{\Psi = 0}$ at the symmetry axis and at infinity. This condition is built into the integral equation, EQ.[**??**]. The integration over (R,Z) in Eq.[**??**] can be broken up into three regions: 1) $\boldsymbol{\Omega_1}$, the force free region, 2)$\boldsymbol{\Omega_2}$, the region inside the plasma but outside the curretn hole, and 3)$\boldsymbol{\Omega_3}$, the rgion outside the plasma that extends to infinity. The integral representation of $\boldsymbol{\Psi}$ can then be written as

$$\boldsymbol{\Psi(R,Z) = -\mu_0 \int_{\Omega_1} R_0 J_\phi(R_0,Z_0) G^*(R,Z;R_0,Z_0) dR_0 dZ_0 - \mu_0 \int_{\Omega_2} R_0 J_\phi(R_0,Z_0) G^*(R,}$$

The integral over region 1 vanishes since J is zero there. The integral over region 3 consists of currents flowig in external coils and hence can be considered to be known apriori. The interesting part is the region 2 integral which we view as a nonlinear integral equation for $\boldsymbol{\Psi}$. In this region we $\boldsymbol{J_\phi}$ must have the well known form :

$$\boldsymbol{J_\phi(R_0, Z_0) = R\frac{\partial P}{\partial \Psi} - \frac{F\frac{\partial F}{\partial \Psi}}{\mu_0 R}} \tag{40}$$

The pressure P and toroidal flux function $\boldsymbol{F\frac{\partial F}{\partial \Psi}}$ can be parameterized in tems of n unknown constants. For example suppose we model P and $\boldsymbol{F\frac{\partial F}{\partial \Psi}}$ as splines with n and m knots respectively. The objext is then to find the value of the pressure and $\boldsymbol{F\frac{\partial F}{\partial \Psi}}$ at each of the n and m knot locations respectively. This can be done by supllying n+m equations to solve for these n+m unknowns.

One way to get these n+m equations is to pick n+m points on the contour of $\boldsymbol{\Psi}$ that separates reqions $\boldsymbol{\Omega_1}$ and $\boldsymbol{\Omega_2}$. On this contour $\boldsymbol{\Psi = \Psi_c}$ and hence we have n+m equations of the form

$$\boldsymbol{\Psi_c = -\mu_0 \int_{\Omega_1} R_0 J_\phi(R_0,Z_0) G^*(R_i,Z_i;R_0,Z_0) dR_0 dZ_0 - \mu_0 \int_{\Omega_2} R_0 J_\phi(R_0,Z_0) G^*(R_i,Z_i;}$$

where i ranges from 1 to n+m. Because the spline parametrization is done in terms of normalized $\boldsymbol{\Psi}$ these equations are in fact nonlinear an iterative soltuion is called for. But this can be doen using existig tools available for example in Onetwo.

3.5 TDEM

The idea behind TDEM (ie Time Dependant Eqdsk Mode) is that often a significant amount of effort has ben made in generating equilibria for confinement analysis. These equilibria are subsequently not used in any transport studies. Rather,if time dependent mhd information is to be included in the transport model,then it is generated anew inside the transport code. This is a non interactive process and generally will not be able to match the finely tuned equilibria already available without a significant amount of additional effort. The TDEM mode allows us to circumvent this process by using the previously available eqdsk to generate the space and time dependent coefficients required in the diffusion equations a priori.

To use TDEM in Onetwo a file,*shotno.tdem*, in netcdf format,containing the required space and time dependent information must first be created. Netcdf is used so that this file can be easily read by other codes (eq IDL) as well as archived. This file is generated outside of Onetwo using the FORTRAN code MEPC and a graphical front end ,MEPC.tcl. To generate the file start up MEPC.tcl and click the help button which will provide all necessary information. Briefly,you use MEPC.tcl to generate a file,*??* that contains a list of eqdsks to be processed. MEPC.tcl then passes this list to MEPC which does the actual calculations and writes the file *??*. A partial third namelist for Onetwo is also created,in file *??* . You must copy this information into the third namelist of your Onetwo input file inone. The TDEM mode is invoked by setting $\boldsymbol{mhdmethod =' tdem', ieqdsk = 0, ifixshap = 0}$, and $\boldsymbol{mhdmode =' no\_coils'}$.

3.6 Non Uniform Grid and Fixed Boundary Calculations

Solving the Grad-Shafranov equation in cylindrical coordinates can be done efficiently if the edge of the plasma is approximated by a stair step type boundary. However this approximation is not always satisfactory and we would like to have a more accurate representation of te boundary. Inverse solvers don not have this problem but typically need a direct solver to supply information to other codes (e.g.the ray tracing codes curray and Toray and the neutral beam code Freya).

In Onetwo the plasma boundary is accommodated in a direct solver by using variable spacing in the $(\boldsymbol{R},\boldsymbol{Z})$ grid for points adjacent to the plasma boundary. The boundary thus becomes part of the computational grid in much the same way as it does for the rectangular boundary case. In this section we document how this is accomplished computationally and describe some routines that are useful when dealing with equilibrium files (eqdsks) that were generated using this method. Because the the number of radial grid points inside the plasma will vary as a function of height Z (and vice versa) we loose the regular structure that allows application of fast Poisson solvers (eq cyclic reduction). This can be compensated by the relative ease with which applicable solvers, such as successive over relaxation, can be run in parallel however.

Consider the standard $(\boldsymbol{R},\boldsymbol{Z})$ grid with nw radial and nh vertical grid points. The object of a fixed bound ay direct solver is to find $\boldsymbol{\psi(R,z)}$ by solving the G.S. equation with the plasma boundary fixed and known apriori. Hence this becomes a simple, elliptic, boundary value problem. For computational purposes we define an index, $\boldsymbol{k=(i-1)*nh+j}$, that will identify any grid point $\boldsymbol{(R_i,Z_j), i\in 1..nw, j\in 1,,nh}$ within the computational domain. The first step is to define an indicator array, called wzero in Onetwo, such that wzero(k) =1 if point k is inside the plasma and wzero(k) = 0 if point k is outside the plasma or on the plasma boundary. The determination of wzero is a standard application of concepts from computational geometry and is not discussed here.

All points in the computational domain but not on the rectangular boundary of the (R,Z) grid will be surrounded by four neighboring points. For grid point l these neighboring points are designated by lb,ll,lt,and lr for bottom,left,top, and right respectively.

In terms of our single indexing scheme point ll has index l-nh, point lt has index l+1,point lr has index l+nh and point lb has index l-1. The computation thus proceeds as follows. For each point k with wzero(k) =1 we check the four neighboring points. Any of the neighboring points that are outside(or on) the plasma will have wzero =0. this indicates that the grid spacing for that point, relative to the central point l, will have to be adjusted. There are in fact 16 possibilities (the number of combinations of 4 points taken 1,2,3 and 4 at a time) that need to be recognized as shown in the following code segment:

```
c  there are 16 cases to be accounted for:
c  ll  lt  lr  lb
c   1   1   1   1       case 15 all points inside
c   1   1   1   0            14 lb outside
c   1   1   0   1            13 lr outside
c   1   1   0   0            12 lr,lb outside
c   1   0   1   1            11 lt outside
c   1   0   1   0            10 lt,lb outside
c   1   0   0   1             9 lt,lr outside
c   1   0   0   0             8 lt,lr,lb outside
c   0   1   1   1             7 ll outside
c   0   1   1   0             6 ll,lb outside
c   0   1   0   1             5 ll,lr outside
c   0   1   0   0             4 ll,lr,lb ouside
```

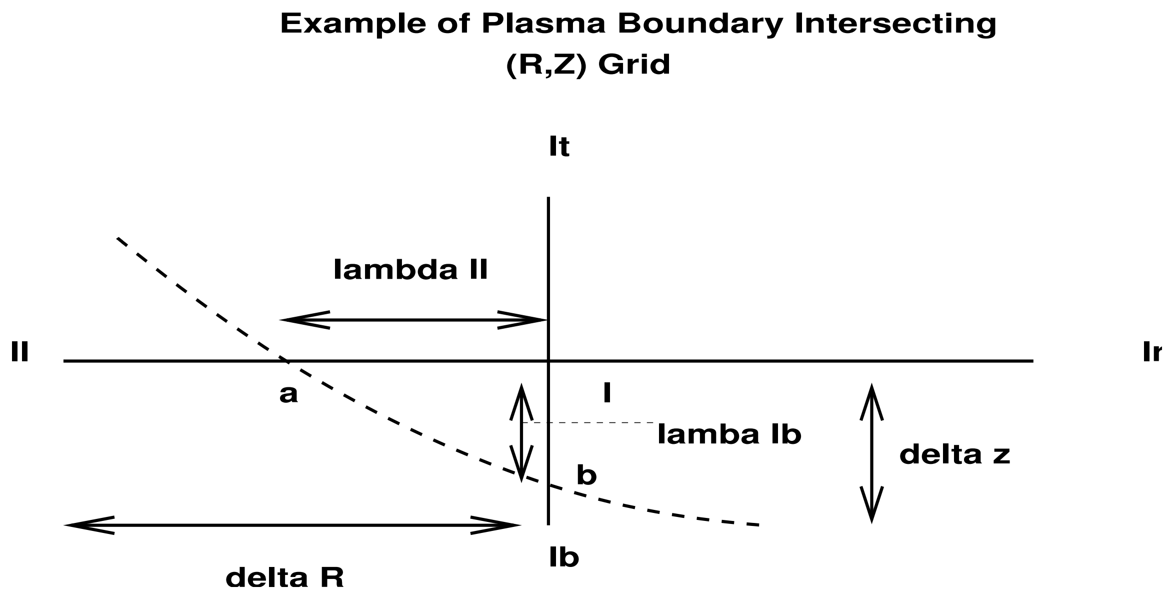


Figure 1: Irregular grid near plasma boundary. $\boldsymbol{\lambda_{ll}\Delta R}$ is the distance that boundary point a is to the left of grid point l. Similar definitions hold for the remaining $\boldsymbol{\lambda}$

```
c   0   0   1   1            3 ll,lt outside
c   0   0   1   0            2 ll,lt,lb outside
c   0   0   0   1            1 ll,lt,lr outside
c   0   0   0   0            0 ,ll,lt,lr,lb outside
c
c  case 10 has no vertical extent and is pathological
c        5       horizontal
c        0       vertical and no horizontal
```

We find the intersection of the plasma boundary contour with the $\boldsymbol{R, Z}$ grid lines using computational geometry methods . From these intersections we define the four parameters $\boldsymbol{\lambda_{lb}, \lambda_{ll}, \lambda_{lt}, \lambda_{lr}}$ associated with each grid point l. These four parameters represent the fraction of the uniform $\boldsymbol{\Delta R}$ and $\boldsymbol{\Delta Z}$ grid spacing associated with each of the points surrounding point l. For example, if all four neighboring points of point l are inside the plasma then all the $\boldsymbol{\lambda}$ are 1.0. If point ll is outside the plasma (l itself is inside of course) then the boundary contour crosses the radial line segment between point ll and point l and hence the actual distance to the boundary will be less than $\boldsymbol{\Delta R}$. $\boldsymbol{\lambda_{ll}}$ then gives the actual fraction of $\boldsymbol{\Delta R}$ that the boundary lies from point l in the direction of point ll. Similar definitions hold for the other three values of $\boldsymbol{\lambda}$. Given the distances that the neighboring grid points are from point l we can define a consistent set of first and second derivatives of $\boldsymbol{\Psi}$ by expanding $\boldsymbol{\Psi}$ in a Taylor series about the point l:

$$\boldsymbol{\Psi(R_i + r, Z_j + z) = \Psi(R_i, Z_j) + \frac{\partial \psi}{\partial R} r + \frac{\partial \psi}{\partial Z} z + 0.5 \frac{\partial^2 \Psi}{\partial R^2} r^2 + 0.5 \frac{\partial^2 \Psi}{\partial Z^2} z^2 + \frac{\partial}{\partial R}\frac{\partial}{\partial Z}\Psi \; rz}($$

The coordinates $\boldsymbol{(r, z)}$ are to be taken as corresponding to the four neighboring points relative to the central point l:

$$\boldsymbol{\Psi_{ll}(-\lambda_{ll}\Delta R, 0)} \tag{43}$$

Applied at the four neighboring points lb, ll, lt, lr the mixed second derivative contribution in the Taylor series is thus zero since either r or z is zero. The remaining derivatives in the Taylor series expansion (all evaluated at grid point l) are the (4) unknowns. Hence we need to solve the 4 by 4 system

$$-\frac{\partial\Psi}{\partial R}\lambda_{ll}\Delta R + \frac{1}{2}\frac{\partial^2\Psi}{\partial R^2}(\lambda_{ll}\Delta R)^2 = \Psi_{ll} - \Psi_l \tag{47}$$
$$\frac{\partial\Psi}{\partial Z}\lambda_{lt}\Delta Z + \frac{1}{2}\frac{\partial^2\Psi}{\partial Z^2}(\lambda_{lt}\Delta Z)^2 = \Psi_{lt} - \Psi_l \tag{48}$$
$$\frac{\partial\Psi}{\partial R}\lambda_{lr}\Delta R + \frac{1}{2}\frac{\partial^2\Psi}{\partial R^2}(\lambda_{lr}\Delta R)^2 = \Psi_{lr} - \Psi_l \tag{49}$$
$$-\frac{\partial\Psi}{\partial Z}\lambda_{lb}\Delta Z + \frac{1}{2}\frac{\partial^2\Psi}{\partial Z^2}(\lambda_{lb}\Delta Z)^2 = \Psi_{lb} - \Psi_l \tag{50}$$

for the derivatives $\frac{\partial\Psi}{\partial R}, \frac{\partial\Psi}{\partial Z}, \frac{\partial^2\Psi}{\partial R^2}, \frac{\partial^2\Psi}{\partial Z^2}$ for each grid point l that is in the vicinity of the plasma boundary. For points where all $\lambda = 1$ this is not necessary since the standard diamond difference scheme then supplies the values for the first and second derivatives. The required solution is

$$\left.\frac{\partial\Psi}{\partial R}\right|_l = -\frac{-\Psi_{lr}\lambda_{ll}^2 + \Psi_l\left(\lambda_{ll}^2 - \lambda_{lr}^2\right) + \lambda_{lr}^2\Psi_{ll}}{\lambda_R\Delta R} \tag{51}$$
$$\left.\frac{\partial\Psi}{\partial Z}\right|_l = \frac{\Psi_{lt}\lambda_{lb}^2 + \Psi_l\left(\lambda_{lt}^2 - \lambda_{lb}^2\right) - \lambda_{lt}^2\Psi_{lb}}{\lambda_Z\Delta z} \tag{52}$$
$$\left.\frac{\partial^2\Psi}{\partial Z^2}\right|_l = -2\frac{-\Psi_{lt}\lambda_{lb} + \Psi_l\left(\lambda_{lt} + \lambda lb\right) - \lambda lt\Psi_{lb}}{\lambda_Z(\Delta Z)^2} \tag{53}$$
$$\left.\frac{\partial^2\Psi}{\partial R^2}\right|_l = -2\frac{-\Psi_{lr}\lambda_{ll} + \Psi_l\left(\lambda_{ll} + \lambda lr\right) - \lambda lr\Psi_{ll}}{\lambda_R(\Delta R)^2} \tag{54}$$
$$\lambda_R = \lambda_{ll}\lambda_l r(\lambda_{ll} + \lambda lr) \tag{55}$$
$$\lambda_Z = \lambda_{lb}\lambda_l t(\lambda_{lb} + \lambda lt) \tag{56}$$

Note that these derivatives reduce to the standard, second order, finite diamond difference scheme when all the λ are equal to 1. Hence we have a simple method for incorporating the boundary but retaining the cylindrical (R, Z) grid.

One additional complication is that the points inside the plasma are renumbered so that we generate equations only for those points that we actually will find a value of psi for. This array, called map(m) ,m =1,..nw*nh, in Onetwo is defined so that map(k) =j means that the grid point numbered k in the rectangular computational domain is actual grid point number j when we solve the discretized GS equation. The solution of the resulting set of equations can be done most easily with an iterative technique such as SOR.

Since most codes that Onetwo uses (eq Freya,Toray,Curray,etc.) routinely require the solution $\boldsymbol{\Psi}$ in the entire rectangular domain we also need to find the solution outside the plasma. Thus in the region outside the plasma, but inside the rectangular domain, we solve the equation $\bigtriangledown^{*}\boldsymbol{\Psi} = \mathbf{0.0}$ with the known value of Grad Psi on the plasma boundary and some known values of $\boldsymbol{\Psi}$ on the rectangular $\boldsymbol{(R, Z)}$ boundary. Typically these later boundary points are assumed to be time independent and given by the solution of the complete free boundary value problem (as determined by Efit for example).

The user can verify that the discretized GS equation now takes the form

$$\boldsymbol{F_l \equiv \Psi_{ll}\alpha_{ll} + \Psi_l\alpha_l + \Psi_{lr}\alpha_{lr} + \Psi_{lt}\alpha_{lt} + \Psi_{lb}\alpha_{lb} - S_l(\Delta R)^2 = 0} \tag{57a}$$

$$\boldsymbol{S_l = -\mu_0 R^2 \frac{\partial P}{\partial \Psi}\bigg|_l - f\frac{\partial f}{\partial \Psi}\bigg|_l} \tag{57b}$$

$$\boldsymbol{\alpha_{ll} = 2\frac{\lambda_{lr}}{\lambda_r} + \Delta R\frac{\lambda_{lr}^2}{\lambda_r R_l}} \tag{57c}$$

$$\boldsymbol{\alpha_{lr} = 2\frac{\lambda_{ll}}{\lambda_r} - \Delta R\frac{\lambda_{ll}^2}{\lambda_r R_l}} \tag{57d}$$

$$\boldsymbol{\alpha_{lt} = 2\frac{\lambda_{lb}}{\lambda_z}\left(\frac{\Delta R}{\Delta z}\right)^2} \tag{57e}$$

$$\boldsymbol{\alpha_{lb} = 2\frac{\lambda_{lt}}{\lambda_z}\left(\frac{\Delta R}{\Delta z}\right)^2} \tag{57f}$$

$$\boldsymbol{\alpha_l = -2\frac{\lambda_{ll} + \lambda_{lr}}{\lambda_r} + \Delta R\frac{\lambda_{ll}^2 - \lambda_{lr}^2}{\lambda_r R_l} - 2\frac{\lambda_{lt} + \lambda_{lb}}{\lambda_Z}\left(\frac{\Delta R}{\Delta Z}\right)^2} \tag{57g}$$

Here $\boldsymbol{F_l}$ labels the GS equation for grid point (i,j) where $\boldsymbol{l = (i-1) * nh + j}$ and we assume that there are nw radial and nh vertical grid points. The non linearity in the source term,$\boldsymbol{S_l}$, in Eq[57a,57b] can be handled using Picard (also called successive substitution) iteration and this option exists in Onetwo. However it is possible to apply a globally convergent Newton based approach to these equations as follows.

Let us suppose that for the usual nw(in R) by nh(in Z) grid there are n grid points interior to the plasma for which we wish to find a solution. Typically $\boldsymbol{n = \zeta * nw * nh}$ where $\boldsymbol{\zeta \approx 0.8}$. Hence we have to solve an n by n system of equations that is both large and sparse. Each of these equations has the form given by Eq[57a]. The n by n Jacobian of this system will have entries of the form

$$J(l,l) \equiv \frac{\partial F_l}{\partial \Psi_l} = 1. - \frac{(\Delta R)^2}{\alpha_l}\frac{\partial S_l}{\partial \Psi_l} \tag{58a}$$

$$J(l,l-nh) \equiv \frac{\partial F_l}{\partial \Psi_{ll}} = \frac{\alpha_{ll}}{\alpha_l} - \frac{(\Delta R)^2}{\alpha_l}\frac{\partial S_l}{\partial \Psi_{ll}} \tag{58b}$$

$$J(l,l+nh) \equiv \frac{\partial F_l}{\partial \Psi_{lr}} = \frac{\alpha_{lr}}{\alpha_l} - \frac{(\Delta R)^2}{\alpha_l}\frac{\partial S_l}{\partial \Psi_{lr}} \tag{58c}$$

$$J(l,l+1) \equiv \frac{\partial F_l}{\partial \Psi_{lt}} = \frac{\alpha_{lt}}{\alpha_l} - \frac{(\Delta R)^2}{\alpha_l}\frac{\partial S_l}{\partial \Psi_{lt}} \tag{58d}$$

$$J(l,l-1) \equiv \frac{\partial F_l}{\partial \Psi_{lb}} = \frac{\alpha_{lb}}{\alpha_l} - \frac{(\Delta R)^2}{\alpha_l}\frac{\partial S_l}{\partial \Psi_{lb}} \tag{58e}$$

Because the source term S is evaluated at grid point l all derivatives of S with respect to the other grid points vanish and hence this term survives only in Eq[58a]. Typically the terms $\frac{\partial P}{\partial \Psi}$ and $f\frac{\partial f}{\partial \Psi}$ that S contains are parameterized in terms of a normalized $\bar{\Psi}$ and the dependence of $\bar{\Psi}$ on the values of Ψ distributed over the grid must be included in Eq[58a]. Hence we have

$$\left.\frac{\partial S_l}{\partial \Psi_l}\right|_{total} = \frac{\partial S_l}{\partial \Psi_l} + \frac{\partial S_l}{\partial \bar{\Psi}}\frac{\partial \bar{\Psi}}{\partial \Psi_l} \tag{59}$$

Typically the magnetic axis is found by fitting a bicubic spline to the current estimate of Ψ over the (R,Z) grid and then searching for the location where $\nabla\Psi = 0$. This creates an analytically intractable dependence of $\bar{\Psi}$ on Ψ_l however. Furthermore resolving this dependency numerically would be too costly since it leads to filling in many of the elements in the otherwise sparse Jacobian. It is in fact possible to create an approximation to the true Jacobian by simply neglecting the term $\frac{\partial S_l}{\partial \bar{\Psi}}$. This does not destroy the positive definiteness of the Jacobian so that we still obtain a direction that leads to decreasing the residuals F_l. But in practice this slows down the convergence rate of the outer iterations to a degree where use of the Newton method is no longer attractive.

The alternative used in Onetwo is to create a 12 point stencil centered on the location of the magnetic axis as shown in Fig[2]. The fiducial point $k_a \equiv (i_a - 1) * nh + ja$ is found as the lower left point of the rectangle ka,ka+1,ka+1+nh,ka+nh This rectangle is is found by searching for two successive R grid points where $\frac{\partial \Psi}{\partial R}$ changes sign and twp successive z points where $\frac{\partial \Psi}{\partial Z}$ changes sign. These derivatives are given by Eq[**??**] and hence, in terms of Ψ the twelve grid points ka-nh,....ka+1+2*nh become involved. Given this information it is easy to find the line $\overline{de}$ which represents the locus of point on which $\frac{\partial \Psi}{\partial Z} = 0$. Similarly the line $\overline{bc}$ represents the set pf points

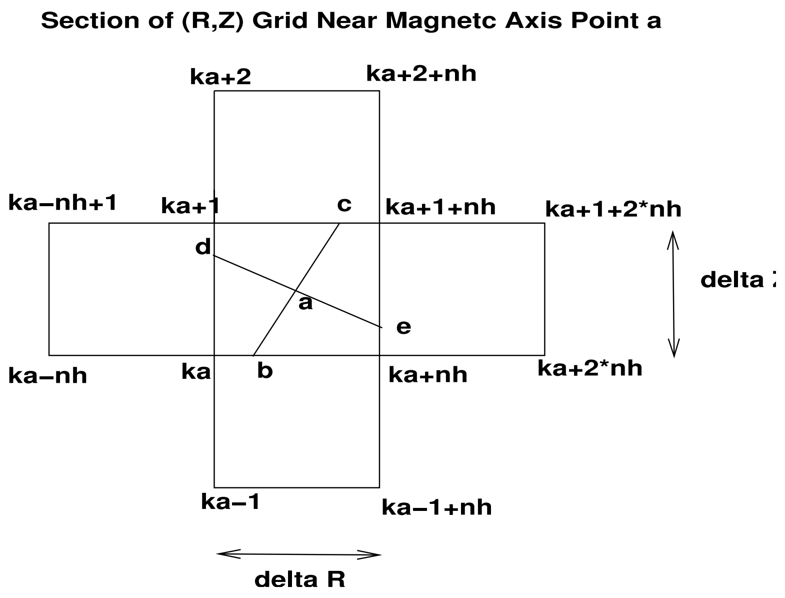


Figure 2: Coordinates involved in determination of magnetic axis

where$\frac{\partial \boldsymbol{\Psi}}{\partial \boldsymbol{R}} = \mathbf{0}$.(Although shown as tilted in the figure, most likely these lines will turn out to be vertical and horizontal for most tokamak equilibria). The intersection point a represent the estimated location of the magnetic axis. Since $\boldsymbol{\Psi}$ at the magnetic axis is thus shown to depend on the 12 given points it is straightforward to incorporate that dependency into the Jacobian. Using this approach it is possible to give an analytic form for the elements $\frac{\partial \boldsymbol{S_l}}{\partial \bar{\boldsymbol{\Psi}}} \frac{\partial \bar{\boldsymbol{\Psi}}}{\partial \boldsymbol{\Psi_l}}$ But since interpolation of the normalized $\bar{\boldsymbol{\Psi}}$ is required anyway we use a numerical perturbative approach for this term.

Hence the Jacobian,J,is known and determined semi analytically. To continue we need to solve the n by n set of equations

$$\underline{\underline{\boldsymbol{J}}} * \underline{\boldsymbol{\psi}} = -\underline{\boldsymbol{F}} \tag{60}$$

Here $\underline{\boldsymbol{F}}$ represents the column of residuals with elements $\boldsymbol{F_l}$ given by Eq.[57a],and $\boldsymbol{\psi}$ represents the incremental corrections to the $\boldsymbol{\Psi}$ values to be made if the full Newton step is accepted. Typically a line search along the Newton step is required in order to avoid divergence of the iterations. The near optimal method for this search used in Onetwo is given in ref?. The n by n system given in Eq[60] is large (for a 129 by 129 grid we will have about 13000 values of $\boldsymbol{\Psi}$ to determine) but sparse. The bandwidth is at least 2* nh and the matrix J does not have a regular structure due to the fact that the number of grid points in the radial and vertical directions is not uniform. In Onetwo we currently use a sparse matrix LU factorization technique (y12m) to solve the equations. Since there is some fill in encountered during the solution we are limited by hardware memory requirements however. A better approach is to use an iterative solver such as a bi conjugate gradient method but such software was not available to me. (A sparse multigrid algorithm does not exist to my knowledge). At this time, for grid sizes larger than 129 by 129 the Newton method is not used. Instead the Picard iteration method is relied on for the outer iterations . In fact high resolution eqdsk are often created by running a 65 by 65 EFIT type eqdsk through the code to produce eqdsk as large as 513 by 513 using Picard iteration.

An example of an ITER type equilibirum calculated using the SOR method on a513 by 513 grid is given in Fig[14]

3.6.1 The Exterior Problem

The region exterior to the plasma but inside the rectangular $(\boldsymbol{R}, \boldsymbol{Z})$ domain must also be considered in order to genrate an eqdsk. In this exterior region we solve the GS equation with zeero current density, subject to the boundary conditons that grad psi is given on the plasma surface and psi is given on the edges of the rectangualr domain. The gradient of psi on the plasma surface is computed from the interior solution. The values of psi on the rectanglar boundary are assumed given. The finite difference representation of the GS equation for grid points outside the plasma but near the plasma boundary is slightly more complicated than the interior case. Consider Fig**??**f1e] where we assume that grid pts lb and lr are inside the plasma. Three gradients of $\boldsymbol{\Psi}$ at the points a,b,c respectively are used to generate the equations for the required derivatives at exterior grid point l. In the figure pts a and c are intersections of the plasma boudnary with the grid lines. Point b was not used in the interior calculations. It is defined by the fact that the direction of the gradient at point b passes through grid point l. Depending on the particular geometry it is possible that point b coincides either with point a or point c. This leads to degeneracy in the equations described below.

We expand $\boldsymbol{\Psi}$ in a Taylor series about the exterior grid point l, Eq[42], and differentiate the result to get

$$\frac{\partial \Psi(r,z)}{\partial R} = \left.\frac{\partial \Psi}{\partial R}\right|_l + \left.\frac{\partial^2 \Psi}{\partial R^2}\right|_l r + \left.\frac{\partial}{\partial R}\frac{\partial}{\partial Z}\Psi\right|_l z \qquad (61)$$

A similar expression holds for the derivative in z. Applied at the points a,b,c in Fig[4] we get the five equations in the five unknowns $\frac{\partial \Psi}{\partial R}|_l, \frac{\partial \Psi}{\partial Z}|_l,$ $\frac{\partial^2 \Psi}{\partial R^2}|_l, \frac{\partial^2 \Psi}{\partial Z^2}|_l,$ $\frac{\partial}{\partial R}\frac{\partial}{\partial Z}\Psi|_l$.

The rhs of eqs represents the assumed knonw gradient of $\boldsymbol{\Psi}$ on the plasma boundary. The last two equations are obtained by applying the Taylor series at the grid points ll and lt. There are 16 different cases that can arise,as previously dicussed for the interior solution. Equations similar to the above set have to be generated for each of the cases. It is always possible to generate 5 equations however. The details are given in subroutine ??.

The remaining issue is how to generate the values of $\boldsymbol{\nabla\Psi}$ on the plasma boundary so they can be used in solving the above equations. In Onetwo this is accomplished in an iterative manner. We first generate an exterior solution by replacing the $\boldsymbol{\nabla\Psi}$ boundary condition on the plasma surface with the condition that $\boldsymbol{\Psi = 0}$ on the plasma surface. Combined with the interior soltuion this gives us a soltuion over th entire rectangular (R,Z) domain. We bicubic spline fit this solution and from the spline fit determine the value of $\boldsymbol{\nabla\Psi}$ at all required boundary points. The exterior problem is then solved a second time with the $\boldsymbol{\nabla\Psi}$ boundary condition applied. The interior soltuion is not changed in this step. This process could be iterated but in practice this is not necessary.(The boundary condition on the outer, rectangualr, boundary is an approximation, not based on coil currents and hence any refinement would be ludicrous).

3.7 Transport- mhd interface

Proper communication between the transport system of equations and the equilibrium (GS) equation is very important in order to get an efficient, stable transport/equilibrium iteration cycle. The time step used to evolve the transport equations,$\boldsymbol{\Delta t_t}$, and the time step between equilibrium calculations,$\boldsymbol{\Delta t_{eq}}$, are independent quantities in Onetwo (but $\boldsymbol{\Delta t_{eq} \geq \Delta t_t}$ must be observed). We typically do not call the equilibrium solver at every transport time step. Instead, a number of transport time steps are taken before a new equilibrium is calculated. The actually number of transport time steps between equilibrium calculations is initially set by the user and is then dynamically decreased if necessary to assure convergence. (There are actually a number of options for setting the intervals $\boldsymbol{t_t}$ and $\boldsymbol{t_{eq}}$ in Onetwo, see the input instructions in cray102.f)

Evolution of the transport quantities depends on space and time dependent metrics obtained from the equilibrium solution. Initially the space dependence is known from a start up equilibrium but the time dependence is not known at all. The actual method used to overcome this difficulty in Onetwo is as follows.

Initially we have an equilibrium solution (in the form of an eqdsk) from a previous fixed boundary mhd calculation. The reason we must start with a fixed boundary eqdsk (meaning an eqdsk that was generated using the fixed boundary code),is that the metrics (see below), do not converge as well at the plasma edge if a standard Efit type eqdsk is used. The fixed boundary code actually solves a problem which is different than the problem that is solved by EFIT. The difference is due to the way that the plasma edge is handled. The fixed boundary code solves two boundary value problems, the interior problem, with the plasma boundary as the mathematical boundary and the exterior problem, with the plasma boundary as one boundary and the boundary of the rectangular grid as the outer boundary.For the interior problem we specify a value of psi at the plasma boundary. For the exterior problem the boundary condition on the plasma boundary is that grad psi is given. This value of grad psi is determined from the interior solution near the plasma boundary. On the rectangular boundary of the exterior problem the values of psi are specified. In Onetwo these values are assumed to be equal to the values at the initial time and do not change in time. Effectively this means that some pseudo coils with appropriate,perhaps non physical, currents are present. One could supply actual machine dependent coils and use Greens function methods to refine the exterior problem but that has not been done at this time.

The eqdsk is used to construct the initial geometry factors, current density and transport grid:

$$\Phi|_{t_0} = 2\pi \int q d\Psi \tag{62a}$$

$$\rho|_{t_0} = \sqrt{\frac{\Phi}{\pi B_{t0}}} \tag{62b}$$

$$F|_{t_0} = \frac{R_0 B_{t0}}{f(\Psi)} \tag{62c}$$

$$G|_{t_0} =< |\nabla\rho|^2 \frac{R_0^2}{R^2} > \tag{62d}$$

$$H|_{t_0} =< \frac{F}{\frac{R_0^2}{R^2}} > \tag{62e}$$

The profiles required in the rhs of Eqs.[62] are are available from the startup eqdsk on a uniform Ψ grid of length nw. Note that these quantities are functions of both space and time. Initially the time dependence is only known at the single time point as indicated. Thus to solve the transport equations for the duration of the first equilibrium cycle we assume that the metrics and other quantities are constant in time as given by Eq.[62].

The transport equations are evolved, using time steps Δt_t, from some initial time t_0 to time $t_1 = t_0 + \Delta t_{eq}$. At time t_1 the equilibrium solver is called with the evolved $\frac{\partial P}{\partial \Psi}$ and $< \frac{J_\phi R_0}{R} >$. The total pressure profile normally includes beam and fusion contributions (using $\frac{2}{3}$ of the stored energy density for the fast ion contributions). These quantities are evolved on the $\rho|_{t0}$ grid and need to be mapped into the appropriate Ψ grid for use in the equilibrium solver. The conversion factor,$\frac{\partial \rho}{\partial \Psi}$, is obtained from the transport results at time t_1. As shown in section 2 Onetwo evolves the quantity $u^4 \equiv FGH\rho B_{p0}$. From the definition of B_{p0} we then obtain the Ψ to ρ mapping:

$$B_{p0}|_{t_1} = \frac{u^4|_{t_1}}{(FGH\rho)|_{t_0}} \tag{63a}$$

$$\frac{\partial \rho}{\partial \Psi}|_{t_1} = \frac{1}{R_0 B_{p0}|_{t_1}} \tag{63b}$$

Since the time dependence of the the parameters associated with the evolution of u^4 during the time interval t_0 to $t_0 + \Delta t_{eq}$ was neglected these quantities are only an approximation to the desired results. Eq.[63b] gives us an in ital approximation of the ρ to Ψ mapping required to evaluate $\frac{\partial P}{\partial \Psi}$ for example.

To get a form for $f\frac{\partial f}{\partial \Psi}$ driven by transport we use $< \frac{J_\phi R_0}{R} >$. This quantity is also obtained from u^4:

$$< \frac{J_\phi R_0}{R} > |_{t_1} = \left(\frac{\mu_0}{H\rho} \frac{\partial}{\partial \rho} (\frac{u^4}{F}) \right) \Bigg|_{t_1} \tag{64a}$$

We map both $\frac{\partial P}{\partial \Psi}$ and $< \frac{J_\phi R_0}{R} >$ to the Ψ grid by integrating Eq.[63b]:

$$(\Psi(\rho_i) - \Psi(0)) |_{t_1} = \left(R_0 \int_0^{\rho_i} B_{p0} d\rho \right) \Bigg|_{t_1} \tag{65}$$

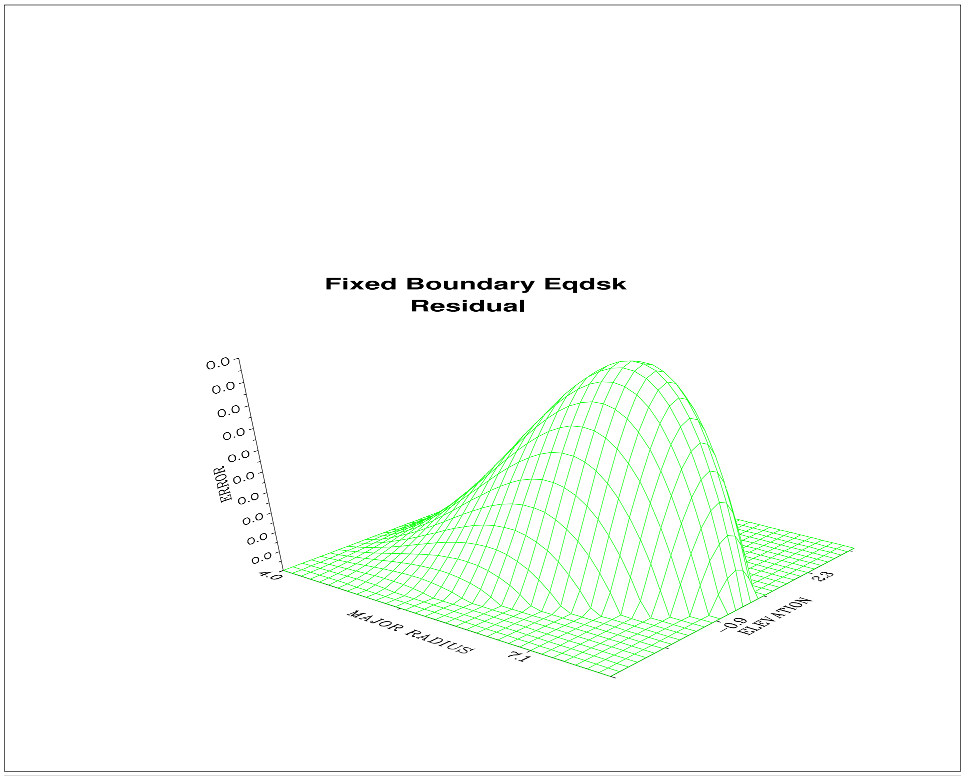
where ρ_i is ρ at radial transport grid point i. Eq.[65] is evaluated in subroutine psirho(cray209.f) and produces a psi grid that corresponds to the transport rho grid (the radial transport rho grid extends from the magnetic axis to the plasma edge and has nj grid point). The toroidal current density,Eq.[66a], is now flux surface averaged to produce and expression for $f\frac{\partial f}{\partial \Psi}$:

$$J_\phi = -R\frac{\partial P}{\partial \Psi} - \frac{f\frac{\partial f}{\partial \Psi}}{u_0 R} \tag{66a}$$

$$\left(f\frac{\partial f}{\partial \Psi} \right) \Bigg|_{t_1} = -\frac{u0}{< \frac{R_0}{R^2} >} \left(R_0 \frac{\partial P}{\partial \Psi} + < \frac{J_\phi R_0}{R} > \right) \Bigg|_{t_1} \tag{66b}$$

Armed with $f\frac{\partial f}{\partial \Psi}$ from Eq.[66b], and $\frac{\partial P}{\partial \Psi} = \frac{\partial P}{\partial \rho}\frac{\partial \rho}{\partial \Psi}$ from Eq.[63b] we are finally able to solve the GS equation. We note again that both of these quantities are known on a Ψ grid which corresponds to the radial transport ρ grid with nj (typically 51) grid points. For the initial step the radial transport grid and all metrics required are given by the values at time $\boldsymbol{t_0}$,Eq.[62]. Note that there are additional quantities that depend on flux surface averages, for example the trapped particle fraction. These quantities are correctly handled by the mhd/transport coupling of the code but are not dealt with explicitly here.

Having obtained a new equilibrium solution at time $\boldsymbol{t_0 + \Delta t_{eq}}$ with the given $\boldsymbol{f\frac{\partial f}{\partial \Psi}}$ and $\boldsymbol{\frac{\partial P}{\partial \Psi}}$ (which still have dependence on parameters defined at time $\boldsymbol{t_0}$) we need to check for convergence of the transport/equilibrium cycle. This is done by evaluating Eqs.[62] with the newly calculated equilibrium. Comparison of the new parameter set at time $\boldsymbol{t_0 + \Delta t_{eq}}$ with the original set at time $\boldsymbol{t_0}$ then yields information as to whether or not our mhd/transport system is sufficiently converged to continue on with the next time interval ,$\boldsymbol{t_0 + t_{eq}}$ to $\boldsymbol{t_0 + \Delta t_{eq} + \Delta t_{eq_1}}$. Here $\boldsymbol{\Delta t_{eq_1}}$ is a new equilibrium time interval which may be determined by the code based on how rapidly parameters are changing. A minimum maximum error criteria is used to establish if our assumptions of time independent metrics was justified. At this point, converged or not, we have estimates of the parameter set, Eqs.[62] at two times, $\boldsymbol{t_0}$ and $\boldsymbol{t_0 + \Delta t_{eq}}$. If the system is not sufficiently converged then we go back to time $\boldsymbol{t_0}$ and start over again. But this time all the parameters,including $\boldsymbol{\rho}$ are linearly interpolated in time to give an estimate of the time dependence.(Section 2 deals with the issues of a time Dependant $\boldsymbol{\rho}$ grid). Arriving in this way once again at time $\boldsymbol{t_0 + \Delta t_{eq}}$ we back average the new estimate of the parameter set with the old one to aid convergence. This process is allowed to continue for a user specified number of times. If the mhd/transport cycle is still not converged then the code either quits or (default) prints a warning message and continues on the the next equilibrium cycle anyway. New equilibrium cycles are just repeats of the above process,with the profiles linearly extrapolated on the first attempt to span the time interval $\boldsymbol{t_i + \Delta t_{eq_i}}$.



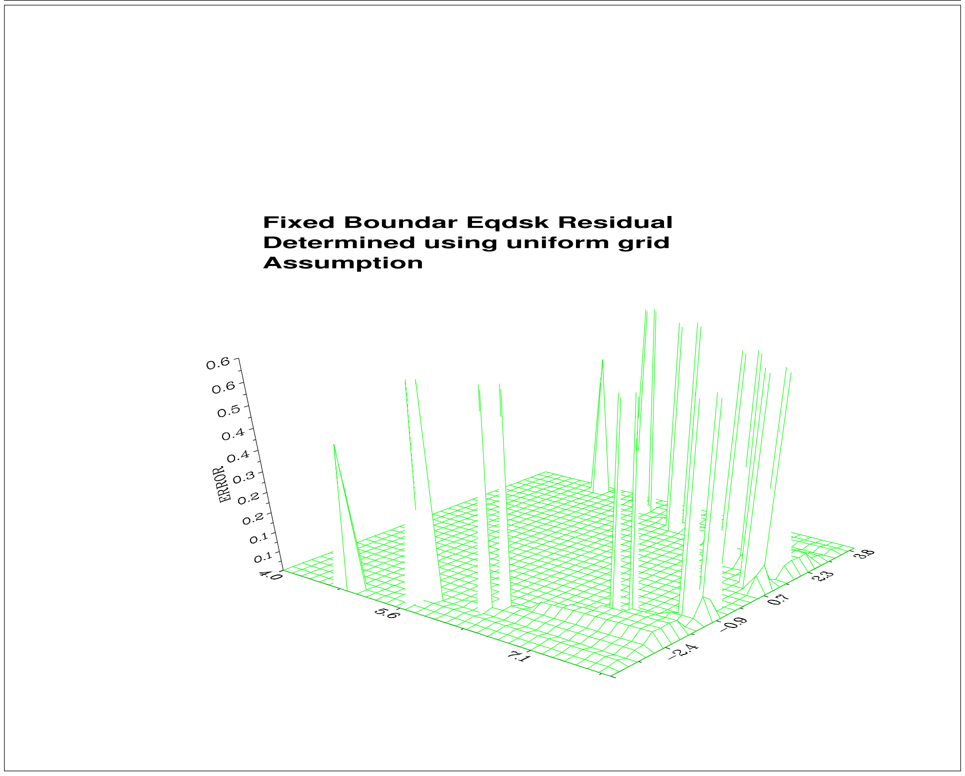


Figure 3: (a):Absolute value of residuals calculated using the variable grid method. As is to be expected the residual is zero near the plasma boundary (since that condition is built into the solver) - (b):ignoring the fact that a non uniform grid was used to generate the solution would lead one to conclude that large errors exist near the plasma boundary

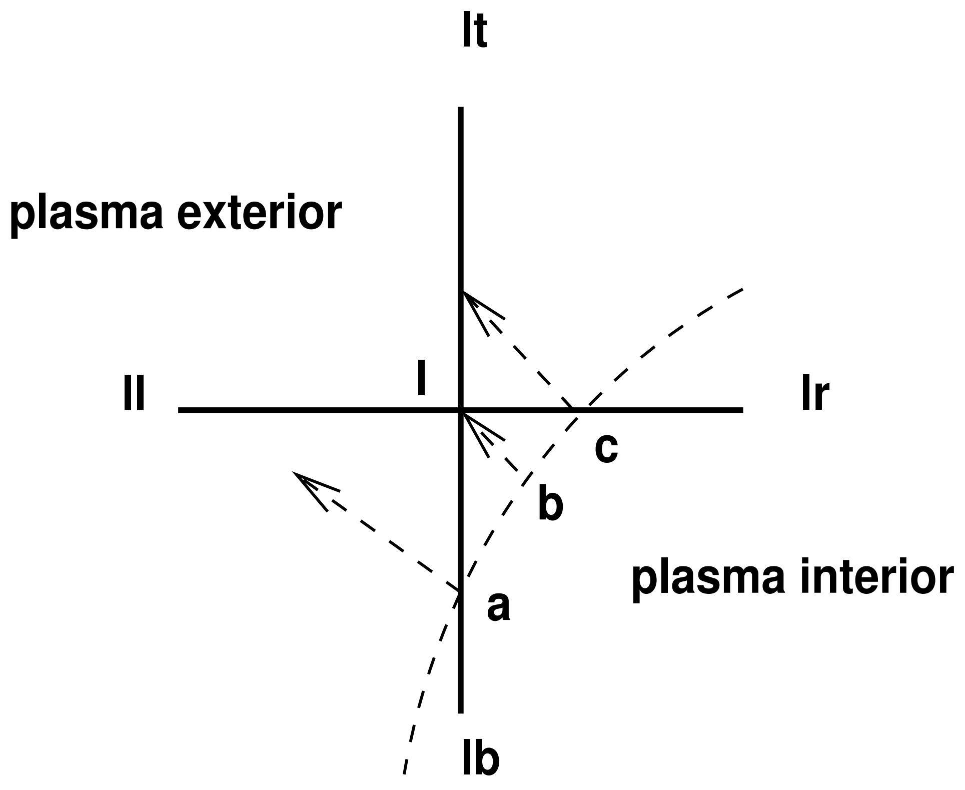


Figure 4: An example of an exterior grid point l,near the plasma boundary. Three gradients of psi at the points a,b,c are used to generate the equations for the required derivatives at grid point l

4 Source Terms

4.1 Beam Related Quantities

At the present time Onetwo relies on the Callen [2] model of fast ion slowing down to generate beam related results. An initial modification of the fast ion deposition profile to account for prompt orbit smearing is optional. Transient effects associated with beam turn on and off are accounted for as explained in section 4.1.1.

Let the index m range over all possible beam energies (and beam lines). Then the beam related terms are defined as follows:

$\boldsymbol{hibrz_m}$ The normalized fast ion birth rate. This is the basic quantity determined by Freya. Hibrz is the normalized birth distribution of fast ions per unit volume per unit time and is determined by

$$\boldsymbol{hibrz_m} = \frac{\frac{n_z}{V_z}}{\frac{n_t}{V_p}} \tag{67}$$

where $\frac{n_z}{n_t}$ is the fraction of ions that make it into the plasma and are born in poloidal flux zone z,$\boldsymbol{V_z}$ and $\boldsymbol{V_p}$ are the flux zone and total plasma volume respectively. Since $\boldsymbol{n_t}$ is given by

$$\boldsymbol{n_t} = \sum_{allzones} \boldsymbol{n_z} \tag{68}$$

we see that the volume average birth rate density is unity:

$$\frac{1}{V_P} \sum_{allzones} \boldsymbol{hibrz_m V_z} = \left(\frac{1}{n_t}\right) \sum_{allzones} \boldsymbol{n_z} = 1 \tag{69}$$

Typically hibrz has a value greater than 1 near the magnetic axes indicating beam penetration better than average .

$\boldsymbol{hdepz_m}$ Prompt orbit averaged version of hibrz (optional).

$\boldsymbol{hdep_m(\rho)}$ Either the birth or orbit averaged deposition (hbirz or hdepz depending on what the user selected) interpolated from the zone to the transport grid.

$\boldsymbol{bke_m}$ Toroidal momentum fraction transferred to electrons. Identical to $\boldsymbol{K_e}$ in [2].

$\boldsymbol{bki_m}$ Toroidal momentum fraction transferred to ions Identical to $\boldsymbol{K_i}$ in [2].

$\boldsymbol{fbe_m}$ The fraction of the initial beam energy that is deposited on electrons during the slowing down process. Identical to $\boldsymbol{G_e}$ in [2].

$\boldsymbol{fbi_m}$ The fraction of the initial beam energy that is deposited on ions during the slowing down process. Identical to $\boldsymbol{G_i}$ in [2]. Both $\boldsymbol{G_e}$ and $\boldsymbol{G_i}$ are based on a fast ion distribution function which allows for charge exchange during the slowing down process. However further re-ionization of the neutral created by the fast ion is neglected. $\boldsymbol{fbe, fbi}$ are output.

$\boldsymbol{pbeam_m}$ The beam power (input)

$\boldsymbol{xloss_m}$ Fraction of beam power lost due to aperture,orbit and shine through.

$\boldsymbol{qb(\frac{w}{cm^3})}$ Beam power deposited in the plasma:

$$\boldsymbol{qb = \sum_m qb_m} \tag{70}$$

$$\boldsymbol{qb_m(\rho) = (1.0 - xloss_m) * pbeam_m * hdep_m(\rho)/V_p} \tag{71}$$

($\boldsymbol{qb_m}$ is printed out in the beam tables)

$\boldsymbol{spbr\frac{g}{(cmsec^2)}}$ Toroidal angular momentum source.

$$\boldsymbol{spbr_m(\rho) = angmpf_m(\rho) * sb_m(rho)} \tag{72}$$

$\boldsymbol{sb_m}$ Is the source density of fast ions,it is defined in terms of the beam energy,$\boldsymbol{e_m}$,and $\boldsymbol{q_b}$ as:

$$\boldsymbol{sb_m(\rho) = qb_m(rho)/e_m} \tag{73}$$

$\boldsymbol{angmpf_m(\frac{gcm^2}{sec})}$ The average momentum of a single beam ion born in a flux zone. The number of flux zones (in poloidal flux space)is controlled by the input value *mf*. Let $\boldsymbol{n_\zeta}$ be the number of ions born in zone $\boldsymbol{\zeta}$. The ions have toroidal speed $\boldsymbol{v_I}$ and major radius $\boldsymbol{R_i}$ at the birth point. The average angular momentum of an ion in the zone is then given by

$$\boldsymbol{\overline{mvR} = \frac{1}{n_\zeta}\sum_{i\in\zeta} m_b v_i R_i} \tag{74}$$

and $\boldsymbol{angmpf_m}$ is this quantity interpolated onto the $\boldsymbol{\rho}$ grid,for beam energy component *m*.

Other beam related quantities required below are:

$\boldsymbol{\tau_s}$ The Spitzer momentum exchange time for electron-ion collisions

$$\boldsymbol{\tau_s = (\frac{3}{4\sqrt{2\pi}})\frac{\sqrt{m_e T_e^3}}{Z_f^2 n_e q_e^4 (24. - \ln(\frac{\sqrt{n_e}}{T_e}))}} \tag{75}$$

τ_f The fast ion lifetime,defined in terms of the critical speed,v_c (speed at which the fast ion slowing down rate on electrons and ions is equal) and the initial fast ion speed,v_b:

$$\tau_f = \frac{\tau_s}{3} \ln \left(1 + \frac{1}{(\frac{v_c}{v_b})^3} \right) \tag{76}$$

The fast ion lifetime against charge exchange is evaluated at the beam energy (per atomic mass unit) relative to the rotating ion distribution, $\frac{e_{rel}}{atw_f}$:

$$\tau_{cx} = \frac{1}{n_n \Xi} \tag{77}$$

n_n is the total neutral density and the reaction rate $\Xi \frac{cm^3}{sec}$ is a function of the relative speed between the beam ions and the thermal neutrals. The original version of Onetwo uses

$$\Xi = \langle \sigma_{cx} v \rangle$$

evaluated at the energy $\frac{e_{rel}}{atw_f}$ Where e_{rel} is the relative energy accounting for bulk rotation but not thermal motion of the neutrals. Furthermore the neutrals are assumed to have the same bulk rotation as the ions in this expression. A new optional form for Ξ is also available:

$$\Xi = \sigma(e_{rel}) v_{rel} (rtstcx)$$

The factor $rtstcx$ is intended to allow some scoping of the sensitivity of the fast ion distribution function to the charge exchange form assumed. Setting $rtstcx < -20.$ causes the code to use the average fast ion speed in determination of Ξ. At this time vrel neglects any thermal motion of neutrals as above. The Maxwellian charge exchange rate,$\langle \sigma_{cx} v \rangle$, and the cross section σ_{cx} are both taken from ref[[3]]. The code assumes $\sigma_{cx} v = 0$ if the energy is greater than 100 kev/amu.

$fbth_m$ represents the fraction of the fast ion population that thermalizes. (This leads to a source of thermal energy as well as thermal ions.) Onetwo assumes that the ratio $\frac{\tau_s}{\tau_{cx}}$ is independent of the fast ion speed. Consequently the fraction of the fast ions that thermalize without charge exchange becomes

$$fbth_m = exp(-\frac{\tau_f}{\tau_{cx}}) \tag{78}$$

Note that the rate at which fast ions are lost from the system due to charge exchange with thermal neutrals is consequently

$$sscxl_m = (1 - fbth_m)sb_m \tag{79}$$

Re-ionization of the these fast neutrals is neglected. sscxl is neglected in the particle source term S_i of Eqs.[220 ,**??**] but is included in the energy term qcx,see below.

4.1.1 Transient Beam Effects

Onetwo evolves the beam related quantities based on the rate equation with a constant source s_l in time interval δt_l:

$$\frac{\partial x}{\partial t} + \frac{x}{\tau_l} = s_l \tag{80}$$

Here x stands for any of a number of fast ion quantities identified in this document as having ben aged. The analytic solution of this equation at the end of the time interval δt_l ,given an initial condition x_0 at the start, is

$$x = x_0 \exp\left(-\frac{\delta t_l}{\tau_l}\right) + s_l \tau_l \left(1. - \exp\left(-\frac{\delta t_l}{\tau_l}\right)\right) \tag{81}$$

In the code the time step δt_l is governed by the predictor/corrector solution scheme (see section ?) and is equal to the time interval $\theta\delta t$ during the course of the solution. (At the start special adjustments are made as discussed below.) During each time interval the source of fast ions ,s_l,is either assumed equal to the source in the previous time interval or is replaced with a new source because a new beam deposition calculation was carried out,based on plasma conditions at the central time $t + \theta\delta t$. (No allowance for changing the beam power is made ,τ_l is also evaluated at that time)

4.2 Particle Sources

S_i is the source density for ions of species i in Eq.[220].

$$S_i = sbcx + scx + sbeam + sfusion + sion + srecom \tag{82}$$

sbcx $\frac{\#}{cm^3sec}$ A source of fast ions (and also thermal neutrals with energy $\frac{3}{2}T$) due to charge exchange of beam neutrals with thermal ions. sbcx is derived either from hbirz (no prompt orbit averaging) or hdepz (with prompt orbit averaging):

$$sbcx_i(\rho) = \sum_m sb_m hicm_m^i \tag{83}$$

$\boldsymbol{hicm_m^i}$ is the fraction of the fast ion birth rate for beam component m that leads to neutrals of type i.

scx $\frac{\#}{cm^3 sec}$ charge exchange between thermal ions and thermal neutrals. If two neutral species,corresponding to two primary ion species are present then scx will be a sink for one species and a source for the other. That is,ion species a charge exchanges with neutral species b producing a neutral of type a and an ion of type b. The charge exchange rate,$\boldsymbol{cx12r(\rho)}$ is based on the Freeman-Jones[3] cross sections. For ion density $\boldsymbol{n_i}$ and neutral density $\boldsymbol{n_{nj}}$ where $\boldsymbol{nj}$ is the other species we have:

$$\boldsymbol{scx_i(\rho) = n_i(\rho) n_{nj}(\rho) cx12r(\rho)} \tag{84}$$

If only one neutral species is present then there is no particle source, $\boldsymbol{scx = 0}$. (There is an energy source however because the ion and neutral temperatures are not assumed equal).

sbeam $\frac{\#}{cm^3 sec}$ Source of thermal ions due to beam slowing down. All fast ions which do not experience charge exchange during their lifetime are assumed to slow down into the corresponding thermal distribution:

$$\boldsymbol{sbeam_i(\rho) = \sum_m fbth_m(\rho) sb_m(\rho)} \tag{85}$$

where $\boldsymbol{sb}$ and $\boldsymbol{fbth_m}$ are defined above.

sion $\frac{\#}{cm^3 sec}$ Source of ions due to electron impact ionization of neutrals:

$$\boldsymbol{sion = \sum_{i=1}^{2} n_{n_i}(\rho) * eirate(\rho)} \tag{86}$$

where $\boldsymbol{eirate}$ is the electron impact ionization rate of of atomic hydrogen taken from [3]

$$\boldsymbol{eirate \equiv n_e \langle \sigma v \rangle} \tag{87}$$

$\boldsymbol{S_i^{2D}}$ represents a source term due to mhd evolution and is given by

$$\boldsymbol{S_i^{2D} = -n_i \frac{\partial}{\partial t} \ln H + \frac{d}{H} \frac{\partial}{\partial \rho} H n_i} \tag{88}$$

This term is included in $\boldsymbol{sother}$ in the table labeled “particle sources”.

4.3 Energy Sources

4.3.1 Electrons

The energy sources and sinks ($\frac{kev}{cm^3 sec}$) for Eq.[222] are as follows:

$\boldsymbol{Q_e}$ is given by

$$\boldsymbol{Q_e = -qexch + qohm - qrad - qione + qbeame + qrfe - qpe + qfuse} \quad (89)$$

qexch represents the electron ion energy exchange term

qohm represents ohmic heating

qrad represents radiative losses(

qione represents an energy sink due to recombination

qbeame represents electron energy source due to neutral beam heating. Let m be a multi-index that ranges over beam lines and beam energy components (ie full,half,and third). Then we may write

$$\boldsymbol{qbeame_m(\rho) = fbe_m(\rho)*qb_m(\rho)+bke_m(\rho)*spbr_m(\rho)*\omega(\rho)} \quad (90)$$

$$\boldsymbol{qbeame(\rho) = \sum_m qbeame_m} \quad (91)$$

qrfe represents electron energy source due to rf heating. Qrfe can be input,obtained from simple models,or dynamic coupling of Onetwo to Toray for ECH heating and current drive is available.

qpe represents energy loss due to pellet ablation

qfuse represents electron heating due to fusion.

$\boldsymbol{\omega L_e}$

$$\boldsymbol{\omega L_e = \omega * sprbeame} \quad (92)$$

$\boldsymbol{S^{2D}_{T_e}}$ represents heating of electrons due to evolvement of the mhd equilibrium.

$$\boldsymbol{S^{2D}_{T_e} = -\frac{5}{2} n_e T_e \frac{\partial}{\partial t} \ln H + \left(\frac{\partial \ln \rho}{\partial t}\right)\left(\frac{5}{2} n_e T_e \frac{\partial \ln H}{\partial \rho}\right.} \quad (93)$$

$$\boldsymbol{\left. + \frac{3}{2} T_e \sum_{i=1}^{nion} Z_I \frac{\partial n_i}{\partial \rho} + \frac{3}{2}\big(n_e + T_e \sum_{i=1}^{nion} n_z \frac{\partial Z_i}{\partial T_e}\big)\frac{\partial T_e}{\partial \rho}\right)} \quad (94)$$

In the output of Onetwo the definition

$$\boldsymbol{qe2d \equiv S^{2D}_{T_e}} \quad (95)$$

is used

4.3.2 Ions

The source terms ($\frac{kev}{cm^3 sec}$) for the ion energy equation,Eq.[225],are defined as follows:

Q The term Q appearing on the rhs of the ion energy equation is defined as

$$Q(\rho) = qexch + qioni - qcx + qbeami + qfusi + qrfi \quad (96)$$

qexch$\frac{kev}{cm^3 sec}$ is the electron ion energy exchange term due to Coul. collisions. (see above for def.)

qioni is defined as electron impact ionization of neutrals minus thermal ion recombination :

$$qioni = \frac{3}{2}\sum_{i=1}^{nprim}\Big(sion_i(\rho)T_{n_i}(\rho) - srecomb_i(\rho)T\Big) \quad (97)$$

qcx qcx is the compound term defined as

$$q_{cx}(\rho) = \sum_{i=1}^{2}\Bigg(\overbrace{1.5T(\rho) * sbcx_i(\rho)}^{a}$$
$$\overbrace{-1.5 * T_{n_i}(\rho) * sscxl_i(\rho) * ibcx}^{b}\Bigg)$$
$$\overbrace{+1.5 * n_{n_i}(\rho) * cext_i(\rho) * n_i(\rho) * (T(\rho) - T_{n_i}(\rho))}^{c}$$
$$\overbrace{+1.5 n_i n_{n_k} r_l * cxr(\frac{T}{atw_i}) * (T(\rho) - T_{n_k}(\rho))}^{d} \quad (98)$$

a represents LOSS of average energy $\frac{3}{2}T(Kev)$ per ion due to charge exchange with beam neutral

b represents source of energy due to fast ion thermal neutral charge exchange (ibcx=0 or 1,user selectable), $sscxl_i$ is the fraction of sscxl that leads to species i ions and T_{n_i} is the temperature of neutral species i.

c represents charge exchange with thermal neutrals of the same species

d represents charge exchange with neutral species k, r_l is a correction factor for plasma elongation.

qbeami represents heating of ions by the beam.

$$\boldsymbol{qbeami_m(\rho) = fbi_m(\rho)*qb_m(\rho)+bki_m(\rho)*spbr_m(\rho)*\omega(\rho) + fbth_m * sb_m(\rho) * erot} \quad (99)$$

$$\boldsymbol{qbeami(\rho) = \sum_m qbeami_m} \quad (100)$$

qfusi heating of ions due to fusion.

qrfi represents the rf ion heating term . Qrfi can be directly input into the code,some simple models are available internally in onetwo and dynamic coupling to the ray tracing code Curray is also available.

$\boldsymbol{S_T^\omega(\frac{kev}{cm^3 sec})}$ Represents the source of kinetic rotational energy which is the sum of four terms:

$$\boldsymbol{S_T^\omega = sprcxe + sprcxree + spreimpe + \omega L_e} \quad (101)$$

sprcxe Gives the source of rotational kinetic energy due to thermal charge exchange . If only a single neutral species is present then the sum involving k below is absent. For the two neutral (and hence also ion) case charge exchange can occur with a neutral of different mass as well as different momentum as given by the second sum:

$$\boldsymbol{sprcxe = \sum_{i=1}^{2} \frac{<R>}{<R^2>} n_{n_i} cexr_i n_i m_i (v_n^2 - v_z^2)} \quad (102)$$

$$\boldsymbol{+ \sum_{\substack{i=1\\k=3-i}}^{2} \left(\frac{<R>}{<R^2>}\right)^2 n_i cx12 n_k (m_k v_n^2 - m_i v_z^2)} \quad (103)$$

sprcxree is the ion rotational kinetic energy source due to recombination it is given in terms of sprcxre(Eq.[115])

$$\boldsymbol{sprcxree = -\frac{1}{2}\omega * sprcxre} \quad (104)$$

spreimpe Gives the source of rotational kinetic energy due to electron impact ionization of thermal neutrals. It is defined in terms of spreimpt (Eq. [113]):

$$\boldsymbol{spreimpe(\rho) = \frac{1}{2}\omega_n * spreimpt} \quad (105)$$

$\boldsymbol{S_T^{2D}}$ represents ion thermal energy sources due to time evolving mhd equilibria.

$$\boldsymbol{S_T^{2D} = -\frac{5}{2}T\frac{\partial \ln H}{\partial t}\sum_{i=1}^{nion} n_i + \frac{\partial \ln \rho}{\partial t}\Big(\frac{5}{2}T\frac{\partial \ln H}{\partial \rho}\sum_{i=1}^{nion} n_i + \frac{3}{2}T\sum_{i=1}^{nion}\frac{\partial n_i}{\partial \rho} + \frac{3}{2}\frac{\partial T}{\partial \rho}\sum_{i=1}^{nion} n_i\Big)} \quad (106)$$

$\boldsymbol{S_T^{2D\omega}}$ represents ion rotational kinetic energy sources due to time evolving mhd equilibria.

$$\boldsymbol{S_T^{2D\omega} = -\frac{1}{2}angrm2d(1) < R^2 > \omega^2\frac{\partial lnH}{\partial t}\sum_{i=1}^{nion} n_i m_i}$$
$$\boldsymbol{+\frac{1}{2}angrm2d(2)\omega\Big(spr2d+\omega(\frac{\partial < R^2 >}{\partial t} + < R^2 > \frac{\partial \ln H}{\partial t})\sum_{i=1}^{nion} m_i n_i\Big)}$$
$$\boldsymbol{-\frac{1}{2}amgrm2d(3)\omega^2\frac{\partial < R^2 >}{\partial t}\sum_{i=1}^{nion} n_i m_i} \quad (107)$$

The sum of the the evolving mhd related energy sources is called qi2d in the output of Onetwo:

$$\boldsymbol{qi2d \equiv S_T^{2D} + S_T^{2D\omega}} \quad (108)$$

The $\boldsymbol{angrm2d(1,2,3)}$ multipliers are user selectable input to Onetwo,defaulted to 1.0

4.4 Toroidal Momentum Sources

All source terms below are in units of $[\frac{g}{cm \cdot sec^2}]$ The electrons are assumed to have negligible momentum (ie no separate equation for the electron toroidal momentum is introduced. However some momentum sources associated with the fast ion electron interactions are included below as ion terms). At the present time sources and sinks associate with *ion impact ionization* are neglected.

$\boldsymbol{S_\omega}$ is defined as

$$\boldsymbol{S_\omega = Sprbeame + Sprbeami + Sprcxl + Spreimpt + Sprcx + Sprcxre} \quad (109)$$

where the definition of each of the individual terms follows.

sprbeame is the (delayed) source of angular momentum transferred from the (beam) fast ions to the electrons during the slowing down process.This angular momentum is rapidly shared with the ions and is thus a source term for the thermal ion toroidal momentum equation:

$$\boldsymbol{sprbeame(\rho) = bke(\rho, e_b, b_j) * spbr(\rho, e_b, b_j)} \tag{110}$$

sprbeami is the (delayed) source of angular momentum transferred from the (beam) fast ions to the thermal ion fluid.

$$\begin{aligned}\boldsymbol{sprbeami(\rho)} &= \boldsymbol{bki(\rho, e_b, b_j) * spbr(\rho, e_b, b_j) +} \\ &\quad \boldsymbol{fbth(\rho, e_b, b_j) * sb(\rho, e_b, b_j) * atw_b} \\ &\quad \boldsymbol{* m_p * v_z(\rho) \frac{< R^2 >}{< R >}}\end{aligned} \tag{111}$$

ssprcxl represents the gain of angular momentum due to charge exchange of a fast ion with a thermal neutral (the thermal neutral adds its momentum to the thermal ion distribution)

$$\begin{aligned}\boldsymbol{ssprcxl(\rho)} &= \boldsymbol{fprscxl * spbr(\rho, e_b, b_j)} \\ &\quad \boldsymbol{+ fscxl * sb(\rho, e_b, b_j) * atw_b} \\ &\quad \boldsymbol{* m_p * v_z(\rho) * \frac{< R^2 >}{< R >}}\end{aligned} \tag{112}$$

spreimpt represents gain of momentum due to electron impact ionization of thermal neutrals

$$\boldsymbol{spreimpt(\rho) = \sum_{i=1}^{nprim} eirate(\rho) * enn_i(\rho) m_i * vneut_i(\rho) * < R >} \tag{113}$$

sprcx represents the sorce/sink of momentum due to charge exchange of thermal neutral with thermal ion. For two ion and neutral species we have

$$\begin{aligned}\boldsymbol{sprcx(\rho)} &= \boldsymbol{\sum_{i=1}^{nprim} enn_i(\rho) * cexr_i(\rho)} \\ &\boldsymbol{en_i(\rho) * atw_i * m_p * < R > * (vneut_i(\rho) - vionz(\rho))} \\ &\quad \boldsymbol{+ cxmix * (atw_k * vneut_k(\rho) - atw_i * v_z(\rho))}\end{aligned} \tag{114}$$

sprcxre represents the sink of thermal angular momentum due to charge exchange of thermal ion with a fast neutral and also includes radiative recombination of thermal ions.

$$\boldsymbol{sprcxre(\rho) = \sum_{i=1}^{nprim} sbcx_i(\rho) * < R^2 > \omega m_i} \qquad (115)$$

$\boldsymbol{S_\omega^{2D}}$ is the source term due to evolution of the mhd equilibrium. It is given by

$$\boldsymbol{S_\omega^{2D} = -\omega \sum_{i=1}^{nprim} \Big(m_i n_i \frac{\partial}{\partial t} < R^2 > - m_i n_i < R^2 > \omega \frac{\partial}{\partial t} \ln H + \frac{d}{H} \frac{\partial}{\partial \rho} H \omega m_i n_i < R^2 > \Big)} \qquad (116)$$

spr2d is the name of this term in the code output. At present the individual contributions are not broken out. In the code this term is multiplied by an input factor $\boldsymbol{angrm2d(4)}$,which is defaulted to 1.0 but the user can assign any value (eq. 0.0) to gauge the effect of this term

4.5 Other Definitions used in Onetwo Output

4.5.1 Flux Tables

angmtm The total flux associated with toroidal rotation, (see Eq.[29]) is given in outone in the table labeled fluxes,under the column headed angmtm.

In the table labeled Energy Fluxes(ref. the ion energy equation, Eq.[225]): (flxangce is The energy flux due to particle convection

omegapi $\boldsymbol{\frac{kev}{cm^2 sec}}$ The energy flux associated with "conduction" (really viscosity) omegapi is defined as

$$\boldsymbol{omegapi \equiv \omega \Pi = \omega \Gamma_\omega^{cond}} \qquad (117)$$

See the discussion of Eq.[31] above regarding how this quantity is obtained in analysis and simulation modes.

cvctvrot $\boldsymbol{\frac{kev}{cm^2 sec}}$ The energy flux due to momentum convection,(flxangce internal to code),given by Eq.[5].

4.5.2 ION Power Balance Tables

The table labeled "ion Energy Sources" contains

$$\boldsymbol{qomegapi \equiv \frac{1}{H\rho}\frac{\partial}{\partial\rho}\Big(H\rho\omega\Pi\Big)} \tag{118}$$

$$\boldsymbol{omegale \equiv \omega L_e = \omega * sprbeame} \tag{119}$$

The following terms are used in the table "Ion Energy Sources Due to Angular Rotation" (ref. the ion energy equation,Eq.[225])

$$\boldsymbol{wdnidt(\rho) \equiv \sum_{i=1}^{nion}\frac{1}{2}m_i\omega^2 < R^2 > \ \frac{\partial n_i}{\partial t}} \tag{120}$$

$$\boldsymbol{niwdwdt(\rho) \equiv \sum_{i=1}^{nion}\frac{1}{2}m_i n_i\omega < R^2 > \frac{\partial \omega}{\partial t}} \tag{121}$$

$$\boldsymbol{omegdgam(\rho) \equiv qomeapi + vischeat} \tag{122}$$

$$\boldsymbol{vischeat(\rho) \equiv -\Pi\frac{\partial\omega}{\partial\rho}} \tag{123}$$

$$\boldsymbol{qangce(\rho) \equiv \frac{1}{H\rho}\frac{\partial}{\partial\rho}H\rho\Gamma_T^{\omega}} \tag{124}$$

$$\boldsymbol{thcx(\rho) \equiv sprcxe =} \tag{125}$$

$$\boldsymbol{rec + fcx(\rho) \equiv sprcxree} \tag{126}$$

$$\boldsymbol{e - impact(\rho) \equiv spreimpe =} \tag{127}$$

4.5.3 Momentum Balance Tables

The table labeled "Momentum Balance and Confinement Time" gives

$$\boldsymbol{qangce(\rho) \equiv} \tag{128}$$

The table labeled "Toroidal Rotation results" gives

flxangce The energy flux due to particle convection, Eq[??] is called flxangce in the code

The table labeled "Toroidal Rotation Sources " gives the terms that appear on the rhs of Eq.[229] These terms covered above in Sect. 2.4.

4.6 Neutral Beam Injection Tables

The column labeled "fast ion energy source" is the rate at which energy is currently instantaneously deposited in the plasma.This quantity is equal to $\boldsymbol{qb_m}$ in the steady state and is larger/smaller than $\boldsymbol{qb_m}$ during transient beam turn/off (see aging of beam parameters). The column labeled "delayed e. source" is the quantity $\boldsymbol{qb_m}$,see Eq.[71] (Note that $\boldsymbol{qbeame + qbeami + qcx = qb}$ at all times) $\boldsymbol{qb_m}$ is equal to the instantaneous energy deposition in the plasma in steady state. Otherwise $\boldsymbol{qb_m}$ reflects the finite buildup and decay times of the fast ion density. The columns labeled "energy fraction deposited in electrons /ions" are the quantities $\boldsymbol{fbe_m, fbi_m}$ (see Beam Related Quantities). The column labeled "p. slowing down time" gives the product $\boldsymbol{\tau_s N}$ defined by Eqs.[75,142] The column labeled "e. slowing down time" gives the product $\boldsymbol{\frac{1}{2}\tau_s G_e}$ (see beam section)

4.7 Items in the Summary Page

Beam power elec. $\equiv$ $\boldsymbol{pbel}$ The beam power delivered to the electrons is given by:

$$\boldsymbol{pbel = \sum_m fpe_m \int qb_m dV} \tag{129}$$

Beam power ions. $\equiv$ $\boldsymbol{pbion}$ The beam power delivered to the ions is given by:

$$\boldsymbol{pbion = \sum_m fpi_m \int qb_m dV} \tag{130}$$

Note that fast ion charge exchange is implicit in the terms $\boldsymbol{fpe_m}$ and $\boldsymbol{fpi_m}$ (the fast ion distribution function used to generate these quantities is given in Eq[136]),where the charge exchange factor,$\boldsymbol{P_{cx}}$,Eq.[137],is calculated using the fixed $\boldsymbol{\tau_{cx}}$ given by Eq.[77].

constant . See definitions under the beam section and Eq.[71].

Beam power cx loss $\equiv$ $\boldsymbol{pbcx}$ This value is gives the the beam power lost due to charge exchange. It is determined by what is left over after the beam power delivered to the electrons and ions is accounted for:

$$\boldsymbol{pbcx = \sum_m \int \left(qb_m - qbeame_m - qbeami_m\right) dV} \tag{131}$$

The sum is over beam components and energies (see Eqs.[71, 90,99]).

5 Neutron Rate Calculations

5.1 Thermonuclear Rate

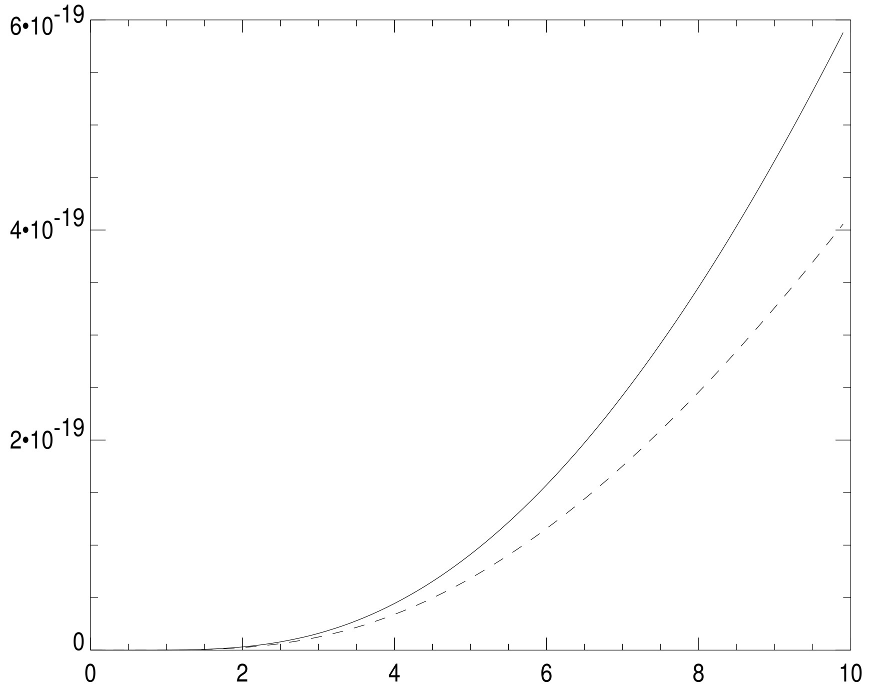


Figure 5: Neutron rate coefficients,$\frac{cm^3}{sec}$,for thermal $d(d,n)he^3$. Solid line is the new Bosch-Hale rate,dashed line is the $('94,'83)$ NRL rate

The NRL formulary parameterization of the reaction rate plotted in Fig.[5] is

$$<\sigma v>=\frac{2.33\cdot10^{-14}T^{-(\frac{2}{3})}exp[-18.76T^{-(\frac{1}{3})}]}{2} \tag{132}$$

THIS PARAMETERIZATION IS NOT ACCURATE AND SHOULD NOT BE USED. (It does not reproduce the rate tables in NRL with sufficient accuracy).

The Bosch and Hale parameterization is

$$< \sigma v > \; = \; C1 \cdot \Theta \sqrt{\frac{\xi}{m_r c^2 T^3}} exp(-3\xi) \quad (133)$$

$$\xi \; = \; \left(\frac{B_G^2}{4 \cdot \Theta} \right)^{\frac{1}{3}} \quad (134)$$

$$\Theta \; = \; \frac{T}{1. - \frac{T(C2+T(C6))}{1+T(C3+T(C5+TC7))}} \quad (135)$$

This expression precisely matches the rate tables in NRL *and* in Bosch and Hale.

5.2 Beam-Thermal Rate

The fast ion distribution function is assumed to be the uniform magnetic field,azimuthally symmetric solution of the Fokker-Planck equation,which asymptotically approaches

$$f_b(v,\zeta) = \frac{\dot{S}\tau_s}{v^3+v_c^3} P_{cx}(v) \sum_{l=0}^{\infty} \frac{2l+1}{2} P_l(\zeta) P_l(\zeta_b) \left[\frac{v^3}{v_b^3} \left(\frac{v_b^3+v_c^3}{v^3+v_c^3} \right) \right]^{\frac{1}{6}l(l+1)Z_2} \quad (136)$$

Where the probability against charge exchange is given by

$$P_{cx}(v) = exp\left(-\tau_s \int_v^{v_b} \frac{v^2 dv}{(v^3+v_c^3)\tau_{cx}} \right) \quad (137)$$

For the neutron rate calculations in Onetwo either the above integral for P_{cx} is evaluated numerically with the mean time against charge exchange given by

$$\tau_{cx} = \frac{1}{n_n \sigma_{cx}(v) v} \quad (138)$$

or it is assumed that τ_{cx} is constant,see Eq.[77] in which case we obtain

$$P_{cx}(v) = \left(\frac{v_b^3+v_c^3}{v^3+v_c^3} \right)^{-\frac{\tau_s}{3\tau_{cx}}} \quad (139)$$

For both cases the charge exchange cross section is the original Freeman and Jones [[3]] expression for charge exchange with hydrogen (not deuterium).

The fast ion distribution function,integrated over polar and azimuthal angles is

$$f_b(v) v^2 dv = \frac{\dot{S}\tau_s P_{cx}(v) v^2 dv}{v^3+v_c^3} \quad (140)$$

The density of fast ions is by definition

$$n_f = \dot{S}\tau_s N \tag{141}$$

$$N = \int \frac{P_{cx}(v)v^2 dv}{v^3 + v_c^3} \tag{142}$$

In ***ONETWO*** an effective time dependent source rate is defined by calculating P_{cx} and N analytically and then setting

$$\dot{S}\tau_s = \frac{n_f}{N} \tag{143}$$

at subsequent times. The fast ion density,n_f,is assumed to to build up or decay away with a time constant given by τ_f. Initially $\dot{S}\tau_s$ is obtained from ***NFREYA***. The speed dependent part of the fast ion distribution function is then taken as independent of P_{cx}. Using energy as the independent variable the transformed fast ion distribution becomes

$$f_b(E)dE = \frac{\frac{n_f}{N}dE}{2E(1 + (\frac{E_c}{E})^{\frac{3}{2}})} \tag{144}$$

The neutron rate density $[\frac{1}{cm^3 sec}]$ is given by

$$R = \int d\vec{v}_t d\vec{v}_b f_t(v_t) f_b(v_b) v_{rel} \sigma(v_{rel}) \tag{145}$$

One approximate formulation available in ***ONETWO*** assumes that the thermal ion speed, $v_t = 0$. The neutron rate density thus reduces to

$$R = \frac{n_d \frac{n_f}{N}}{\sqrt{(2m_b)}} \int \frac{dE\sigma_{DD}(E)}{E^{\frac{1}{2}}(1 + (\frac{E_c}{E})^{\frac{3}{2}})} \tag{146}$$

Using the approximate **(*NRL Formulary*)** cross section

$$\sigma_{DD} = \frac{c\epsilon^{\frac{-b}{\sqrt{E}}}}{E} \tag{147}$$

an analytic expression for R is obtained by assuming that $\frac{E_c}{E} = \frac{E_c}{E_b}$ in Eq(146).

The neutron rate density becomes (Eq.(148) was originally obtained from Scott())

$$R = \frac{\frac{n_f}{N} ckn_D e^{\frac{-b}{\sqrt{E_b}}}}{(1 + \left(\frac{E_c}{E_b}\right)^{\frac{3}{2}})b} \tag{148}$$

This is the (old) form used in $\boldsymbol{ONETWO}$. To account for bulk rotation the beam energy is modified to reflect its value in the rotating frame. In the rotating plasma frame the beam energy,E_b^R, is

$$E_b^R = E_b + \frac{m_b}{m_{th}} E_{bulk} - |v_f v_{bulk}| m_f cos(\theta) \tag{149}$$

Where $cos(\theta)$ is the angle between the velocity vector of the injected fast ion (which has an energy of E_b in the lab frame) and the (zero temperature) thermal ion (which has only bulk motion with velocity $\vec{v_{bulk}}$ and energy $E_{bulk} = \frac{1}{2} m_{th} v_{bulk}^2$. The beam-thermal neutron rate determined from Eq(148),with E_b replaced by E_b^R of Eq(149) is used in $\boldsymbol{ONETWO}$ when the input $iddfusb = 0$ is given.

A more accurate option available in $\boldsymbol{ONETWO}$ uses the Bosh and Hale( ) cross section,accounts for the Maxwellian nature of the thermal ion distribution, and does the integrals in Eq(150) for the neutron rate density numerically:

$$R = 2\pi\alpha\dot{S}\tau_s \int dv_{th} v_{th}^2 e^{-\beta v_{th}^2} \int dv_f \frac{v_f^2 P_{cx}(v) F(v_f, v_{th})}{v_f^3 + v_c^3} \tag{150}$$

$$F(v_f, v_{th}) = \int d\zeta_f \sigma(E_{com})(v_f^2 + v_{th}^2 - 2 v_f v_{th} \zeta_f)^{\frac{1}{2}}$$

Eq.(150) is selected by setting $iddfusb = 1$. The option of using the effective source $\frac{n_f}{N}$ together with neglecting $P_{cx}(v)$ may be used in this expression as well to speed up the computations(set $icalc\_cxrate = 0$). Otherwise,with $icalc\_cxrate = 1$ the computations are somewhat (but not significantly) slower due to the numerical evaluation of $P_{cx}(v)$,Eq(137). The time dependence of the fast ion distribution function is accounted for by adjusting the lower and upper limits of integration over the fast ion speed to reflect the fact that after beam turn on no fast ions exist below speed v_f^{lim} given by

$$v_f^{lim} = [(v_b^3 + v_c^3) exp(-3\frac{t}{\tau_s}) - v_c^3]^{\frac{1}{3}} \tag{151}$$

for times $t < \frac{1}{3}\tau_s \ln(\frac{v_b^3 + v_c^3}{v_c^3})$. Similarly, after beam turnoff, no fast ions exist above speed v_f^{lim} where t is measured from beam turnoff. Only a single beam turn on and off is accounted for in the code. Time dependent beam power models are currently not included. The calculations are done in the plasma frame with the initial beam energy given by Eq(149). To eliminate rotation from the neutron rate calculations of Eq(150) (by using E_b instead of E_b^R) set $iddfusb\_bulk = 0$.

5.3 Beam-Beam Rate

calculations are in place in the code but documentation is not yet done

6 Time Dependent Beam Input

6.1 The Fast Ion Slowing Down Problem

The simplest form of fast ion slowing down on a background plasma is given by the Fokker-Plank equation:

$$\tau_s \frac{\partial f_b}{\partial t}(v,\zeta,t) = \frac{1}{v^2}\frac{\partial}{\partial v}(v^3+v_b^3)f_b(v,\zeta,t) + \frac{Z_2 v_c^3}{2v^3}\frac{\partial}{\partial \zeta}(1-\zeta^2)\frac{\partial f_b}{\partial \zeta}(v,\zeta,t)$$
$$+\frac{\tau_s}{\tau_{cx}}f_b(v,\zeta,t) + \frac{\tau_s}{\tau_{fus}}f_b(v,\zeta,t) + \frac{\tau_s}{\tau_{terhm}}f_b(v,\zeta,t) + \tau_s S^i(v,\zeta,t) \quad (152)$$

where the azimutahl dependence of the fast ions velocity has been integrated out due to assumed symmetry in that variable. Typically the charge exchange loss rate detrmined by (τ_{cx}), the loss rate due to fast ion fusion (τ_{fus}), and especially the explicit loss rate due to thermalization (τ_{therm}), are neglected Ref[]. We are interested in the solution of this equation when the source term has a (possibly repetitive) pulsed time dependance of the form

$$S^i(v,\zeta,t) = \frac{\dot{S}_0^i}{v^2}\delta(v-v_0)\delta(\zeta-\zeta_0)\Big(H(t-t_0^i) - H(t-t_1^i)\Big) \quad (153)$$

which is turned on and off at times t_0^i, t_1^i (H is the Heavyside step function). The resulting fast ion distribution function due to this source is

$$f_b^i(v,\zeta) = \frac{\dot{S}_0^i \tau_s}{v^3+v_c^3}P_{tot}(v)\sum_{l=0}^{\infty}\frac{2l+1}{2}P_l(\zeta)P_l(\zeta_b)\left[\frac{v^3}{v_b^3}\left(\frac{v_b^3+v_c^3}{v^3+v_c^3}\right)\right]^{\frac{1}{6}l(l+1)Z_2}$$
$$\left[H\Big(t-t_0^i-\tau_0(v_b)+\tau_0(v)\Big) - H\Big(t-t_1^i-\tau_0(v_b)+\tau_0(v)\Big)\right] \quad (154)$$

Where $\tau_0(v)$ is the time required for a fast ion of speed v to thermalize (ie to reach speed v_t).

$$\tau_0(v) = \frac{1}{3}\tau_s \log\left(\frac{v^3+v_c^3}{v_t^3+v_c^3}\right) \quad (155)$$

and $P_{tot}(v)$ is the total loss rate due to charge exchange,fast ion fusion and thermalization. In what follows we assume that fast ion interactions are negligible(however beam-beam fusion is ??) so that we may use the linear superpostion of solutions from various sources to get the total fast ion distribution function at any given time:

$$f_b^{tot}(v,\zeta) = \sum_i f_b^i(v,\zeta) \quad (156)$$

In order to use these results in a thermal transport code we form various moments of the distribution function with the collision operators. For each individual pulse we are interested in the following moments.
The fast ion density :

$$n_b^i(t) \equiv \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} f_b^i(v,\zeta) v^2 dv d\zeta \tag{157}$$

The fast ion stored energy density :

$$E_b^i(t) \equiv \frac{1}{2} m_b \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} v^2 f_b^i(v,\zeta) v^2 dv d\zeta \tag{158}$$

The power desnity delivered to the electrons:

$$Q_e^i(t) \equiv \frac{1}{2} m_b \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} (\frac{\partial}{\partial v} v^3 f_b^i(v,\zeta)) v^2 dv d\zeta \tag{159}$$

The energy delivered to the ions:

$$Q_i^i(t) \equiv \frac{1}{2} m_b \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} (\frac{\partial}{\partial v} v_c^3 f_b^i(v,\zeta)) v^2 dv d\zeta \tag{160}$$

The fast ion momentum transfer to electrons:

$$n_b^i(t) \equiv \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} f_b^i(v,\zeta) v^2 dv d\zeta \tag{161}$$

The fast ion momentum transfer to ions:

$$n_b^i(t) \equiv \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} f_b^i(v,\zeta) v^2 dv d\zeta \tag{162}$$

The fast ion fusion rates:

$$n_b^i(t) \equiv \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} f_b^i(v,\zeta) v^2 dv d\zeta \tag{163}$$

The fast ion charge exchange rate:

$$n_b^i(t) \equiv \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} f_b^i(v,\zeta) v^2 dv d\zeta \tag{164}$$

The fast ion thermalization rate:

$$\dot{S}_{th}^i = \dot{S}^i - \dot{L}^i - \frac{\partial n_b^i}{\partial t} \tag{165}$$

Here $\dot{L}^i$ is the loss rate of fast ions due to all factors except thermalization:

$$L^i(t) = \int_{v^i_{min}(t)}^{v^i_{max}(t)} \int_{-1}^{1} f^i_b(v,\zeta) P(v) v^2 dv d\zeta \tag{166}$$

The reason Eq.[165] is written in this particular form rather than in terms of a more fundamental definition such as

$$\dot{S}^i_{th} = \lim_{\Delta t \to 0} \frac{\int_{v_{th}(t)}^{v(t+\Delta t)} \int_{-1}^{1} f^i_b(v,\zeta) v^2 dv d\zeta}{\Delta t} \tag{167}$$

is that the formulation of the Fokker-Planck equation [152] is not valid near thermal energies and hence the form of f_b is not adequate to evaluate eq.[167]. On the other hand Eq.[167] consists of quantitites that depend on the slowing down distribution at higher energies and hence yields a reasonable approximation. We note that the time delay associated with the appearance of thermalized fast ions is properly given by this approach. For example during the linear (in time) rise of the stored fast ion density that occurs when the loss rate is zero the time derivative term cancels the source term initially, leading to a thermalization rate of zero until the derivative term changes.

In these equations both the upper and lower limits of integration are functions of time due to the transient nature of the source:

$$v_{max}(t) = v^i_b \qquad t^i_0 \le t \le t^i_1 \tag{168}$$

$$= [((v^i_b)^3 + v^3_c)\exp(\frac{-3(t-t^i_1)}{\tau_s}) - v^3_c]^{\frac{1}{3}} \quad t^i_1 \le t \le t^i_1 + \tau_0(v_b) \tag{169}$$

$$= v_t \qquad t \ge t^i_1 + \tau_0(v_b) \tag{170}$$

$$v_{min}(t) = [((v^i_b)^3 + v^3_c)\exp(\frac{-3(t-t^i_0)}{\tau_s}) - v^3_c]^{\frac{1}{3}} \quad t^i_0 \le t \le t^i_0 + \tau_0(v_b) \tag{171}$$

$$= v_t \qquad t \ge t^i_0 + \tau_0(v_b) \tag{172}$$

The upper limit of integration remains fixed at the fast ion birth speed v^i_b until the source $\dot{S}^i_0$ is shut off. Thereafter this upper limit decreases since ions of speeds greater than v_{max} are not replenished by the source. The lower limit starts at the same value as the upper limit and decreases as time goes on until it reaches an arbitrarily defined thermal cutoff value v_t. If the source is on long enough ($t^i_1 - t^i_0 \ge \tau_0(v_b)$) the moments become time independent when the lower limit of integration reaches the value v_t at time $t_0^i + \tau_0(v_b)$ (typically v_t is taken as 0.0, but in Onetwo it is user specifiable). Additionally,since we are neglecting losses during the slowing down process,n_b^i may also display a time independent plataue when the pulse length, $t_1^i - t_0^i$,is shorter than $\tau_0(v_b)$. In this case both $v_{max}(t)$ and $v_{min}(t)$ are decreasing at the same constant rate so that the density doesnt start to change until $v_{min}(t) = v_t$. Thereafter the contribution to the density made by the integral, Eq.[173] decreases steadily as $v_{max}(t)$ approaches v_t.

Toroidal plasma roation is taken into account by

6.2 Interpretation of results

By neglecting charge exchange,fusion and explicit thermalization losses the fast ion density has a simple solution useful for demonstration purposes:

$$n_b^i(t) \equiv \int_{v_{min}^i(t)}^{v_{max}^i(t)} \int_{-1}^{1} f_b^i(v,\zeta) v^2 dv d\zeta \qquad (173)$$

The relationship between the beam slowing down time and the on/off switching frequency of the beam sources,coupled with the superposition of sources from different beams and energy components can produce complex waveforms. Furthermore the fundamental time step dt that the transport equations are solved with is typically dynamically adjusted and could lead to innapropriate sampling times for the neutral beams,even if the user is careful to pick consistent times initally. A final complication is that the starting time of the transport simulation is typically not the same as the initial startup of the beams. Thus an initial condition for neutral beam related quantities must be generated from the user input. These issues are addressed in this section and some simple examples are presented.

The situation described above is depicted in Fig.[6]. Here we show a number of individual pulses of duration 60 msec, spaced 10 msec appart.The beam slowing down time is approximately $\tau_0(v_b) = 90ms$. Due to the assumption of linearity the total fast ion density in the figure is the linear superpostion of the individual responses to each beam pulse. As shown, up to three individual pulses combine to form the resultant fast ion density for this hypothetical case. Since the pulse length is less than the slowing down time individual responses never reach the steady state value Eq.[173] ($90ms * 1x10^{20}\frac{1}{cm^3 s} = 9x10^{18}\frac{1}{cm^3}$). As a consequence the individual responses (aa,bb,cc,...) have a flat top (plataue) value because the density is not changing as the group of fast ions in the range $(v_{max}(t), v_{min}(t))$ slow down as was explained above. The individual responses such as the one labeled aa in he figure are the result of evaluating the integral in Eq.[173] in

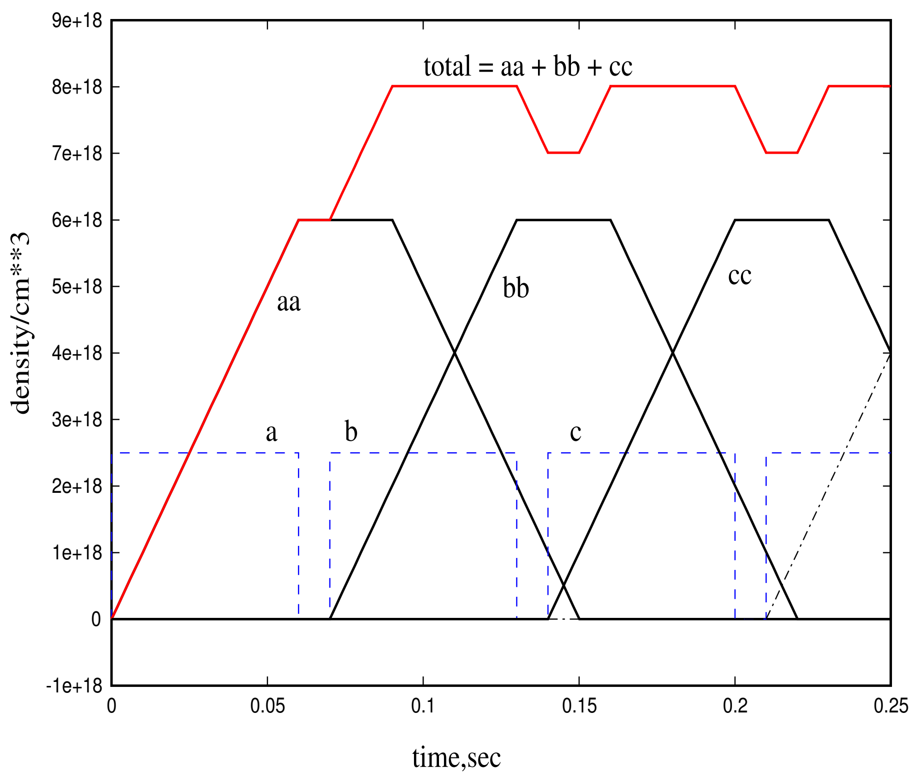


Figure 6: Example Waveform From Neutral Beam Deposition. The square wave,composed pulses a,b,c,etc., are the beam switching times in arbitray units. With the given slowing down time the resulting fast ion density waveform (total) is due to the summation of up to three individual components,aa,bb,cc due to the pulses

accordance with the limits of integration given above:

$$\begin{aligned} \boldsymbol{n_b^i(t)} &= \boldsymbol{\dot{S}_0^i(t - t_0^i)} & \boldsymbol{t_0^i \le t \le min(t_1^i, \tau_0(v_b))} && (174)\\ &= \boldsymbol{\dot{S}_0^i(\tau_0(v_b) - t_0^i)} &&& (175)\\ &= \boldsymbol{\dot{S}_0^i(t_1^i - t_0^i)} &&& (176)\\ &= \boldsymbol{\dot{S}_0^i(t_1^i - t_0^i)} &&& (177)\\ &&&& (178) \end{aligned}$$

When the pulse length is increased so that the fast ion density reaches its fully developed state before the pulse is turned off we have the situation shown in fig ??.

It should be noted that time step control must be asserted by these computations. Unfortunately it is not enough to simply enforce the rule that each source switching time be observed exactly by the time stepping routines that advance the transport equations. Two additional time values must be ex-

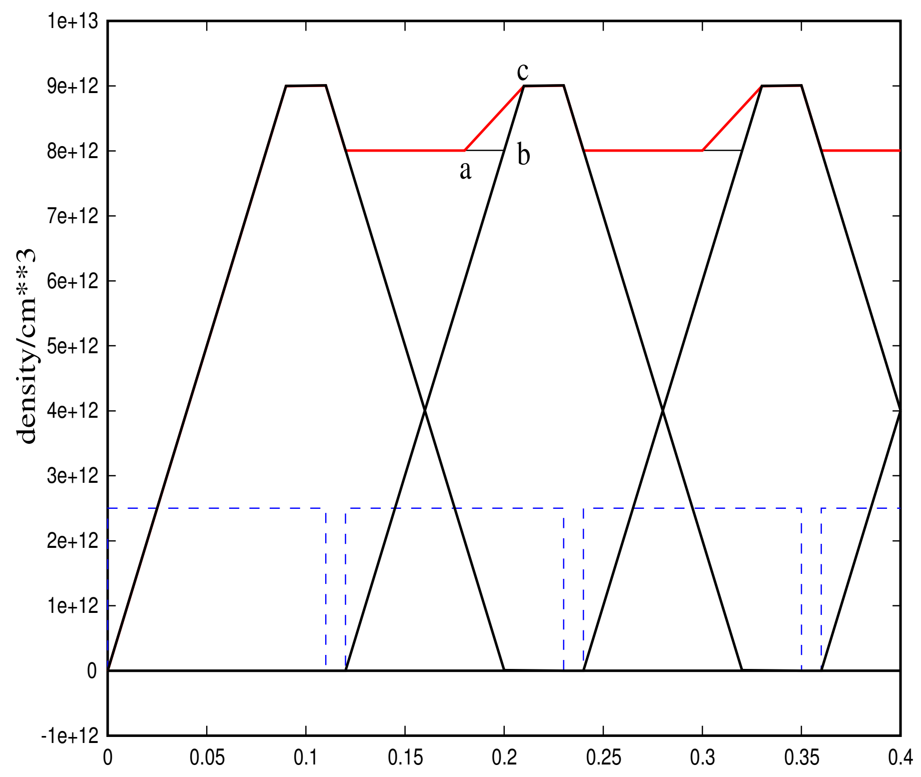


Figure 7: Example Waveform From Neutral Beam Deposition. The square wave(with pulses a,b,c is the beam switching times in arbitray units. The two resulting fast ion density waveforms are due to different time steps used in the calculations

plicitly enforced. The first,t_2^i, is when $vmin(t_2^i) = v_t$. The second,t_3^i, is when $n_b^i(t_3^i)$ reaches zero. Unlike the beam switching times (t_0^i, t_1^i) the set of values (t_2^i, t_3^i) is not amenable to prediction on an apriori basis. Instead these times are determined as the calculations proceed.

Finally we remark that the computations are designed to accept an arbitrary time step dt. Where the signficance of dt is that the state of the system is to be updated from time t to time $t_{new} = t + dt$. However, on return from the modules that perform the computations described above, it is not guaranteed that $t_{new} = t + dt$. In fact the new time will be given by

$$t_{new} = t + min(t_j^i - t, dt) \tag{179}$$

where only those values for which $t_j^i - t > 0.0$ are to be considered,$j = 0, 1, 2, 3$ and i ranges over all pulses of each source of each beam. Since there is no apriori assumed relationship between the phasing of various beam lines and sources it may become quite tedious to advance the transport equations if the time step is controlled by Eq.[179] rather than by other physics. To avoid this situation times $\boldsymbol{t_3^i}$ and/or $\boldsymbol{t_2^i}$ may optionally be ignored. This will result in some details of the waveforms being incorrect. However transport results are not necessarily affected since the generated waveform will be correct at the times that output is requested. For example,consider Fig[**??**] . This case is similar to the previous one but has a longer pulse length $\boldsymbol{(110ms)}$ and hence the fast ion density reaches its saturated value before the beam turns off. In the figure we show the exact solution as well as two approximate solutions where $\boldsymbol{t_3^i}$ and where both $\boldsymbol{t_3^i andt_2^i}$ are ignored.

Special waveforms are possible since there is no assumption about the relationship of sources for a given beam or between beams. For example in Fig[8] we have a single beam line with two sources. The first source produces

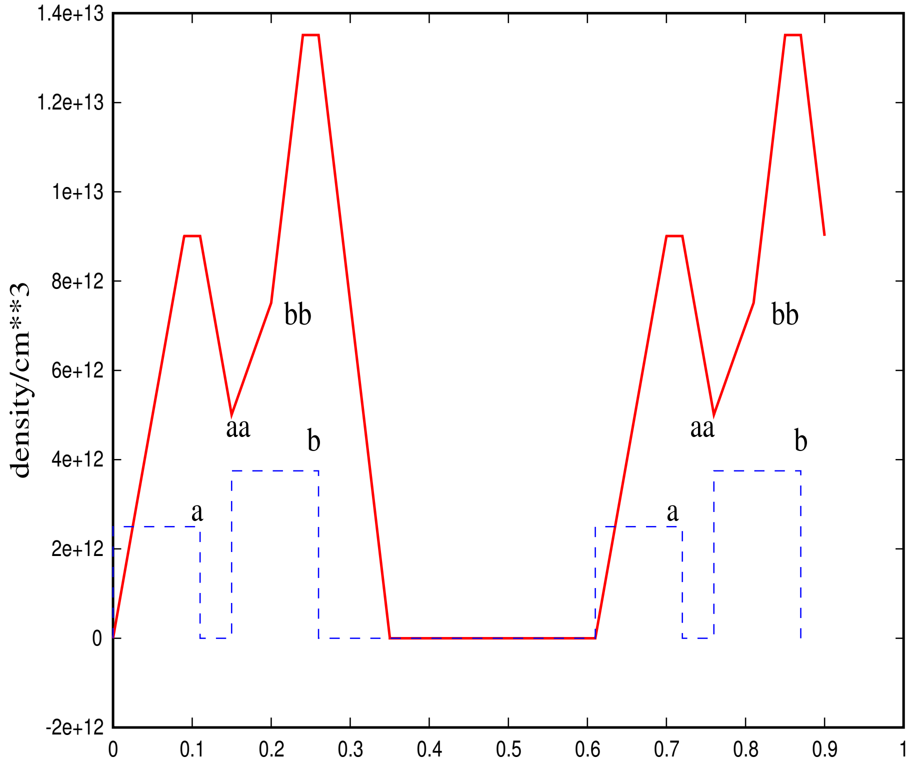


Figure 8: Example Waveform From Neutral Beam Deposition. Two independent sources produce wavetrains with pulses a and b respectively. The resulting fast ion density does not have a plateau value (aa-bb) due to the different intensities of the sources.

pulse train a and the second source pulse train b. The second source has 1.5 times the intensity of the first source. The resulting fast ion density has a rsing region (aa-bb) instead of a plateau from (aa-bb) due to the difference in source strengths.

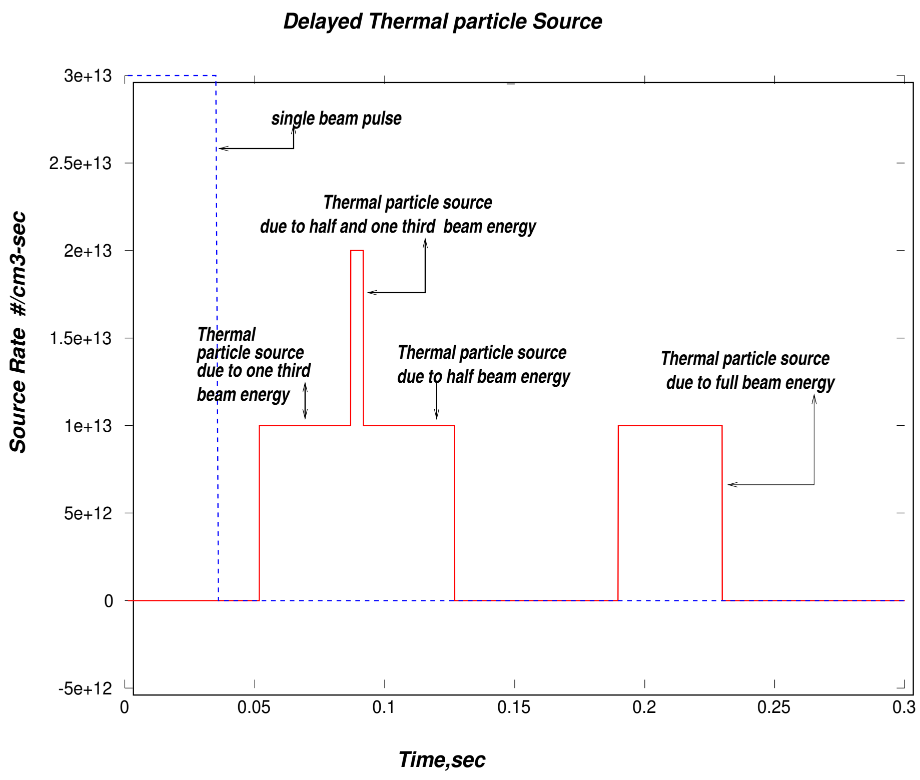


Figure 9: Illustration of the delayed particle source due to a single beam pulse that starts at 1 msec and ends at 41 msec.The delayed sources are due to the different slowing down times of the three beam energy components.

6.3 Beam Initial Conditions

Suppose that for the situation shown in Fig[8] we wish to start our analysis at time $\boldsymbol{t = 0.5sec}$ Considering only the fast ion density we see that it is in fact possible to assume that the beam was not on prior to $\boldsymbol{t = 0.5sec}$ because its influemce has decayed away. Unfortunately the general situation is not so simple. Suppose instead that we want to start the transport analysis at time $\boldsymbol{t = 0.28sec}$ in Fig[8]. In this case the beam history must be taken into account for time prior to the start of the analysis. Furthermore automatic corrections in the mhd/transport coupling of Onetwo can result in arbitrary time rollbacks to previous solution points. Thus we are forced to generate and periodically update some sort of beam history file as part of the solution scheme. The proper way to generate an intial history file depends on how the code is run however. For example in tdem mode the user will expect that the tdem information is used in creating the file. In a non tdem run other options have to be pursued.

7 Solution of Faraday's Law: Some obervations

The solution of Faraday's and Ohm's law places restrictions on the problem which may preclude a steady state solution from existing in some of the restrictred modeling that can be done with Onetwo. The effect is most easily demonstrated when the pressure profiles and the mhd equilibirum are keept fixed in time but the current density is allowed to evolve to steady state. In Fig10 we present an example where a steady state solution (eg a solution with a flat parallel electric field) could not be found with the assumed given total toroidal current of 10 MA. A simple test, keeping the equilibrium constant, but simply varying the total current, yields the results shown in the figure. Ssq represents the residual of Faraday's law. A true solution of the equation will have ssq $\approx \mathbf{0}$. As is seen the solution approaches a zero residual value only if the total current is increased to about 13.5 MA. From the smoothnes of the curve it is concluded that the non-linear solver is in fact finding the best solution in each case (otherwise we would expect the curve the have a fluctuating behavior). For each value of the total current the solver is finding the best approximate solution in the sense that the residual of Faraday's law is minimized.

The reason why we can not obtain a valid solution until the total current is increase to 13.5 Ma is due to the unique nature of the bootstrap current. Inside of $\boldsymbol{\rho} \approx \mathbf{0.25}$ the ohmic current is required to be positive Near the magnetic axis the bootstrap current forces the

The above results apply to the $\boldsymbol{Iter_feat}$ case with 6kev pedestal temperature. It is interesting to note that a similar case with 5kev pedestal temperature does not display this behavior. In Fig(11) we show the GLF23 evolved steady state temperatures associated with these two cases. Note the gradient in te near $\boldsymbol{\rho} = \mathbf{0.75}$ in the 6kev case.

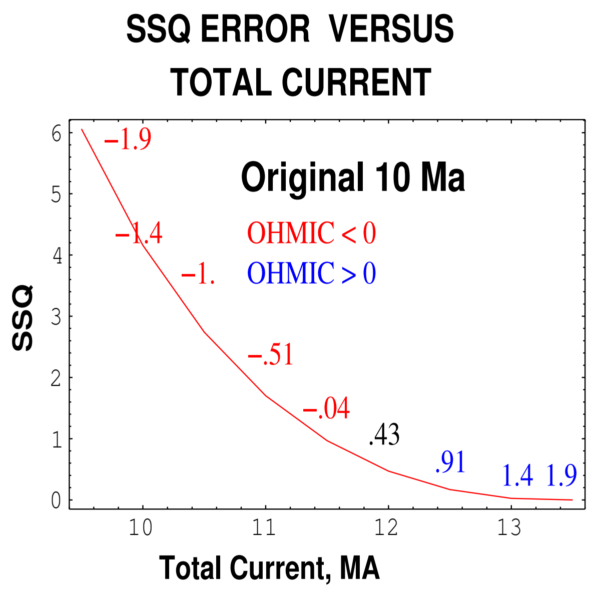
Shot 84293
(L mode)

FW MHz	P_e, kW	P_i, kW	I_{CD}, kA	P_a
60	501	499	56	41 %
60	491	509	51	80 %
83	744/607	256/398	80/71	31%
83	728	272?	74	80%
117	950	50	111	29%
117	895	105	99	82%

Shot 111221
(H mode)

FW MHz	P_e, kW	P_i, kW	I_{CD}, kA	P_a
60	340/287	660/773	22/16	95%
83	420	580	30	99%
117	395/212	605/788	30/18	99%

Table 1: Heating and Current drive results for DIII-D L and H mode cases.The second number for electron,and ion absorbed power and current drive indicates results obtained using Transp profiles. P_a is the percent of injected power absorbed. For the L mode shot results are quoted for 6 and 100 edge reflections (with the larger value of P_a corresponding to 100 edge reflections and the smaller value to 6 reflections)



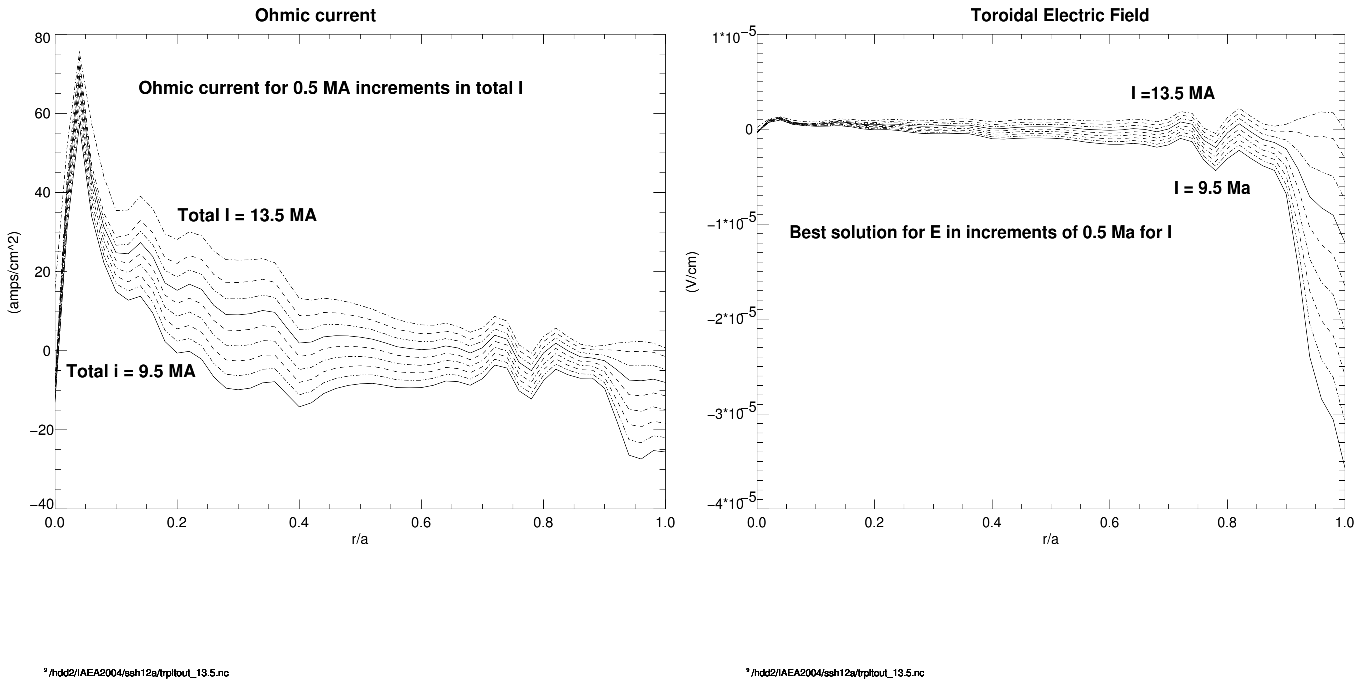
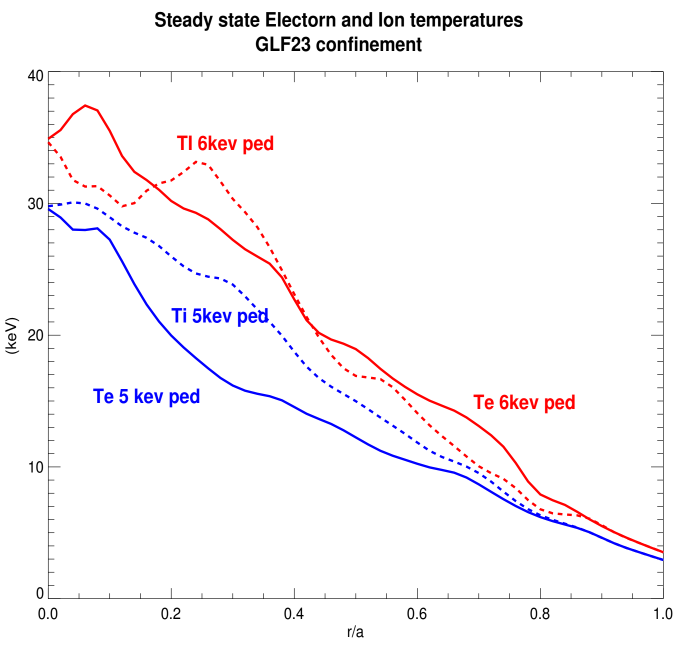
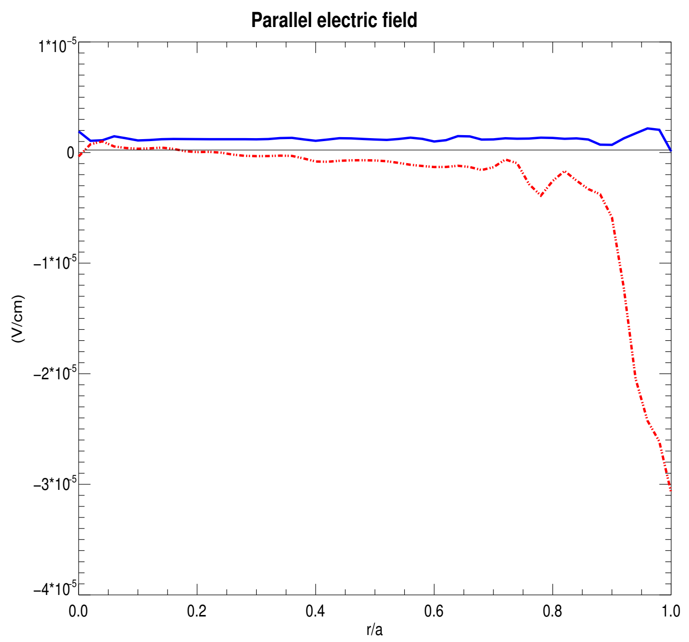


Figure 10: (a)Increasing the total current causes the non linear solver to smoothly approach the solution with zero residual. This is indicative of the fact that the solver is finding the best,approximate, “solution” in each instance. Part (b) and (c) show the corresponding ohmic current profiles and electric fields





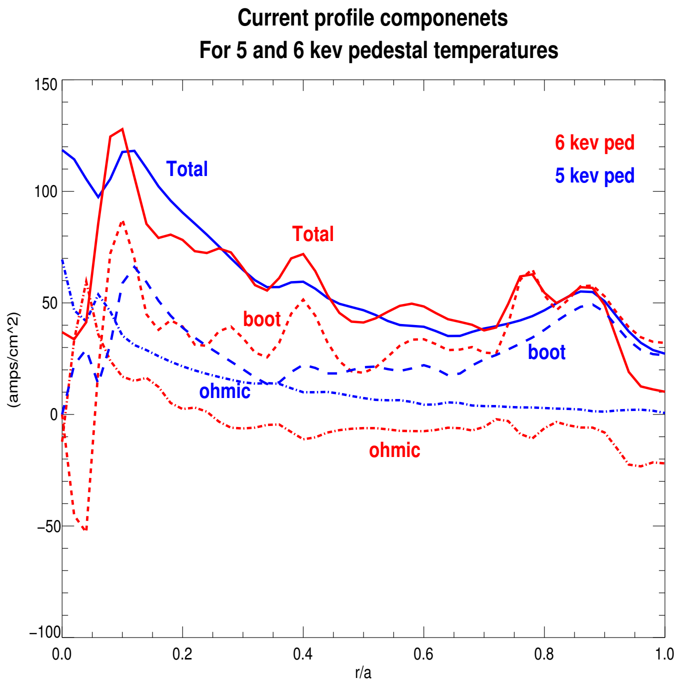


Figure 11: (a)The steady state temperatures obtained with GLF23 for the 5 and 6 kev pedestal cases. (b) The corresponding electric field. The electric field for the 6 kev case is the one shown in Fig(10) for the 10 MA case and is not a true solution as explained in the text. (c) The corresponding current density profiles.For simplicity the beam and rf driven current are not shown. The 6 kev pedestal case is the same as the 10 MA case in Fig(10).

A Some Mathematical Details

The element of area on the surface of the toroidal plasma is $2\pi Rdl$ where dl is an element of length along a given (cross sectional) psi contour. The total plasma surface area is thus

$$S(\rho) = \oint 2\pi Rdl = 2\pi \oint \frac{B_p Rdl}{B_p} \tag{180}$$

The flux surface average operation is defined,for an arbitrary function A, as

$$< A >\equiv \frac{\oint \frac{Adl}{B_p}}{\oint \frac{dl}{B_p}} = \frac{\oint \frac{Adl}{B_p}}{\frac{1}{2\pi}\frac{\partial V}{\partial \psi}} \tag{181}$$

Where the rate of change of plasma volume with respect to flux surface label ψ, $\frac{\partial V}{\partial \psi}$, has been introduced. Hence we can write equation(180) as

$$S(\rho) = \left(\frac{\partial V}{\partial \rho}\right)\left(\frac{\partial \rho}{\partial \psi}\right) < B_p R >= 4\pi^2 R_0 H\rho \frac{\partial \rho}{\partial \psi} < B_p R > \tag{182}$$

Where H is defined as

$$H \equiv \frac{\frac{\partial V}{\partial \rho}}{4\pi^2 R_0 \rho} \tag{183}$$

The flux surface average poloidal B field is defined as

$$B_{p0} \equiv \frac{1}{R_0}\frac{\partial \psi}{\partial \rho} \tag{184}$$

so equation (182) becomes

$$S(\rho) = 4\pi^2 R_0 H\rho \left\langle \frac{RB_P}{R_0 B_{P0}} \right\rangle \tag{185}$$

Since $\nabla\rho = \frac{\partial \rho}{\partial \psi}\nabla\psi$ and $B_P = \frac{|\nabla\psi|}{R}$ we can also write the surface area as

$$S(\rho) = 4\pi^2 R_0 H\rho <| \nabla\rho |> \tag{186}$$

By the definition of H this can also be expressed as

$$S(\rho) = \frac{\partial V}{\partial \rho} <| \nabla\rho |> \tag{187}$$

Note that $S(\rho)$ represents the true physical surface area of the nested flux surfaces.

Quantities that have to be integrated over the cross sectional area of the flux tubes,such as the toroidal current density $\boldsymbol{J_\phi}$, are developed as follows:

$$\int J_\phi dA = \int J_\phi(R,Z)dRdZ = \int J_\phi \frac{d\Psi dl}{RB_p} \tag{188}$$

where we have used $dRdZ = \frac{d\Psi dl}{RB_p}$. This result can be manipulated into the flux surface average expression

$$\int J_\phi dA = 2\pi \int \left\langle \frac{J_\phi(R,Z)R_0}{R} \right\rangle H\rho d\rho \tag{189}$$

by using $\frac{dV}{d\Psi} = 2\pi \int \frac{dl}{B_p}$.

A.1 Flux Surface Averaging of Diffusion Equations

As an example of how flux surface averaging proceeds consider the electron energy balance equation. It can be written in the form

$$\frac{3}{2}\frac{\partial P_e}{\partial t} + \nabla \cdot \left(Q_e + \frac{5}{2} P_e V_e \right) = S + \vec{J} \cdot \vec{E} - Q_\delta + V_i \cdot \nabla P_i \tag{190}$$

If we apply the flux surface averaging operation to this equation and multiply by V' we get

$$\frac{3}{2}V' \left\langle \frac{\partial P_e}{\partial t} \right\rangle + V' \left\langle \nabla \cdot \left(Q_e + \frac{5}{2} P_e V_e \right) \right\rangle = V' \langle rhs \rangle \tag{191}$$

An easily established property of the flux surface averaging is that

$$\left\langle \nabla \cdot \vec{A} \right\rangle = \frac{1}{V'} \frac{\partial}{\partial \rho} \left[V' \langle A \cdot \nabla \rho \rangle \right] \tag{192}$$

Applying this result to the divergence term in equation(191) we obtain

$$\frac{3}{2}V' \left\langle \frac{\partial P_e}{\partial t} \right\rangle + \frac{\partial}{\partial \rho} \left(V' \left\langle \left(Q_e + \frac{5}{2} P_e V_e \right) \cdot \nabla \rho \right\rangle \right) = V' \langle rhs \rangle \tag{193}$$

Now use

$$V' \left\langle \frac{\partial A}{\partial t} \mid_{R,Z} \right\rangle = \frac{\partial}{\partial \rho} \mid_t \left(V' \langle A \vec{u_\rho} \cdot \nabla \rho \rangle \right) \tag{194}$$

to write equation(193) in the form

$$\frac{3}{2}\frac{\partial}{\partial t} \mid_\rho (V' P_e) - \frac{3}{2}\frac{\partial}{\partial t} \mid_t \left(V' \langle P_e \vec{u_\rho} \cdot \nabla \rho \rangle \right) +$$

$$\frac{\partial}{\partial\rho}\mid_t \left(V' \left\langle \vec{Q}_e\cdot\nabla\rho\right\rangle + \frac{5}{2}\left\langle V' P_e V_e\cdot\nabla\rho\right\rangle\right) = V'\left\langle rhs\right\rangle \tag{195}$$

The important definition is that of the flux surface average energy flux,which as seen from the above equation should be defined as

$$q_e \equiv \langle Q_e\cdot\nabla\rho\rangle \tag{196}$$

The conductivity,K_e, can depend on more than just one space dimension. It is defined as the constant of proportionallity relating the flow to the gradient:

$$\vec{Q}_e = -K_e\nabla T_e \tag{197}$$

If we assume that T_e is a flux surface functions then we can write

$$-K_e\nabla T_e = -K_e\frac{\partial T_e}{\partial\rho}\nabla\rho \tag{198}$$

Hence

$$q_e = -\frac{\partial T_e}{\partial\rho}\left\langle K_e \mid\nabla\rho\mid^2\right\rangle \tag{199}$$

The actual amount of energy flowing out of any given surface is

$$\int\!\!\int \vec{Q}_e\cdot d\vec{A} = \frac{\partial V}{\partial\rho}q_e \tag{200}$$

The flow of energy out of any flux surface,as given by Eq200,is the quantity that has physical significance and must be conserved. Note however that the rhs of Eq[200]is not expressed in the customary form of the surface area times the energy flux flowing through that area. We can change Eq.[200] into this form by multiplying and dividing by the factor $<\mid\nabla\rho\mid>$:

$$\begin{aligned}\int\!\!\int \vec{Q}_e\cdot d\vec{A} &= <\mid\nabla\rho\mid> \frac{\partial V}{\partial\rho}\left(\frac{q_e}{<\mid\nabla\rho\mid>}\right) && (201)\\ &\equiv S(\rho)q_e^* && (202)\end{aligned}$$

Here $S(\rho)$ is the true physical area of the flux surface(see Eq.[187]) and q_e^* is the average flux through this surface. Some codes use this definition with the result that the divergence part of the diffusion equation looks like

$$\frac{1}{V'}\frac{\partial}{\partial\rho}\bigg|_t \left(V' <\mid\nabla\rho\mid> q_e^* + \cdots\right) \tag{203}$$

As is obvious from examination of the equations in section two,Onetwo does not use the convention given by Eq.[203]. Instead Onetwo cancels the common factor of $<|\nabla\rho|>$ from $S(\rho)$ and q_e^* and uses q_e as an effective flux. This flux must then be multiplied by an effective surface area, $\frac{\partial V}{\partial\rho}$, in order to maintain the proper energy flow as given by Eq.[200]. Internally the code calculates the conductive flux from the expression

$$q_e = -\kappa_e \frac{\partial T_e}{\partial\rho} \tag{204}$$

Hence for the anomalous transport models the user must supply a definition of κ adjusted so that the physical flow of energy is given by Eq.[200].

As an example we have the IFS anomalous conductivity model ??] which states that the conduction of energy out of any given flux surface is the lhs of the following equation:

$$\left(-n\chi_{IFS}\frac{\partial T_e}{\partial\rho}\right)V' <|\nabla\rho|> = \frac{\partial V}{\partial\rho}q_e \tag{205}$$

The rhs of this equation follows from the physical condition, Eq.[200] and gives us the definition of κ, Eq.[204] (or χ), that should be used in Onetwo:

$$\chi_{Onetwo} = \chi_{IFS} <|\nabla\rho|> \tag{206}$$

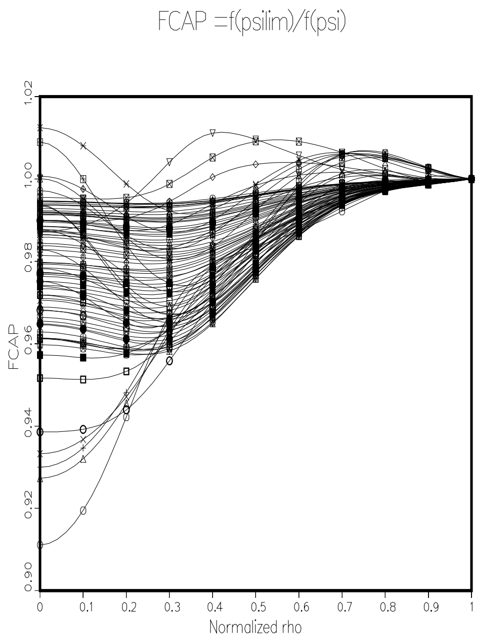
The effective flux put out by Onetwo will be defined so that it must be multiplied by V' (and not the true surface area $V' <|\nabla\rho|>$) to get the true flow as illustrated with the following sequence of equations:

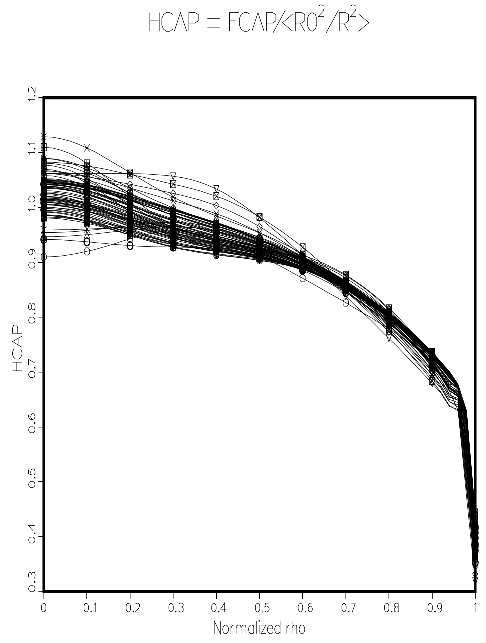
$$\begin{aligned}
q_{eOnetwo} &= -n_e\chi_e\frac{\partial T_e}{\partial\rho} && (207)\\
&= -n_e\chi_{IFS} <|\nabla\rho|> \frac{\partial T_e}{\partial\rho} && (208)\\
&= q_{eIFS} <|\nabla\rho|> && (209)\\
q_{eOnetwo}V' &= q_{eIFS} <|\nabla\rho|> V' && (210)\\
&= q_{eIFS}S(\rho) && (211)
\end{aligned}$$

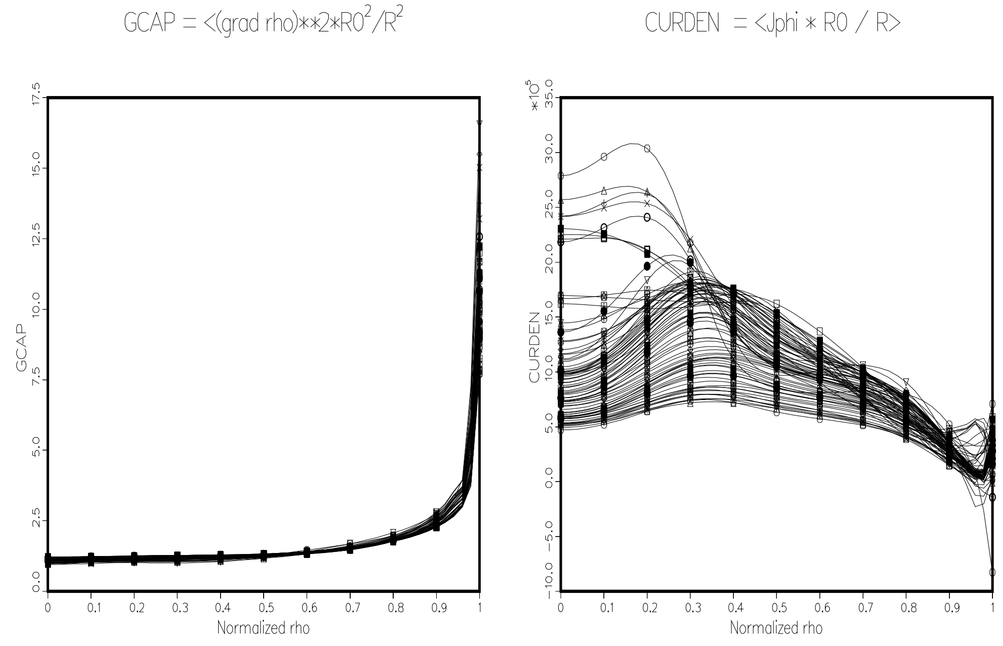
B An Example of TDEM Results

The time dependent eqdsk mode of operation was discussed in the main part of the text. Here we give some examples of the results obtained with this method,for the current drive scenario. The reader is reminded that the TDEM mode also applies(and in fact was constructed for) to confinement analysis. The application to current drive is quite subtle however so we present some details here.

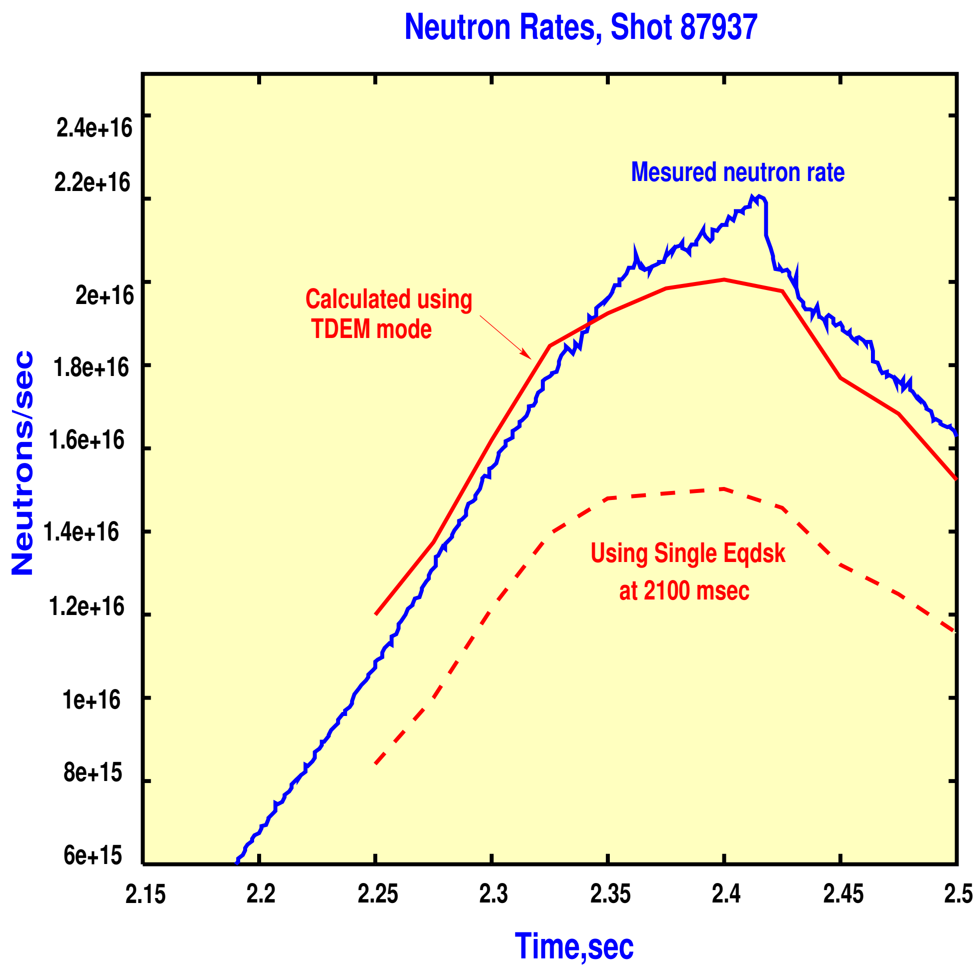
To begin with figures ? show the typical variation of the metric parameters $\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}$ and the parallel current density,Eq.[]. Each curve in these figures represents a particular time during the mhd evolution. The situation illustrated is for a negative shear case (shot 87953). The curves were generated using the (fortran) mepc code (this code can easily be modified to generate flux surface averages of various kinds so that the user does not have to start from scratch if new data is required).

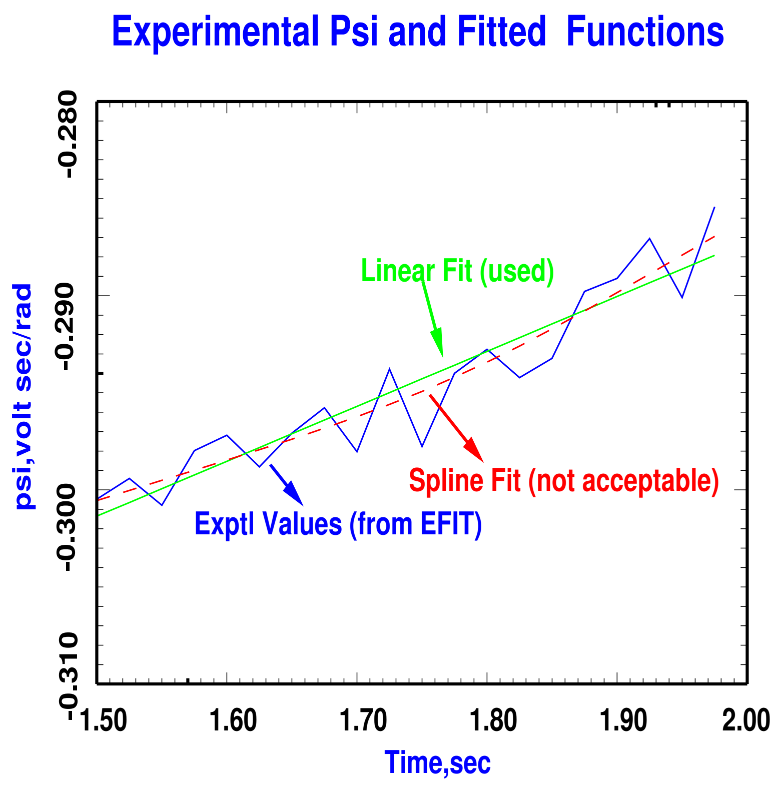




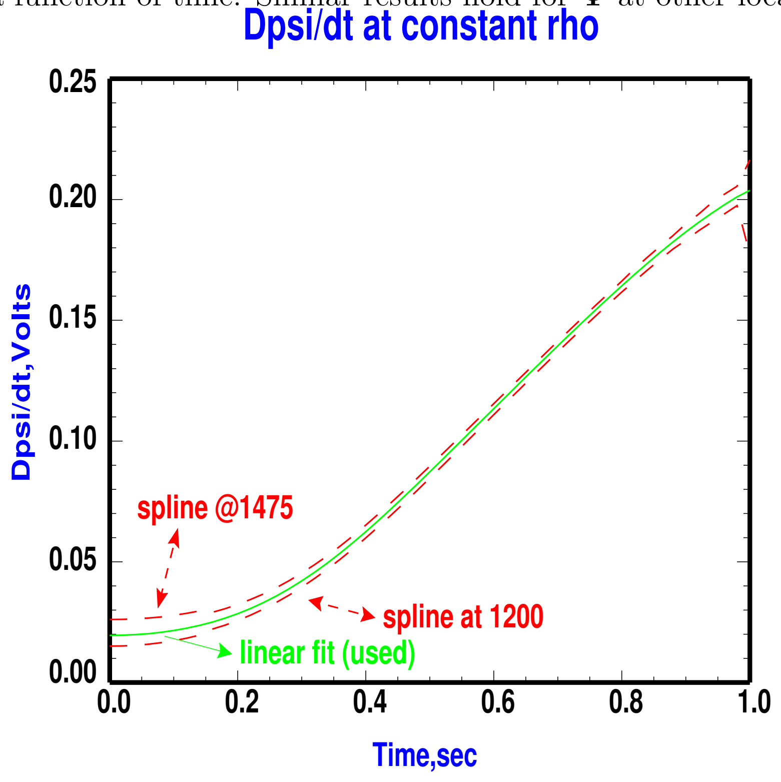


The variation in the metric parameters can have siginificant effects on the transport derived quantities. For example in Fig. [] we show the neutron rate calculated using the TDEM mode and the result of the same calculation when a single eqdsk is used.

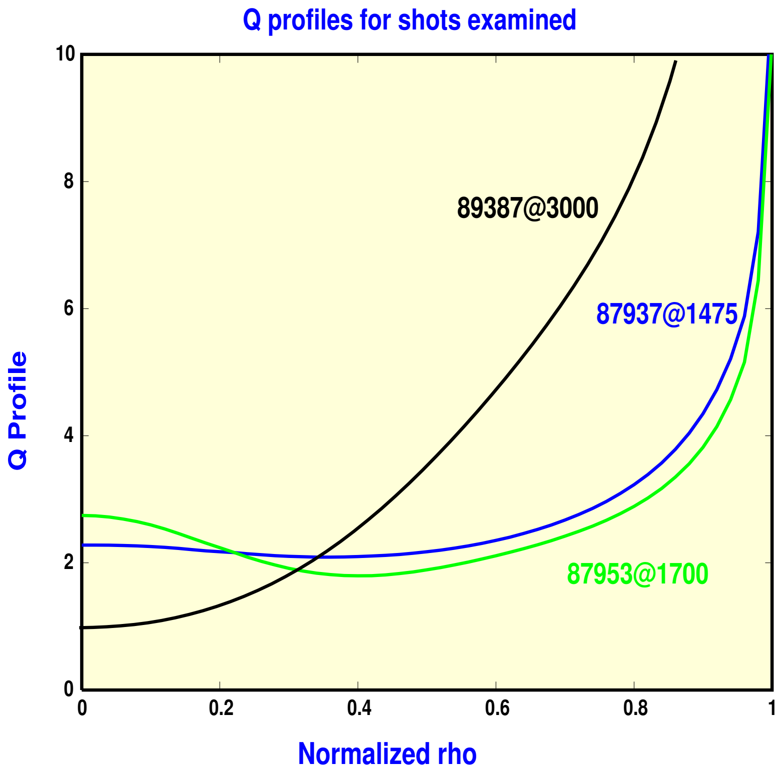

Neutron Rates, Shot 87937
Mesured neutron rate
Calculated using TDEM mode
Using Single Eqdsk at 2100 msec
Neutrons/sec
Time,sec
2.4e+16
2.2e+16
2e+16
1.8e+16
1.6e+16
1.4e+16
1.2e+16
1e+16
8e+15
6e+15
2.15
2.2
2.25
2.3
2.35
2.4
2.45
2.5



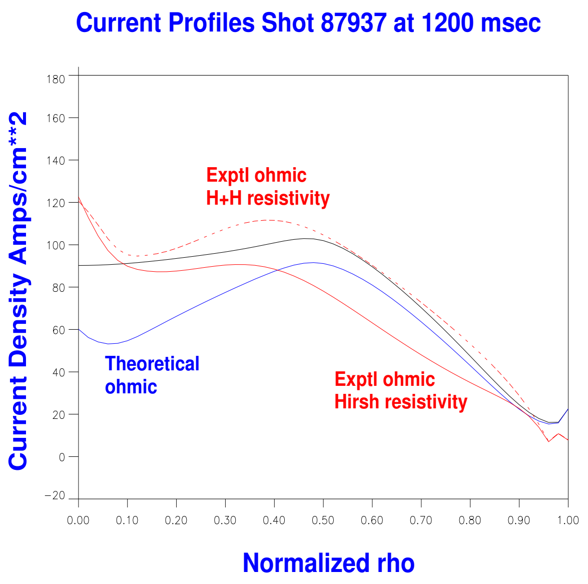
$\boldsymbol{\Psi}$,the linear and a possible spline fit used for $\frac{\partial \boldsymbol{\Psi}}{\partial t}$ Shown is the magnetic axis value as a function of time. Similar results hold for $\boldsymbol{\Psi}$ at other locations.

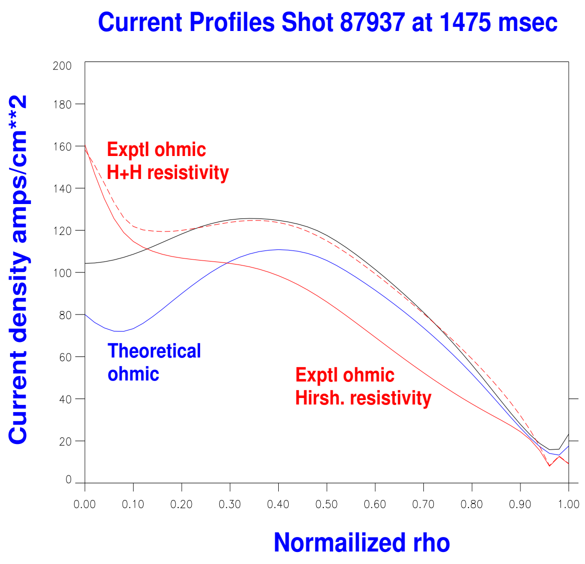


A number of beam heated shots ranging from strong negative shear (87953), weak negative shear (87937) and positive shear (89387,89388,89389) were examined using this method.

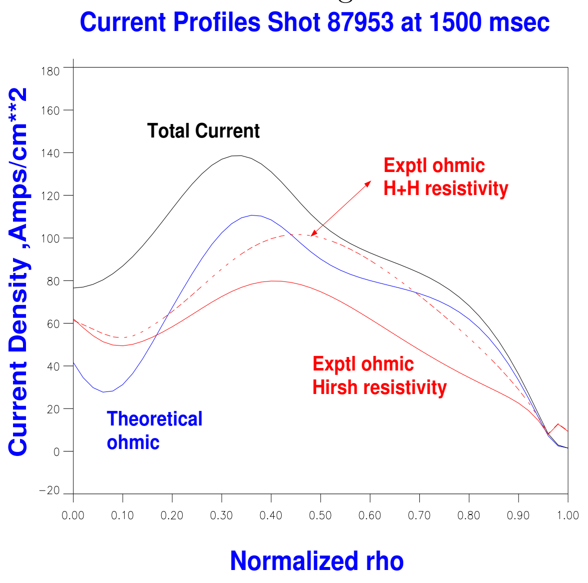


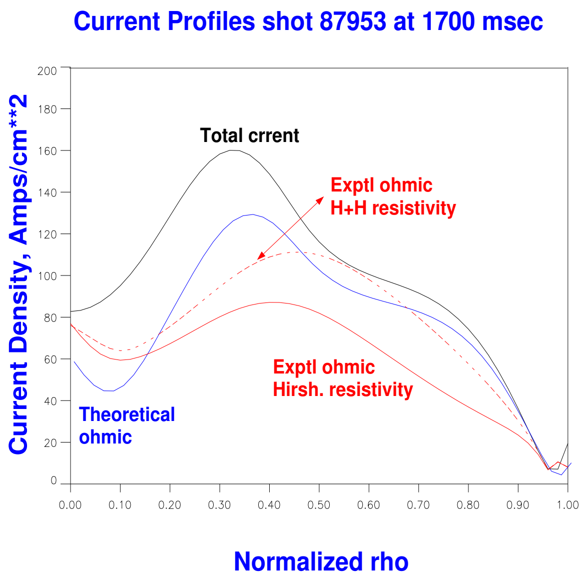
The object was to determine if this approach can be made to work in such a wide variety of cases. As seen in Eqs.[16] and [21] only the product of resistivity and ohmic current is determined by this approach. Consequently additional information is required in order to separate the product. In quiescent discharges the resistivity should be neoclassical. The sensitivity to the form of the neoclassical model resistivity used can be gauged by comparing the Hinton and Hirshman models of the effective resistivity . As is seen in the following figures strong negative central shear can be modeled about as well as the weak shear case . Both cases deviate from the theoretical value of the ohmic current significantly,despite the fact that a “well behaved section “ of psi was used .



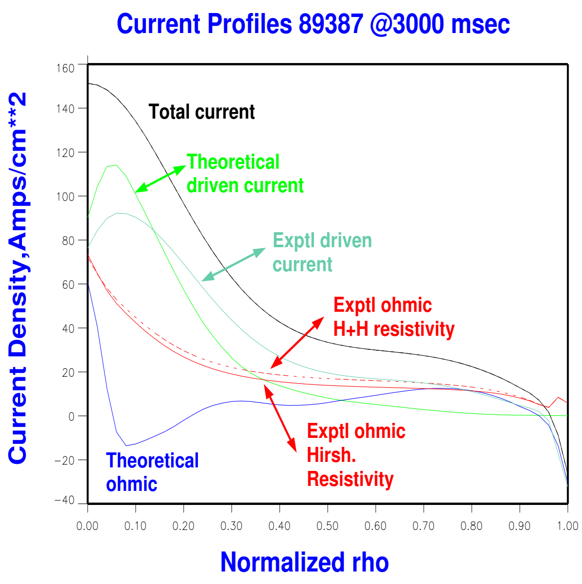


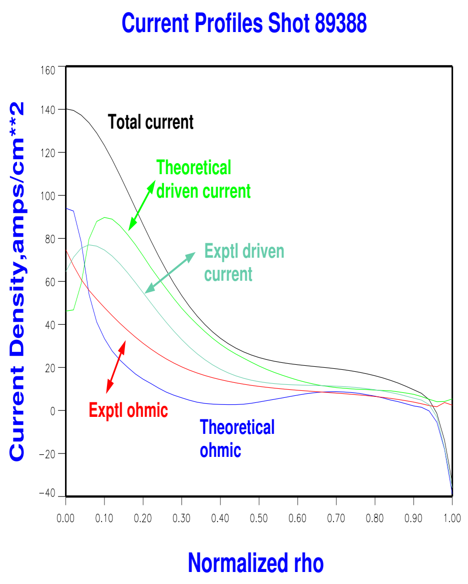
The results for the strong NCS case are :



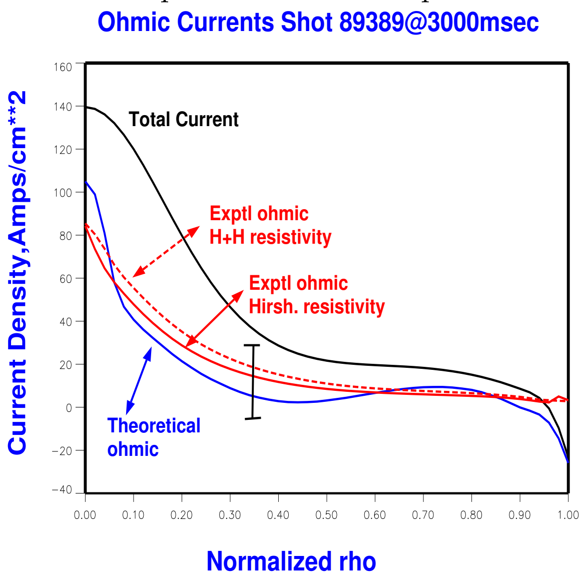


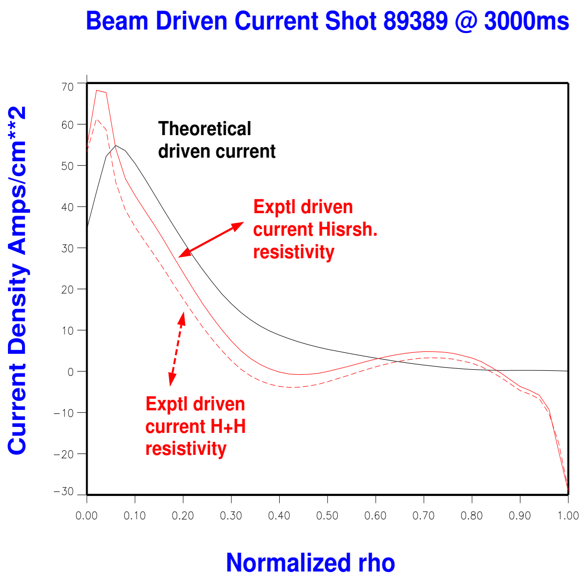
For the positive shear cases the comparison of experimental and theoretical current profiles is as follows:

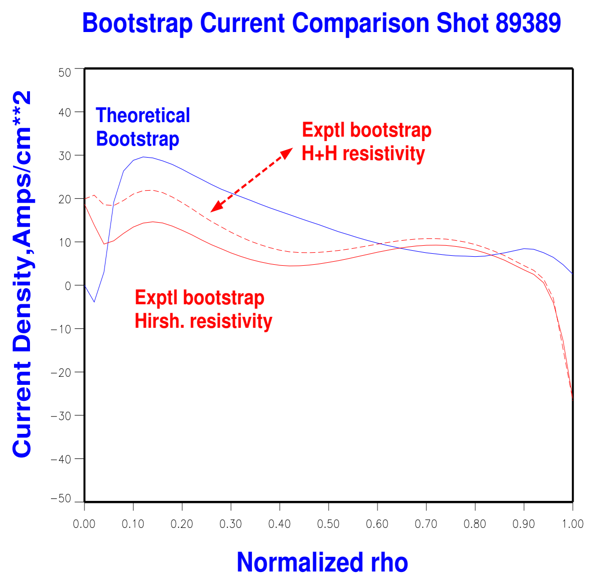




If,we use the experimental ohmic current together with the total parallel current (which is derived from the pressure and poloidal current functions in the mhd fit) the non inductive current may be estimated. If we also assume that the beam driven current is known then an experimental determination of the bootstrap current becomes possible:






Bootstrap Current Comparison Shot 89389
Theoretical Bootstrap
Exptl bootstrap H+H resistivity
Exptl bootstrap Hirsh. resistivity
Current Density,Amps/cm**2
Normalized rho
50
40
30
20
10
0
−10
−20
−30
−40
−50
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00

C Cubic Spline Review

With n knots,$[a_1, a_n]$, there are n-1 cubics to be determined. Each cubic has 4 unknowns so we need to find 4(n-1) unknowns altogether. A spline is defined by its function values at the knots, the continuity of the first and second derivatives at the interior knots and two boundary conditions. This arisesesas follows:

The value of the function for each of the n-1 cubics at the left and right ends of the interval in which the cubic is defined results in 2(n-1) equations, the continuity of the first derivative at the n-2 interior knots yields n-2 equations and we get another n-2 equations for the continuity of the second derivatve. Hence we have 2(n-1) +2(n-2) = 4n-6 equations . An addtional equation is given at the first knot and at the last knot to get a total of 4(n-1) as equired.

In practice the above procedure to find the equations for the spline coefficients is greatly simplified by taking advantage of the following general representtion of a cubic in an arbitrary interval. This representation already incorporates the function values at the knots and the continuity of the second derivative at the interior knots. For $x \in [a_{i-1}, a_i]$ a general cubic can be written in the form

$$c(x) = M_{i-1}\left(\frac{(a_i - x)^3}{6h_i} - \frac{h_i}{6.}(a_i - x)\right) + M_i\left(\frac{x - a_{i-1})^3}{6.h_i} - \frac{h_i}{6.}(x - a_{i-1})\right) + c_{i-1}\frac{a_i - x}{h_i} + c_i\frac{x - a_{i-1}}{h_i} \tag{212}$$

The four constants that define the cubic have been written in terms of the function values (c_{i-1}, c_i) and the second derivatives ,M_{i-1}, M_i at the end points of the interval. The knot spacing is given by $h_i = a_i - a_{i-1}$ and the index i takes on values from 2 to n (so there are n-1 intervals). In order to turn this representation of a cubic into a cubic spline representatstion of a function we have to consider n-1 contiguous intervals, with internal boundaries $a_2, ...a_{n-1}$ and edge(eg boundary) values at either end,$a_1 and a_n$. The above prescription allready satisfies the condition that the second derivative is continuos across the knots since, at each interior knot the right end of interval i is the same as the left end of inerval i+1. Hence we need only force the first derivative to be continuus at the interior knots. Applying this conditon at $x = a_2, ...a_{n-1}$ yields n-2 equations for the n unknonwsn,$M_1, ...M_n$.

$$\frac{h_i}{6}M_{i-1} + \frac{h_i + h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{c_{i-1}}{h_i} - c_i(\frac{1}{h_i} + \frac{1}{h_{i+1}}) + \frac{c_{i+1}}{h_{i+1}} \tag{213}$$

Note that Eq.[213] is valid only at the interior knots, $i = 2, ...n - 1$. Two additional equations are thus required to solve for the M_i uniquely. The general form of these two additional equations is given by Eq.1 specialized to the first and last intervals (i =2, and i = n)respectively). To cleanly blend with the IMSL splines routines these two equations are taken in the form

$$\begin{aligned} 2M_1 + bpar(1)M_2 &= bpar(2) \quad (214) \\ bpar(3)M_{n-1} + 2M_n &= bpar(4) \quad (215) \end{aligned}$$

To make sense out of these two equations we look at the cases that typically arise: a) the first derivative is given or b) the second derivative is constant. Usually one of (a) or (b) is applied at a_1 and, independently, one of (a) or (b) is also applied at a_n (the energy code allows for an additional alternative specification as a point of inflection. - This is not discussed here). Differentiating Eq.[212] and applying the results at a_1 and a_n we have:

$$\begin{aligned} 2M_1 + M_2 &= \frac{6}{h_2}(\frac{c_2 - c_1}{h_2} - d) \quad (216) \\ M_{n-1} + 2M_n &= \frac{6}{h_n}(e - \frac{c_n - c_{n-1}}{h_n} \quad (217) \end{aligned}$$

Here d is the desired first derivative at $x = a_1$ and e is the desired first derivative at $x = a_n$. Comparing Eqs [216,217] with Eqs[214,215] we can read off the definition of bpar required to set the first derivatives to the values d,e . This confirms the IMSL definition of Bpar.

The more mundane case of constant second derivative at the left and right ends is simply given by the fact that $M_1 = M_2$ and $M_{n-1} = M_n$. These equations must be written in the IMSL conformable way as:

$$\begin{aligned} 2M_1 - 2M_2 &= 0 \quad (218) \\ -2M_{n-1} + 2M_n &= 0 \quad (219) \end{aligned}$$

Hence we see that bpar(1) = -2 ,bpar(2) =0.0 and bpar(3) = -2, bpar(4) =0,will achieve this in Eqs[214,215]

Obviously there are other boundary conditions that could be fit into this scheme. To summarize, the cubic spline is defined by a set of n equations for the M_i. The first equation is taken as Eq[214], equations 2 to n-1 are of the form Eq[213], and the last (n'th) equation is Eq.[215]. Bpar suitably specialized yields the desired end point conditions.

D Stiff Confinement Modeling With Onetwo

Tokamak transport analysis typically involves from 3 to 7 coupled non-linear equations and up to 200 grid points. Depending primarily on the nonlinearities introduced by the confinement models this set of equations can range from trival to very difficult to solve efficiently. Typically specialized methods are needed. This involves representation of the equations suitable for computational solution methods, both linear and nonlinear and direct and iterative for the resulting set of equations.

D.1 The Set of TransportEquations

The equation that governs the evolution of the density of primary ion species i is

$$\left.\frac{\partial n_i}{\partial t}\right|_{\zeta} + \frac{1}{H\rho}\frac{\partial}{\partial \rho}(H\rho\Gamma_i) = S_i + S_i^{2D} \tag{220}$$

The second term in this and other equations appearing below represents the azimuthally symmetric flux surface averaged divergence of the flux. Here $\boldsymbol{\Gamma_i}$ is the particle flux $(\frac{\#}{cm^2 sec})$ of ion species i. The 2D source term appearing on the rhs of this and subsequent equations below are due to the grid motion and is given by

$$S_i^{2D} = -n_i\left.\frac{\partial}{\partial t}\right|_{\zeta}\ln H + \frac{1}{H}\left(\left.\frac{\partial \rho}{\partial t}\right|_{\zeta}\right)\frac{\partial}{\partial \rho}Hn_i \tag{221}$$

The equation for describing the evolution of the electron thermal energy is

$$\frac{3}{2}\Big(T_e\sum_{i=1}^{nion}\big(n_i\left.\frac{\partial Z_i}{\partial T_e}\right|_{\zeta}\big) + n_e\Big)\frac{\partial T_e}{\partial t} + \frac{3}{2}T_e\sum_{i=1}^{nion}Z_i\left.\frac{\partial n_i}{\partial t}\right|_{\zeta}$$

$$+\frac{1}{H\rho}\frac{\partial}{\partial \rho}\Big(H\rho(q_e + \frac{5}{2}\Gamma_e T_e)\Big) = Q_e - \omega L_e + S_{T_e}^{2D} \tag{222}$$

$[S_{T_e}^{2D}]$ represents heating of electrons due to grid motion:

$$S_{T_e}^{2D} = -\frac{5}{2}n_e T_e\frac{\partial}{\partial t}\ln H + \left(\frac{\partial \ln\rho}{\partial t}\right)\Big(\frac{5}{2}n_e T_e\frac{\partial \ln H}{\partial \rho} \tag{223}$$

$$+\frac{3}{2}T_e\sum_{i=1}^{nion}Z_I\frac{\partial n_i}{\partial \rho} + \frac{3}{2}\big(n_e + T_e\sum_{i=1}^{nion}n_z\frac{\partial Z_i}{\partial T_e}\big)\frac{\partial T_e}{\partial \rho}\Big) \tag{224}$$

The equation for describing the evolution of the ion thermal and rotational kinetic energy is

$$\sum_{i=1}^{nion}\left\{\frac{3}{2}n_i\frac{\partial T}{\partial t}\bigg|_\zeta+\frac{\partial n_i}{\partial t}\bigg|_\zeta\left(\frac{3}{2}T+\frac{1}{2}m_i\omega^2<R^2>\right)\right\}+\sum_{i=1}^{nion}m_in_i\omega<R^2>\frac{\partial\omega}{\partial t}\bigg|_\zeta$$
$$+\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left\{H\rho\left(\sum_{i=1}^{nion}(q_i+\frac{5}{2}\Gamma_iT)+\Gamma_T^\omega+\Pi\omega\right)\right\}$$
$$=Q+S_T^\omega+S_T^{2D}+S_T^{2D\omega} \quad (225)$$

With source term due to grid motion given by:

$$S_T^{2D}=-\frac{5}{2}T\frac{\partial\ln H}{\partial t}\sum_{i=1}^{nion}n_i+\frac{\partial\ln\rho}{\partial t}\left(\frac{5}{2}T\frac{\partial\ln H}{\partial\rho}\sum_{i=1}^{nion}n_i+\frac{3}{2}T\sum_{i=1}^{nion}\frac{\partial n_i}{\partial\rho}+\frac{3}{2}\frac{\partial T}{\partial\rho}\sum_{i=1}^{nion}n_i\right) \quad (226)$$

$$S_T^{2D\omega}=-\frac{1}{2}<R^2>\omega^2\frac{\partial lnH}{\partial t}\sum_{i=1}^{nion}n_im_i$$
$$+\frac{1}{2}\omega\left(spr2d+\omega(\frac{\partial<R^2>}{\partial t}+<R^2>\frac{\partial\ln H}{\partial t})\sum_{i=1}^{nion}m_in_i\right)$$
$$-\frac{1}{2}\omega^2\frac{\partial<R^2>}{\partial t}\sum_{i=1}^{nion}n_im_i \quad (227)$$

The evolution of the poloidal B field is given by Faraday's Law. In Onetwo this equation takes the form

$$\frac{1}{FG(H\rho)^2\alpha}\frac{\partial(FGH\rho B_{p0})}{\partial t}-\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left(H\rho(d_{4,1}\frac{\partial n_i}{\partial\rho}+d_{4,2}\frac{\partial T_e}{\partial\rho}+d_{4,3}\frac{\partial T}{\partial\rho}\right.$$
$$\left.+d_{4,4}\frac{\partial FGH\rho B_{p0}}{\partial\rho})\right)-\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left(\frac{\partial\rho}{\partial t}\bigg|_\zeta B_{p0}\right)=-\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left(\eta_\parallel cH<\vec{J_{aux}}\cdot\frac{\vec{B}}{B_{t0}}>\right)$$
$$+\frac{1}{H\rho}\frac{\partial}{\partial\rho}\left(H\rho(D_f^e+D_f^b)\frac{\partial n_f}{\partial\rho}\right)+\frac{B_{p0}}{H\rho}\frac{\partial}{\partial t}\left(lnFGH\rho\right)-\frac{B_{p0}}{H\rho}\frac{\partial}{\partial\rho}\left(\frac{\partial\rho}{\partial t}\bigg|_\zeta\right) \quad (228)$$

The equation for toroidal momentum and rotation used in Onetwo assumes that all of the momentum and energy is carried by the ions. All ions have the same temperature and rotation speed, the associated momentum of each ion fluid depends on the mass of the ion however. The actual equation solved by Onetwo is

$$\sum_{i=1}^{nprim} m_i n_i < R^2 > \frac{\partial \omega}{\partial t}\bigg|_{\zeta} + \omega \sum_{i=1}^{nprim} m_i < R^2 > \frac{\partial n_i}{\partial t}\bigg|_{\zeta} + \frac{1}{H\rho}\frac{\partial}{\partial \rho}\left(H\rho\Gamma_{\omega}\right) = S_{\omega} + S_{\omega}^{2D} \quad (229)$$

$S_T^{2D\omega}$ represents ion rotational kinetic energy sources due to grid motion:

$$S_T^{2D\omega} = -\frac{1}{2}(1) < R^2 > \omega^2 \frac{\partial lnH}{\partial t} \sum_{i=1}^{nion} n_i m_i + \frac{1}{2}(2)\omega\left(spr2d + \omega(\frac{\partial < R^2 >}{\partial t} + < R^2 > \frac{\partial \ln H}{\partial t}\bigg|_{\zeta}) \sum_{i=1}^{nion} m_i n_i\right) - \frac{1}{2}\omega^2 \frac{\partial < R^2 >}{\partial t}\bigg|_{\zeta} \sum_{i=1}^{nion} n_i m_i \quad (230)$$

In matrix form the set of equations [220,222, 225,228,229] is compactly written as

$$\underline{\underline{M}}\frac{\partial}{\partial t}\bigg|_{\zeta}\underline{u} - \frac{1}{H\rho}\frac{\partial}{\partial \rho}\left(H\rho\underline{\underline{D}}\frac{\partial}{\partial \rho}\underline{u}\right) + \frac{1}{H\rho}\frac{\partial}{\partial \rho}\left(H\rho\underline{\underline{V}}\underline{u}\right) + \underline{\underline{W}}\underline{u} = \underline{S_{ext}} \quad (231)$$

Here the vector $\underline{u} \equiv [n_1, ..n_N, T_e, T_i, FGH\rho B_P, \omega]$ represent the dependent variables $\underline{\underline{M}}$ is an $N+4$ by $N+4$ coefficient matrix with N ion species. For $N = 2$ we have:

$$\mathrm{M} = \begin{pmatrix} 1 & , & 0 & , & 0 & , & 0 & , & 0 & , & 0 \\ 0 & , & 1 & , & .. & , & 0 & , & 0 & , & 0 \\ \frac{3}{2}T_e\langle Z_1\rangle & , & \frac{3}{2}T_e\langle Z_N\rangle & , & \frac{3}{2}\left(n_e + T_e\sum n_i\frac{\partial Z_i}{\partial T_e}\right) & , & 0 & , & 0 & , & 0 \\ \frac{3}{2}T + \frac{1}{2}\left\langle R^2\right\rangle\omega^2 m_1 & , & \frac{3}{2}T + \frac{1}{2}\left\langle R^2\right\rangle\omega^2 m_N & , & 0 & , & \frac{3}{2}\sum\langle n_i\rangle & , & 0 & , & \sum m_i\left\langle n_i R^2\right\rangle\omega \\ 0 & , & 0 & , & 0 & , & 0 & , & \frac{1}{FGH^2\rho^2} & , & 0 \\ \omega m_1\left\langle R^2\right\rangle & , & \omega m_N\left\langle R^2\right\rangle & , & 0 & , & 0 & , & 0 & , & \sum m_i\left\langle n_i R^2\right\rangle \end{pmatrix}$$

The matrix D has a form which depends on the confinement models under investigation. A simple diagonal model would be

$$\underline{\underline{\mathrm{D}}} = \begin{pmatrix} d, & 0, & 0, & 0, & 0, & 0 \\ 0, & d, & .., & 0, & 0, & 0 \\ 0, & 0, & k_e, & 0, & 0, & 0 \\ 0, & 0, & 0, & \sum k_i, & 0, & 0 \\ 0, & 0, & .0, & 0, & \frac{c^2\eta}{4\pi F^2 H\rho^2}, & 0 \\ 0, & 0, & .0, & 0, & 0, & \sum d_{\omega} \end{pmatrix}$$

The matrix V has a form which depends on the confinement models under investigation. A simple model would be

$$\underline{\underline{\mathbf{V}}} = \begin{pmatrix} 0\,,0\,, & 0 & , & 0 & ,0\,, & 0 \\ 0\,,0\,, & .. & , & 0 & ,0\,, & 0 \\ 0\,,0\,, & \frac{5}{2}\Gamma_e & , & 0 & ,0\,, & 0 \\ 0\,,0\,, & 0 & , & \frac{5}{2}\sum\Gamma_i & ,0\,, & \frac{1}{2}\sum m_i\left\langle R^2\right\rangle\omega\Gamma_i+\pi_i \\ 0\,,0\,, & .0 & , & 0 & ,0\,, & 0 \\ 0\,,0\,, & .0 & , & 0 & ,0\,, & \sum m_i\left\langle R^2\right\rangle\Gamma_i \end{pmatrix}$$

The matrix **W** is introduced for numerical stability purposes in the finite difference approximation. Its effect is to split the source term into explicit and implicit parts. It can be shown that without this splitting the finite difference solution is unstable. The simplest form of **W** is

$$\underline{\underline{\mathbf{W}}} = \begin{pmatrix} 0\,,0\,, & 0 & , & 0 & ,0\,,0 \\ 0\,,0\,, & 0 & , & 0 & ,0\,,0 \\ 0\,,0\,, & c_\Delta & , & -c_\Delta & ,0\,,0 \\ 0\,,0\,, & -c_\Delta & , & c_\Delta & ,0\,,0 \\ 0\,,0\,, & 0 & , & 0 & ,0\,,0 \\ 0\,,0\,, & 0 & , & 0 & ,0\,,0 \end{pmatrix}$$

Where the term c_Δ represents the electron ion energy exchange term :

$$Q_\Delta = c_\Delta\left(T_e - T_i\right) \tag{232}$$

$$c_\Delta = \sum_i \frac{3m_e\left\langle Z_i^2\right\rangle n_i}{m_i Z_{eff}\tau_e} \tag{233}$$

- Putting all the pieces together we get

$$\begin{aligned} &\underline{\underline{M}}_j\frac{\partial \underline{u}_j}{\partial t} - \frac{1}{(H\rho)_j\,\Delta\rho_j}\left(\frac{(H\rho)_{j-\frac{1}{2}}\,\underline{\underline{D}}_{j-\frac{1}{2}}}{\Delta\rho_{j-\frac{1}{2}}} + (H\rho)_{j-\frac{1}{2}}\,\underline{\underline{V}}^+_{j-\frac{1}{2}}\right)\underline{u}_{j-1} \\ &-\frac{1}{(H\rho)_j\,\Delta\rho_j}\left(-\frac{(H\rho)_{j+\frac{1}{2}}\,\underline{\underline{D}}_{j+\frac{1}{2}}}{\Delta\rho_{j+\frac{1}{2}}} - \frac{(H\rho)_{j-\frac{1}{2}}\,\underline{\underline{D}}_{j-\frac{1}{2}}}{\Delta\rho_{j-\frac{1}{2}}} - (H\rho)_{j+\frac{1}{2}}\,\underline{\underline{V}}^+_{j+\frac{1}{2}} + (H\rho)_{j-\frac{1}{2}}\,\underline{\underline{V}}^-_{j-} \right. \\ &-\frac{1}{(H\rho)_j\,\Delta\rho_j}\left(\frac{(H\rho)_{j+\frac{1}{2}}\,\underline{\underline{D}}_{j+\frac{1}{2}}}{\Delta\rho_{j+\frac{1}{2}}} - (H\rho)_{j+\frac{1}{2}}\,\underline{\underline{V}}^-_{j+\frac{1}{2}}\right)\underline{u}_{j+1} = \underline{S}_{exp,j} \end{aligned} \tag{234}$$

- To bring out the structure we define new matrices $\underline{\underline{P}}, \underline{\underline{Q}}, \underline{\underline{R}}$,and write the last equation as:

$$\underline{\underline{M}}_j \frac{\partial \underline{u}_j}{\partial t} - \underline{\underline{P}}_{j-1}\underline{u}_{j-1} - \underline{\underline{Q}}_j \underline{u}_j - \underline{\underline{R}}_{j+1}\underline{u}_{j+1} = \underline{S}_{exp,j} \tag{235}$$

- The implicit time difference scheme is developed by evaluating the explicitly appearing dependent variable $\underline{u}$ at time t^{n+1} and at time t^n and averaging:

$$\underline{\underline{M}}_j^{n+\theta}\left(\frac{\underline{u}_j^{n+1} - \underline{u}_j^n}{\Delta t}\right) - \underline{\underline{P}}_{j-1}^{n+\theta}\underline{u}_{j-1}^{n+1} - \underline{\underline{Q}}_j^{n+\theta}\underline{u}_j^{n+1} - \underline{\underline{R}}_{j+1}^{n+\theta}\underline{u}_{j+1}^{n+1} = \underline{S}_{exp,j}^{n+1} \tag{236}$$

$$\underline{\underline{M}}_j^{n+\theta}\left(\frac{\underline{u}_j^{n+1} - \underline{u}_j^n}{\Delta t}\right) - \underline{\underline{P}}_{j-1}^{n+\theta}\underline{u}_{j-1}^{n} - \underline{\underline{Q}}_j^{n+\theta}\underline{u}_j^{n} - \underline{\underline{R}}_{j+1}^{n+\theta}\underline{u}_{j+1}^{n} = \underline{S}_{exp,j}^{n} \tag{237}$$

- With some appropriate definitions :

$$\underline{\underline{A}}_j^{n+\theta} \equiv -\underline{\underline{P}}_{j-1}^{n+\theta}\theta$$

$$\underline{\underline{B}}_j^{n+\theta} \equiv \frac{\underline{\underline{M}}_j^{n+\theta}}{\Delta t} - \theta\underline{\underline{Q}}_j^{n+\theta}$$

$$\underline{\underline{C}}_j^{n+\theta} \equiv -\underline{\underline{R}}_{j+1}^{n+\theta}\theta$$

$$\begin{aligned}\underline{g}_j^{n+\theta} \equiv\ & \underline{\underline{P}}_{j-1}^{n+\theta}(1-\theta)\underline{u}_{j-1}^n \\ & + \left(\frac{\underline{\underline{M}}_j^{n+\theta}}{\Delta t} + \underline{\underline{Q}}_j^{n+\theta}(1-\theta)\right)\underline{u}_j^n \underline{\underline{R}}_{j+1}^{n+\theta}(1-\theta)\underline{u}_{j+1}^n + \underline{S}_{exp,j}^{n+\theta}\end{aligned}$$

- we can cast the last result into a compact matrix form:

$$\underline{\underline{A}}_j^{n+\theta}\underline{u}_{j-1}^{n+1} + \underline{\underline{B}}_j^{n+\theta}\underline{u}_j^{n+1} + \underline{\underline{C}}_j^{n+\theta}\underline{u}_{j+1}^{n+1} - \underline{g}_j^{n+\theta} = 0 \tag{238}$$

  which holds for all interior mesh point $j \neq 1, j \neq nj$

- A similar approach is used to generate matrix equations for $j = 1$ and $j = nj$ using the boundary conditions .

- Final assembly results in a Block tri-diagonal system :

$$\begin{pmatrix} \underline{\underline{B}}_1^{n+\theta} & \underline{\underline{C}}_1^{n+\theta} & 0 & 0 & 0 \\ \underline{\underline{A}}_2^{n+\theta} & \underline{\underline{B}}_2^{n+\theta} & \underline{\underline{C}}_2^{n+\theta} & 0 & 0 \\ 0 & \underline{\underline{A}}_3^{n+\theta} & \underline{\underline{B}}_3^{n+\theta} & \underline{\underline{C}}_3^{n+\theta} & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \underline{\underline{B}}_{nj}^{n+\theta} \end{pmatrix} \begin{pmatrix} \underline{u}_1^{n+1} \\ \underline{u}_2^{n+1} \\ \underline{u}_3^{n+1} \\ \vdots \\ \underline{u}_{nj}^{n+1} \end{pmatrix} - \begin{pmatrix} \underline{g}_1^{n+\theta} \\ \underline{g}_2^{n+\theta} \\ \underline{g}_3^{n+\theta} \\ \vdots \\ \underline{g}_{nj}^{n+\theta} \end{pmatrix} = 0 \quad (239)$$

- Each sub-matrix,$\underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}}$ is n+4 by n+4 ,where n is the number of ion species and the 4 comes from the remaining dependent variables (T_e, T, B_P, ω) The vector $\underline{u}_j$ contains the dependent variables at grid point j and we have assumed a grid of size nj.

- Equations(239) represent the commmon set that is solved using either a predictor corrector method where the parameter $\theta = 0.5$ or a fuly implicit method where $\theta = 1$.

- With $\theta = 1$ Eq.(239) represent a set of non-linear equations of the form $F_i = 0, i = 1..(nj - 1) * (n + 4) - 1$ to be solved for $ni_j, Te_j, Ti_j, RBP_j, \omega_j$ at each grid point $r_j, j = 1, ..nj$. Note that EQ.(239) represent the steady sate solution if time derivative terms (associated matrices M and g)are deleted.

- Such sets of equations can be solved using a Newton type method enhanced with a strategy that insures global convergence. Typically one minimizes the sum of squares of residuals, $F^T F$, that result when an approximate solution is substituted into EQ.(239).

- We found that no single global strategy will work reliably with confinement models such as GlF23 (which is part of matrix D). Instead three methods are used in a round robin type approach to generate the solution:
  - line search
  - and two trust region methods which change both the stepsize and direction:
    - dog leg
    - hook step

- Any of the three methods will satisfactorily solve the set of EQS.(239) if neoclassical transport is done. However for some turbulent confinement models such as GLF23 none of the methods will work without help from the others. This appears to be due to the fact that we encounter regions where the functions $\boldsymbol{F_i}$ are not well represented locally by a quadratic form which is the basis of all the solution methods.

- an optimized line search in the Newton direction is standard and is not discussed here (see Nocedal et. al.)
- The two trust region strategies solve the problem by limiting the length of the step taken as well as defining an intermediate direction between the steepest descent and Newton directions.
- The absolute minima of the function

$$\boldsymbol{f(u) = \frac{1}{2}\sum F_i(u)^2} \tag{240}$$

contain the solution(s) of EQS(239). The relative minima also present in Eq.(240) have the property that at such points

$$\boldsymbol{\nabla f = \underline{\underline{J^T}}\ \underline{F} = 0} \tag{241}$$

But a relative minimum has $\boldsymbol{\underline{F} \neq 0}$ which implies that the columns of J must be linearly dependent at such points. Such singular J would also cause our solution method to fail or produce poor steps. Hence the Jacobian is perturbed away from such points by adding a minimal perturbation to the diagonals that insures that J has an acceptable condition number.

- The local linear representation of the set of EQS(239):

$$\boldsymbol{\underline{F(\underline{u_c} + \underline{s})} = \underline{F(\underline{u_c})} + \underline{\underline{J(\underline{u_c})}}\ \underline{s}} \tag{242}$$

leads to the Newton step $\boldsymbol{\underline{s}}$ to be taken from the current approximate solution point $\boldsymbol{\underline{u_c}}$ by solving

$$\boldsymbol{\underline{\underline{J(\underline{u_c})}}\ \underline{s} = -\underline{F(\underline{u_c})}} \tag{243}$$

- To introduce higher order terms in the global strategy that allow for a deviation from the Newton direction we note that the quadratic form

$$\boldsymbol{\frac{1}{2}\underline{F(\underline{u_c} + \underline{x})}^T \underline{F(\underline{u_c} + \underline{x})} = \frac{1}{2}\underline{F^T F} + \left(\underline{\underline{J^T}}\underline{F}\right)^T \underline{x} + \frac{1}{2}\underline{x^T \underline{\underline{J^T}} J x}} \tag{244}$$

is positive for all $\boldsymbol{\underline{x}}$ except the Newton solution $\boldsymbol{\underline{x} = \underline{s}}$ where its value is zero.

- The quadratic form,EQ(244) is closely related to the quadratic form of f defined in EQ(240)

$$\boldsymbol{f(\underline{u_c} + \underline{x}) = f(\underline{u_c}) + \left(\underline{\underline{J^T}} \underline{F}\right)^T \underline{x} + \frac{1}{2} \underline{x^T} \underline{\underline{H}} \underline{x}} \tag{245}$$

  where the Hessian H is $\underline{\underline{\boldsymbol{J^T J}}}$ plus terms involving the second derivative of f. To minimize f we would look for the minimizer of this local representation of f.

- The dogleg and hook step trust region strategies are based on finding the minima in f using the modified form,EQ(244) subject to the constraint that the step size is limited to an apriori specified length,$\boldsymbol{\delta}$. It can be shown that any such constrained local minimizer of EQ(244) is in a direction that decreases the value of f so long as the approximation to H is positive definite.

- For the hookstep the solution is

$$\boldsymbol{\underline{x} = -\left(\underline{\underline{J^T J}} + \mu \underline{\underline{I}}\right)^{-1} \underline{\underline{J^T}} \underline{F}} \tag{246}$$

  where $\boldsymbol{\mu_c \geq 0}$ is found iteratively so that $\boldsymbol{||x|| \approx \delta}$. For small $\boldsymbol{\mu}$we approach the newton direction while for large $\boldsymbol{\mu}$ the steepest decent direction is approached.

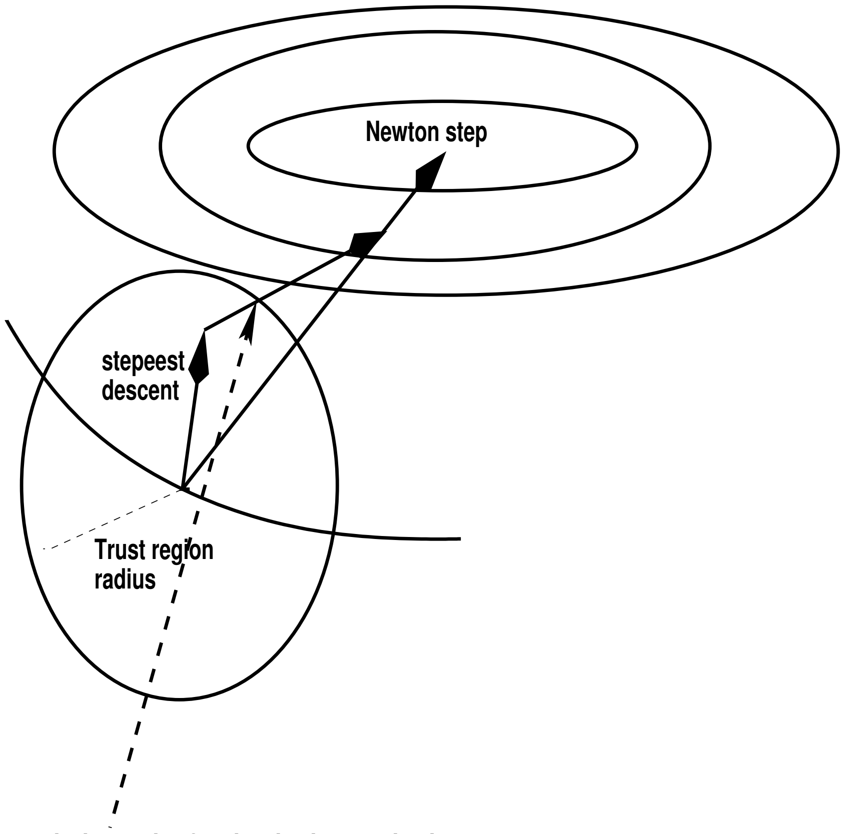
- The dogleg method effectively uses the size of $\boldsymbol{\delta}$ to interpolate between the steepest descent and Newton directions. The dogleg is less optimal than the hookstep but the computations are less expensive. The Cauchy point for the dogleg is defined as the minimizer of EQ(244) in the steepest descent direction:

$$\boldsymbol{\underline{x_{cp}} = \underline{u_c} + \lambda \nabla f(\underline{x_c})} \tag{247}$$

  where $\boldsymbol{\lambda = \frac{||\nabla f||_2^2}{\nabla f^T J^T J \nabla f}}$ The point N on the Newton path is some multiple $\boldsymbol{\left|\left|\underline{x_c} - \underline{x_{cp}}\right|\right| \leq \alpha \leq 1}$ of the newton step $\boldsymbol{s = -J^T F}$:

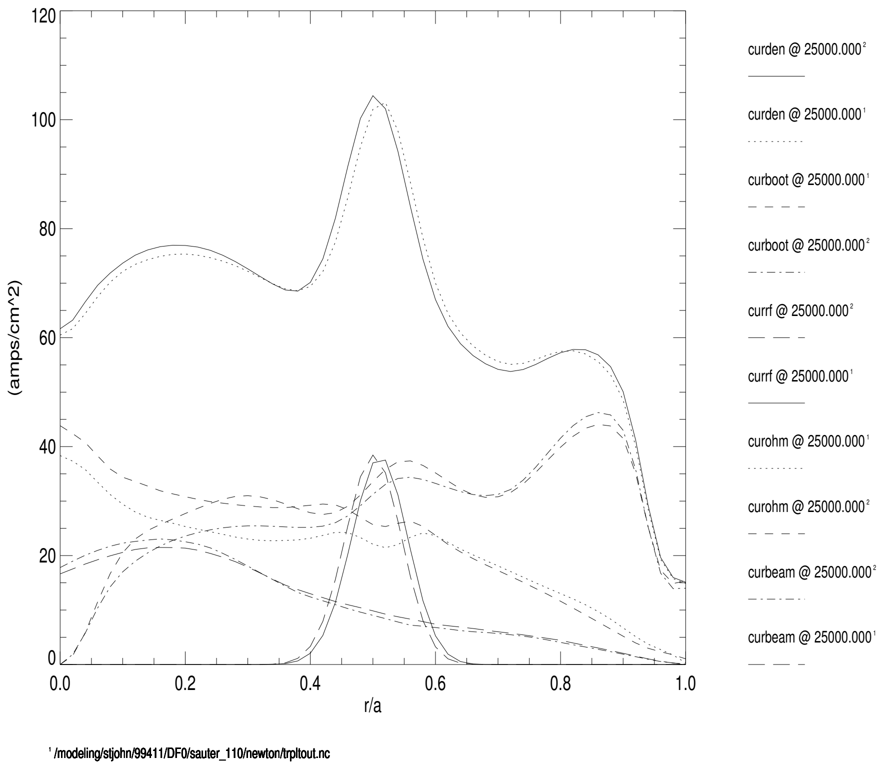
$$\boldsymbol{\underline{x_N} = \underline{u_c} \alpha \underline{s}} \tag{248}$$

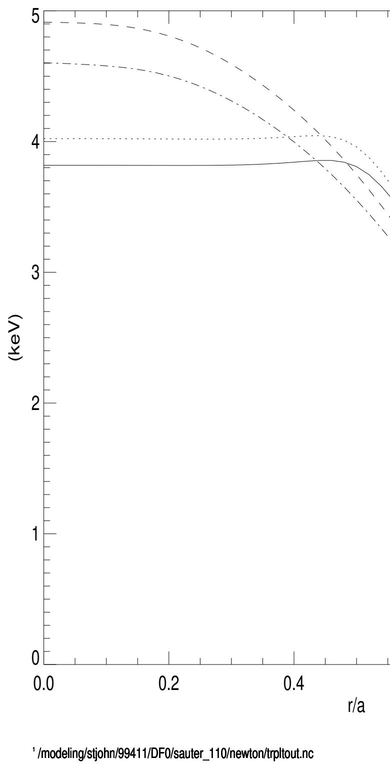
  The value of $\boldsymbol{\alpha}$ was originally set to 1. Experimentation has shown that an optimal alpha is approximately $\boldsymbol{\alpha = 0.8 * \gamma + 0.2}$ where $\boldsymbol{\gamma \leq 1}$ is ratio of the Cauchy step length to the Newton step length.



- **the solution point for the dogleg method**

- A major advantage of the fully implict approach described above is the ability to generate a steady state solution directly. For the usual AT scenario this involves finding quasi stationary profiles for temperatures,densities,poloidal magnetic field and toroidal momentum. This can be done in only a small fraction of the computational time that would be required in a standard approach and makes "what if" type investigations much more accessible.





- We have found that it is possible to solve the nonlinear sets of equations associated with stiff confinement models (that typically lead to formation of internal transport barriers) using globaly convergent modifications of the standard Newton method. In particular we find that ad hock modifications of the GLF23 confinement are not necessary in order to achieve convergence.

- Successful application of the method required that three approaches be used in a round robin type fashion to achieve convergence. In particular the use of exisitng solvers that just implemented the line search method were found to be unsatisfactory.

- The primary benefit of an adaptive grid is that fewer grid points can be used. This leads to more rapid convergence of the non-linear iterations and consequently can yield a significant decrease in computational time involved. The dynamic adjustment of the grid remains to be investigated.

- Rapid determination of steady state results made possible by a fully implicit approach allow effeicient investigation of AT and other steady state scenarios.

- There are many aspects of this problem which remain un investigated that could conceivably improve the performance with a minimal expenditure of code development time.

E Adaptive Grid Method

- Need a fully automatic robust method to generate smooth radial (in $\boldsymbol{\rho} = \sqrt{\frac{\boldsymbol{\Phi}}{\pi \boldsymbol{B_{T0}}}}$) grid that will adapt itself to the time dependent solution.
- The method used here defines the adaptation through an independently specifiable positive definite weight function $\boldsymbol{W(\zeta, t)}$. Where $\boldsymbol{\zeta}$ is a normalized,uniformly spaced grid defined on $[\mathbf{0}, \mathbf{1}]$ and fixed for all times.
- We take the grid spacing in rho to be proportional to the $\boldsymbol{\zeta}$ grid spacing and to the weight function $\boldsymbol{W}$.

$$\boldsymbol{\Delta\rho = cW\Delta\zeta}$$

This suggests that a differential equation for the $\boldsymbol{\rho}$ grid can be taken as

$$\frac{\partial}{\partial\zeta}\left(\frac{1}{W}\frac{\partial\rho}{\partial\zeta}\right) = 0 \qquad (249)$$
$$\rho(0,t) = 0, \rho(1,t) = \rho_a(t), \rho(\zeta,0) = \rho_a(0)\zeta$$

- The actual form for $\boldsymbol{W}$ used is a product of curvature and derivative weightings :

$$\begin{aligned} W &= (1+\beta|C_w|)D_w \qquad (250) \\ C_w &= \frac{\frac{\partial\rho}{\partial\zeta}\sum_i^n \beta_i \frac{\partial^2 u_i(\rho,t)}{\partial\zeta^2} - \sum_i^n \epsilon_i (\frac{\partial u_i(\rho,t)}{\partial\zeta})^2}{\left(\left(\frac{\partial\rho}{\partial\zeta}\right)^2 + \sum_i^n \epsilon_i (\frac{\partial u_i(\rho,t)}{\partial\zeta})^2\right)^{\frac{3}{2}}} \\ D_w &= \frac{\frac{\partial\rho}{\partial\zeta}}{\sqrt{\left(\frac{\partial\rho}{\partial\zeta}\right)^2 + \sum_i^n \epsilon_i (\frac{\partial u_i(\rho,t)}{\partial\zeta})^2}} \end{aligned}$$

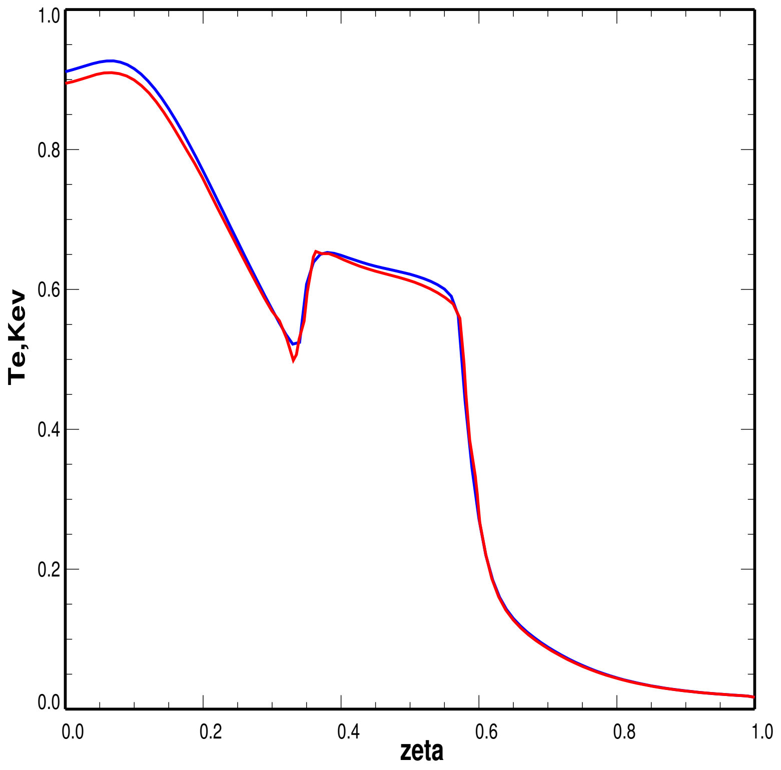
The $\boldsymbol{\beta}$**,** $\boldsymbol{\beta_i}$,and $\boldsymbol{\epsilon_i}$ are user selected weights that determine the sensitivity to curvature and gradient effects in the dependent variables $\boldsymbol{u_i}$. Originally only the derivative weighting was used. It was found that this tended to put very small and very large spacings adjacent in the grid which can be problematic in calculating derivatives numerically.

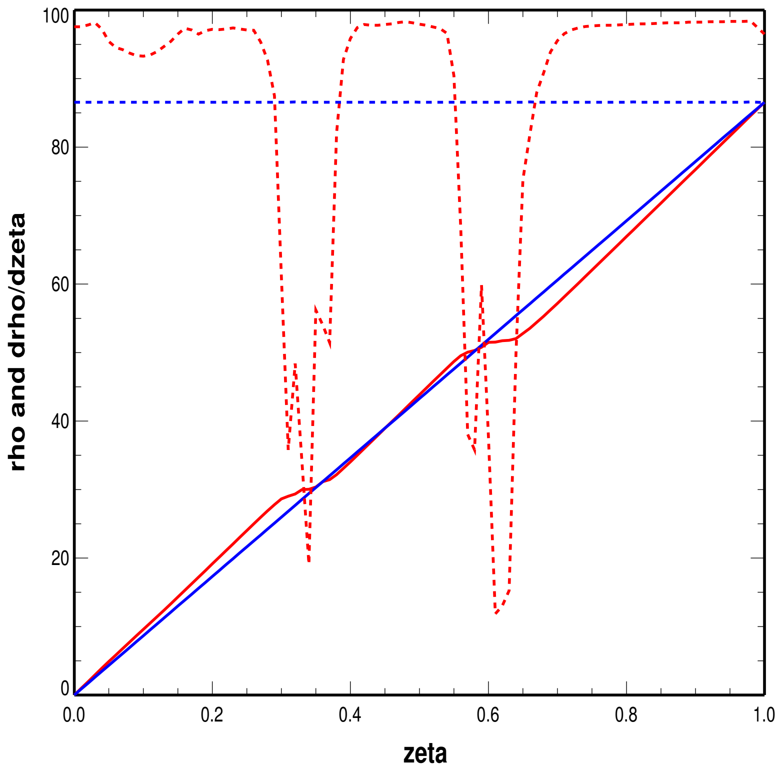
- This method fits in naturally with free boundary equilibrium and transport coupled calculations where rho is generally a function of time. Thus it is possible to adapt the grid not only for kinetic profile solutions but also simultaneously for changes in the mhd equilibrium properties,using the weight function $\boldsymbol{W(\zeta, t)}$ and the boundary condition of $\boldsymbol{\rho}$,Eq.[250].

- The sources of movement of the grid are twofold:

  1. the solution of of Eq [250] at time $\boldsymbol{t + \Delta t}$ will be different from the solution at time $\boldsymbol{t}$ even if $\boldsymbol{\rho_a(t)}$ is constant so long as the profiles, $\boldsymbol{u_i}$,evolve.
  2. the value of the plasma radius at time t as determined from a free boundary equilibrium code.

As an *illustrative* example consider the adaption of the $\boldsymbol{\rho}$ grid to the electron temperature profile generated by a localized source of ech heating, see Fig.[E].





Fig[1]The $\boldsymbol{T_e}$ profile which results from localized heating for the adaptive,51 pt, (red) and uniform, 201 pt, (blue) $\boldsymbol{\rho}$ grids.
Fig[2]The adaptive (red) and uniform (blue) rho grids as a function of $\boldsymbol{\zeta}$ and the derivatives $\frac{\partial\rho}{\partial\zeta}$ (dashed).

- The blue curve in Fig[1] is the solution of Eq.[250] with constant $\boldsymbol{W}$:

$$\rho(t) = \rho_a(t)\zeta \tag{251}$$

and represents uniform grid spacing at any time t since the $\boldsymbol{\zeta}$ grid is uniform. The red curve represents the final solution of Eq.[250], with the time dependent weight function given by Eq.[250]. The change in slope of the $\boldsymbol{\rho}$ curve represents a change in grid spacing. Thus whenever $\frac{\partial\rho}{\partial\zeta}$ is greater than the constant slope of the uniform case we have expansion of the grid and when $\frac{\partial\rho}{\partial\zeta}$ is less than the constant slope of the uniform case the grid is compressed compared to the original uniform starting grid.

The evolvement of the $\boldsymbol{\rho}$ grid is determined by the time derivative of

Eq.[250]:

$$\frac{\partial}{\partial t}\left(\frac{\partial}{\partial \zeta}\left(\frac{1}{W}\frac{\partial \rho}{\partial \zeta}\right)\right) = 0 \tag{252}$$

$$\tag{253}$$

The general solution of this equation is

$$\rho(\zeta, t) = h(t)\int_0^{\zeta} W(\zeta, t)d\zeta + F(t) \tag{254}$$

where $h(t)$ is independent of ζ. The boundary condition at $\zeta = 0$ implies $F(t) \equiv 0$ and at $\zeta = 1$ we must have

$$h(t)\int_0^1 W(\zeta, t)d\zeta = \rho_a(t) \tag{255}$$

- In practice we use the normalized weight function W^* so that Eq[255] is automatically satisfied:

$$W^*(\zeta, t) = \frac{\rho_a(t) * W(\zeta, t)}{\int_0^1 W(\zeta, t)d\zeta} \tag{256}$$

- To advance the rho grid in time we use the expression

$$\rho(\zeta, t + \Delta t) = \rho(\zeta, t) + s_\rho \left.\frac{\partial \rho}{\partial t}\right|_{\zeta} \Delta t \tag{257}$$

where the derivative is obtained from

$$\left.\frac{\partial \rho}{\partial t}\right|_{\zeta} = \int_0^{\zeta} \frac{\partial W^*(\zeta, t)}{\partial t} d\zeta \tag{258}$$

- The time derivative of W comes from the original dependent variables U_i where i ranges over particle densities,electron and ion temperatures,Faraday's law and toroidal rotation as well as ρ,see Eq.[250] Hence we see that the ρ grid will adjust itself, at any fixed ζ grid point, with a speed that depends on how rapidly the selected profiles are changing in time at that same value of ζ.

- Each of the dependent variables must be available on the appropriate ρ grid. This typically entails interpolation of the these quantities from one grid to the next and adds some overhead to the calculations. This overhead is generally (but not always at this time) compensated by increased accuracy and reduced iteration count required compared to a similar solution method which uses more grid points to achieve the same accuracy and stability.

- To obtain the $\boldsymbol{\rho}$ grid at any given time we have the following algorithm:
  1. Given $\boldsymbol{u_i^n(\rho, t)}$ at time point n and the grid $\boldsymbol{\rho^n(\zeta, t)}$ obtain $\boldsymbol{u_i^{n+1}(\rho, t)}$ by using a predictor method (see below).
  2. Interpolate the predicted solution onto the $\boldsymbol{\zeta}$ grid
  3. Obtain a new $\boldsymbol{\rho}$ grid by solving Eq[254] The time dependence is from the predictor step
  4. Interpolate $\boldsymbol{u_i^{n+1}(\rho, t)}$ onto the new $\boldsymbol{\rho^{n+1}}$ grid.
  5. iterate the above steps until converged.

- To account for the evolvement of rho from the initial prescribed rho grid (which may or may not be uniform) to an adaptive,moving grid the time derivative of the transport quantities,$\boldsymbol{u_i}$ at constant $\boldsymbol{\rho}$ is changed to one at constant $\boldsymbol{\zeta}$ using the relationship

$$\left.\frac{\partial \boldsymbol{u}_i}{\partial t}\right|_{\boldsymbol{\rho}} = \left.\frac{\partial \boldsymbol{u}_i}{\partial t}\right|_{\boldsymbol{\zeta}} - \left(\left.\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{\rho}}\right|_{\boldsymbol{t}}\right)\left(\left.\frac{\partial \boldsymbol{\rho}}{\partial t}\right|_{\boldsymbol{\zeta}}\right) \tag{259}$$

  The transformed diffusion equations will thus have additional source terms due to the moving $\boldsymbol{\rho}$ grid .

F Generating Profiles Consistent with Mhd Equilibrium Pressure Profiles

Quite often the situation arises where we have an equilibrium that we would like to use together with profiles of densities and tempertures that are scaled to yield satisfactory performance. For example scaling up of DIII-D AT scenarios to Iter discharges represents us with this situation.

There are some basic tools available to accomplish this task. Depending on what one chooses as known profiles there are some varioations i the method. The basic idea is to consider the equations for charge neutrality,zeff,and eqdsk pressure together with sufficient auxiliary information to allow a unique detrermination of profiles that are then self consistent. The basic set of equations we work with are

$$n_e - Z_{p1}n_{p1} - Z_{p2}n_{p2} - Z_{imp1}n_{imp1} - Z_{imp2}n_{imp2} = Z_b n_b + Z_\alpha n_\alpha \tag{260}$$

$$\begin{aligned} Z_{eff}n_e - < Z_{p1}^2 > n_{p1} - < Z_{p2}^2 > n_{p2} \\ - < Z_{imp1}^2 > n_{imp1} - < Z_{imp2}^2 > n_{imp2} = Z_b^2 n_b + Z_\alpha^2 n_\alpha \end{aligned} \tag{261}$$

$$\begin{aligned} n_e C_e T_e + n_{p1} C_i T_i + n_{p2} C_i T_i \\ + n_{imp1} C_i T_i + n_{imp2} C_i T_i = P - \frac{2}{3}(w_{beam} + w_\alpha) \end{aligned} \tag{262}$$

$$Z_{frac}n_{p1} - n_{p2} = 0 \tag{263}$$

$$Z_{impfrac}n_{p1} - n_{imp2} = 0 \tag{264}$$

Here we assume two primary (ie hydrogenic) ions and two impurities with the last two equations specifying the amounts of the second species in each case. We assume that the beam and α densitites and stored energy densitites are given. This implies that an iterative process is required to solve the linear set of equations since beam deposition and fusion rates depend on the unknown densitites and temperatures. P is the known pressure profile from the equilibrium calculations. Typically we assume that the electron density profile is known which eliminates the first of the equations EQ[260]. The above set of eqautions applies at each value of the minor radius grid ρ. The parameters multiplying the electron and ion temperatures, C_e, C_i are set to unity if EQ.[260] is included or they are automatically adjusted at each radius in such a way that the error in Eq[260] is minimized if that equation is not included. Typically we want to fix the electron density at a specified value so the later case is generally the one that arises. The general prodeedure is a follows:

1. Start with an appropriate inone and eqdsk file. The inone file will contain densitites and temperatures that are not consistent with the eqdsk pressure profile. Run onetwo (snapshot mode perhaps) to generate the qikone file.

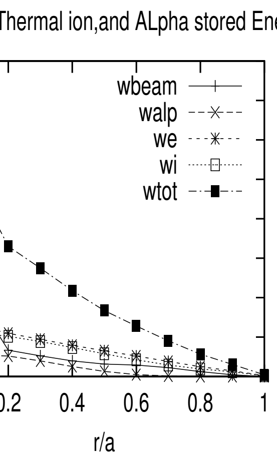
2. Use the (Python) script $\boldsymbol{gnuplot_qikone.py}$ to read the relevant information out of qikone. $\boldsymbol{gnuplot_qikone.py}$ also genrates a page of graphs if possible and most importantly, the file fixbd.in is written. Fixbd.in contains the informaion required by the fixbdry code so that the above set of equations can be solved.

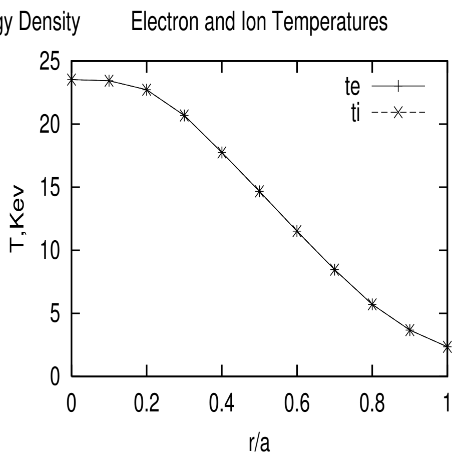
3. Edit fixbd.in to fill in the fields that are marked with *****. These are user prefrences and determine,for example, if 4 or 5 equations are solved.

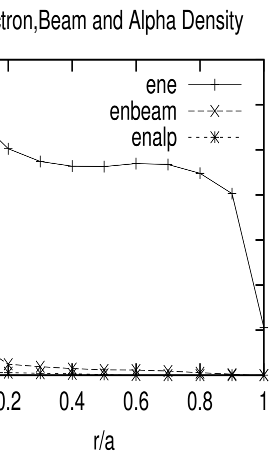
4. Run the fixbdry code:
   fixbdry129x129 fixbd.in
   The fixbdry code generates an output file profiles.dat which are suitable for inclusion iin the inone file. Python program $\boldsymbol{gnuplot_profiles.py}$ also reads this file and generates some graphics output.

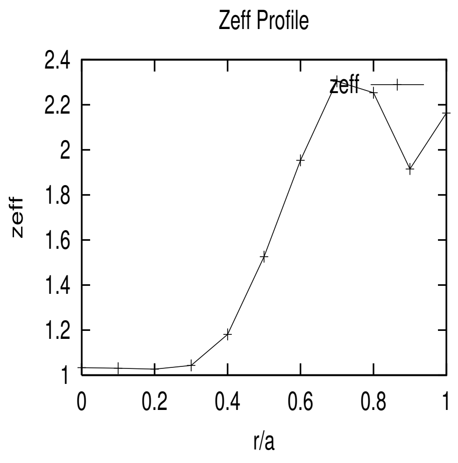
5. After pasting the appropriate inforamtion from profiles.dat into the inone file youreturn to step 1 for the next iteration. Typically 2-3 iterations are sufficient to stabilize the solution.

The graphical output from gnuplot_qikone.py and gnuplot_profiles.py for a typical case is shown in Figs .


Thermal ion,and ALpha stored Energy Density
wbeam
walp
we
wi
wtot
0.2
0.4
0.6
0.8
1
r/a


Electron and Ion Temperatures
te
ti
T,Kev
0
5
10
15
20
25
0
0.2
0.4
0.6
0.8
1
r/a


tron,Beam and Alpha Density
ene
enbeam
enalp
0.2
0.4
0.6
0.8
1
r/a


Zeff Profile
zeff
zeff
1
1.2
1.4
1.6
1.8
2
2.2
2.4
0
0.2
0.4
0.6
0.8
1
r/a

G Toray Interface

Toray can be run both as a stand alone code as well as a slave process controlled by Onetwo. In either case Toray requires at least the input file mhddat before it can be run. Namelist files, toray.in and gafit.in are required in some circumstances and the kinetic data file, echin, is required before file mhddat can be generated. The file echin is generally obtained by running Onetwo even if toray is to be run in stand alone mode. The mhddat file is created by Gafit. In versions of Toray prior to version 1.4 the only way that Gafit could generate the mhddat file was by reading the mhd information from an input file,psiin, generated by Onetwo. Starting with version 1.4 of Toray, Gafit is no longer an indepedent code. Instead, the functionality of Gafit has been absorbed in Toray. This requires that Onetwo must spawn Toray directly if Toray v. 1.4 or greater is used or Onetwo must spawn Gafit and then spawn Toray if a version of Toray prior to version 1.4 is used. Since it appears that there may be use for situatons where gafit is run independently of Toray, even for Toray versions that have a gafit built into them, /ot lets the user decide which option to use.

To spawn Toray directly requires that Onetwo must create the file toray.in with appropriate information in it or, alternatively, the user can create toray.in before running Onetwo. The kinetic data file, echin, must still be created by Onetwo. The file psiin can still be created by Onetwo, or alternatively, an eqdsk file, which must be called eqskin, can be read by the internal Gafit section of Toray 1.4 and the file psiin generated that way. The decision as to which way psiin is generated is made by setting ipsi =0 or 1 in input file gafit.in.

Assuming that we are running Toray v1.4 or greater as an Onetwo spawned process we have to set switch igafit in toray.in to 0 or 1 to indicate that file mhddat is to be generated by gafit or just read in because it allready exists. The default value of igafit is 0 .

A number of different equilibrium and transport grid sizes have arisen over the past several years. Since neither Toray nor Onetwo are dynamically allocated we need to have some way of determining what the relevant grid sizes are that each code is compiled with. The convention choosen for Onetwo is that the code name is followed by the mhd grid size and then the transport grid size. For square mhd grids the second mhd dimension is left off. hence we have onetwo_129_51, onetwo_65_129_51, onetwo_65_201, etc.(the last number is always the radial transport grid size) The toray and gafit versions linked by Onetwo are commensurate with this scheme and stored in palaces that the Onetwo code knows about. The user can take control of which version of Toray should be run by Onetwo,see below. Alternatively Onetwo will use the latest version of Toray that it knows about. The version of Toray run will is printed out in file outone as well as to the terminal screen).

G.1 Implementation

The default setting of switch ipsi in gafit.in for Toray 1.6 is 0. This is the correct value for spawing Toray from Onetwo and hence gafit.in is not required unless the user has some other reason to supply it. If gafit.in is supplied then ipsi =0 must be set. Onetwo WILL FORCE THE CORRECT SETTING BY MODIFYING THE USERS GAFIT.IN IF NECESSARY. The default setting of igafit is 0, which is not appropriate for spawning Toray from Onetwo. Hence a toray.in file must be created by Onetwo if the user did not supply one. If the user did supply a toray.in file then that file is checked to make sure that igafit=1 is set. If is is not Onetwo WILL FORCE THE CORRECT SETTING BY MODIFYING THE USERS toray.in file. (actually both gafit.in and toray.in will be copied to gafit.in_user and toray.in_user and new copies with the switches ipsi and igafit set correctly will be generated).

The following new switches must be set in the second namelist of Onetwo in order to run Toray as a slave process. This additional input applies to all versions of Onetwo.

```
$namelis2
 ...     beam input, etc.
 toray_version = 0.97
 toray_path  =
 echin_save  =  0
 irfcur(i) = 1.0 , 1.5, etc now a flaoting point number
 ...     rest of namelist
$end
```

Plot12 now has heating and current drive for the various rf cases broken out individually. Since there is some question about current drive efficiency the values of the current drive multiplier,irfcur, are now floating point nubers instead of integers as noted in the above namelist segement. This allows application of the full heating power, independent of the amount of current drive that is wanted. A description of the new input variables is given in cray102.f For convenience the descriptions are also repeated below.

There are three ways to spawn a version of Toray. The first is to just rely on the default settings in Onetwo (see default setting of the switches). This way should almost always be what the user wants. The second is to specify that a particular version of Toray is to be run,using switch toray_version. Onetwo will search its Toray paths on the architecture it is running on and if an appropriate version with the right mhd and transport grid sizes is found, it will be used. If this fails the user will be informed and the code will quit.(It is becoming quite challenging to keep up with all Toaty versions, grid sizes, and architectures. Hence users that require soem special version/ architecture may specifically have to request it). The third way is to specify a path to a custom built Toray. (This allows the latest versions of Toray to be used before they become public for example). Set toray_path to the complete executable name. For example(on Hydra): /u/stjohn/toray/toray/129_51/hp/mytoray. This will force Onetwo to use the executable called mytoray in /u/stjohn/toray/toray/129_51/hp For your custom toray you must be sure that the Onetwo and Toray grid sizes are commensurate. The set of standard sizes for Onetwo are (65x65,51), (129x129,51), other standard sizes will be defined as necessary or requested. Note that mytoray above is not restricted in any way. You could for example put a script in file mytoray (which must be executable) that goes to some remote machine, does some toray like calculations and returns the results in file echout that onetwo knows how to read.

Toray_path, if set, takes precedence. If not set (in inone) then toray_version is checked. If this is not set then the default mode is used. The prebuilt versions of Toray that Onetwo has access to are given in file // /u/stjohn/onetwo_nubeam/set_rf_data.f9
//

For reference that file is listed here (but it will probbly change).

```
      subroutine set_rf_data
!  this pculiar way of initalizations was forced upon me by the pg90
!  compiler -HSJ
        USE rf_info
            host_names =                          &
                 (/'TAURUS','lohan1','lohan2','HYDRA ', &
                              'delphi','cardea','katze '/)
            !it should be fairly obvious  how to modify this
            !the assumed properties (in sub get_toray) are
            !given here:
            ! (1) all hosts must have n_versions  versions available:
             versions = (/ 0.97,1.41 /)

            !(2) last three fields in toray_paths
            !(ie fileds after root_str) must follow the
            !root_str(j)//'/vx.xx/grid/toray'
            !pattern otherwise sub get_toray will fail !!!!!!
            !(3) obviously if any of the parameters
            !ncpu_arch,n_versions,grid_types are
            !changed then the following data statements must be
            !changed accordingly.
            !(I do not have  the luxury of creating code generators
            !to do such things - HSJ )


            !taurus data:
            toray_paths(1,1,1)= '/usr/local/bin/toray/v1.41/129_51/toray'
            toray_paths(1,2,1)= '/usr/local/bin/toray/v0.97/129_51/toray'
            toray_paths(1,1,2)= '/usr/local/bin/toray/v1.41/65_51/toray'
            toray_paths(1,2,2)= '/usr/local/bin/toray/v0.97/65_51/toray'
            toray_paths(1,1,3)= '/usr/local/bin/toray/v1.41/129_201/toray'
            toray_paths(1,2,3)= '/usr/local/bin/toray/v0.97/129_201/toray'
            root_str(1) = '/usr/local/bin/toray'


            !lohan1 data:
            toray_paths(2,1,1)= '/usr/local/bin/toray/v1.41/129_51/toray'
            toray_paths(2,2,1)= '/usr/local/bin/toray/v0.97/129_51/toray'
            toray_paths(2,1,2)= '/usr/local/bin/toray/v1.41/65_51/toray'
            toray_paths(2,2,2)= '/usr/local/bin/toray/v0.97/65_51/toray'
            toray_paths(1,1,3)= '/usr/local/bin/toray/v1.41/129_201/toray'
            toray_paths(1,2,3)= '/usr/local/bin/toray/v0.97/129_201/toray'
            root_str(2)  = '/usr/local/bin/toray'


            !lohan3 data:
            toray_paths(3,1,1)= '/usr/local/bin/toray/v1.41/129_51/toray'
            toray_paths(3,2,1)= '/usr/local/bin/toray/v0.97/129_51/toray'
            toray_paths(3,1,2)= '/usr/local/bin/toray/v1.41/65_51/toray'
            toray_paths(3,2,2)= '/usr/local/bin/toray/v0.97/65_51/toray'
            toray_paths(1,1,3)= '/usr/local/bin/toray/v1.41/129_201/toray'
            toray_paths(1,2,3)= '/usr/local/bin/toray/v0.97/129_201/toray'
            root_str(3) = '/usr/local/bin/toray'

            !hydra data:
            toray_paths(4,1,1)= '/u/stjohn/toray/v1.41/129_51/toray'
            toray_paths(4,2,1)= '/u/stjohn/toray/v0.97/129_51/toray'
            toray_paths(4,1,2)= '/u/stjohn/toray/v1.41/65_51/toray'
            toray_paths(4,2,2)= '/u/stjohn/toray/v0.97/65_51/toray'
            toray_paths(1,1,3)= '/u/stjohn/toray/v1.41/129_201/toray'
            toray_paths(1,2,3)= '/u/stjohn/toray/v0.97/129_201/toray'
            root_str(4) = '/u/stjohn/toray/'

            !delphi data:
            toray_paths(5,1,1)= '/usr/local/bin/toray/v1.41/129_51/toray'
            toray_paths(5,2,1)= '/usr/local/bin/toray/v0.97/129_51/toray'
            toray_paths(5,1,2)= '/usr/local/bin/toray/v1.41/65_51/toray'
            toray_paths(5,2,2)= '/usr/local/bin/toray/v0.97/65_51/toray'
            toray_paths(1,1,3)= '/usr/local/bin/toray/v1.41/129_201/toray'
            toray_paths(1,2,3)= '/usr/local/bin/toray/v0.97/129_201/toray'
            root_str(5) = '/usr/local/bin/toray'

            !cardea data:
            toray_paths(6,1,1)= '/usr/local/bin/toray/v1.41/129_51/toray'
            toray_paths(6,2,1)= '/usr/local/bin/toray/v0.97/129_51/toray'
            toray_paths(6,1,2)= '/usr/local/bin/toray/v1.41/65_51/toray'
            toray_paths(6,2,2)= '/usr/local/bin/toray/v0.97/65_51/toray'
            toray_paths(1,1,3)= '/usr/local/bin/toray/v1.41/129_201/toray'
            toray_paths(1,2,3)= '/usr/local/bin/toray/v0.97/129_201/toray'
            root_str(6) = '/usr/local/bin/'

            !katze data:
            toray_paths(7,1,1)= '/usr/local/bin/toray/v1.41/129_51/toray'
            toray_paths(7,2,1)= '/usr/local/bin/toray/v0.97/129_51/toray'
            toray_paths(7,1,2)= '/usr/local/bin/toray/v1.41/65_51/toray'
            toray_paths(7,2,2)= '/usr/local/bin/toray/v0.97/65_51/toray'
            toray_paths(1,1,3)= '/usr/local/bin/toray/v1.41/129_201/toray'
            toray_paths(1,2,3)= '/usr/local/bin/toray/v0.97/129_201/toray'
            root_str(7)  = '/usr/local/bin/toray'
        end subroutine set_rf_data
```

you can check the build date in these directories to find out what the latest build is.

Issues involved with tdem mode of operation of Onetwo are handled automatically if the user has choosen to run TDEM mode.

gafsep This quantity was previously passed to Toray with a fixed value of 1.e-6. The user selected input vaue was ignored. It is now passed to toray as expected. The default has been set to 1.e-6 .

toray_version default is the most recent version that Onetwo knows about. This information is keept in file ext_prog_info.f90 mentioned above. Specify a number . gt. 1.3 for new Toray f90 version with an internal gafit . Specify a number less than 1.3 to get the old version of toray with gafit run as a separate program.

toray_path The default path is set internally to point at the selected version of toray. If you want to run a specific version then you can set the path to that version here. If toray_path is set you must also set toray_version (so that Onetwo can select the proper interface)! Note that if you use gafit.in and/or toray.in then these files must be commensurate with the version of toray that you specified in toray_path ! toray_path is a 256 (or less) character variable. Onetwo spawns gafit and toray as separate processes for versions less than 1.3 and spawns just the toray process otherwise. Note that no attempt is made to do remote procedure calls so that a path pointing to a dfferent machine other than the one that Onetwo is running on is not allowed.

echin_save integer, either 0 or 1, default 0. Toray can be run in standalone mode using just the files echin and mhddat. Hence if the files echin and mhddat are saved as Onetwo executes it is possible to go back and rerun toray (perhaps changing something in the echin file). echin_save =0 retains only the last version of echin,mhddat created by Onetwo. echin_save =1 saves all versions of these files indexed by a time stamp. Notice that if you have a time sequence of eqdsks processed using TDEM mode then this option is a convenient way to get time interpolated mhddat files. Note however that mhddat is a binary file which means its non portable across machines.(A netcdf file would make more sense here I think but thats a Toray issue). Finally recall that Toray can be run using only the input files echin and mhddat only if toray.in has igafit =0 (which is the default in the Toray v1.4 code).

Note that mhddat is a binary file hence it cannot be read on a different machine architecture than it was written on.

G.2 MEPC Code

The multiple eqdsk processor code (MEPC) must be run before a Onetwo tdem run can be done. The MEPC code takes as input a list of up to kbctim = 150 (parameter kbctim is defined in param.f90) time evolved eqdsks and generates a single netcdf output file. Please note that Onetwo requires that at least 3 eqdsks are in the netcdf file. The netcdf file name is then input into the third namelist of inone in place of an eqdsk name as explained above. MEPC will print out instructions on how to use it if you execute the code without any command line arguments. The required arguments are listed by the code and hence they are not repeated here. The list of eqdsk that MEPC will process is given in a file (with a name specified as one of the command line arguments. For definitenes we will assume the file is called eqdsk.list here). Eqdsk.list contains a list of eqdsks that will be used to generate the netcdf output file. The Eqdsk.list file can be created by a gui program, MEPC.tcl or, alternatively, this file can be created manually with a text editor of your choice. When reading the Eqdsk.lsit file the MEPC code scans each line in the file for special sentinels. Any line starting with #,;, or ! will be ignored. Valid input lines that specify eqdisks that cant be found are skipped over. The sentinel END on a line by itself will terminate the reading of the Eqdsk.list file. Eqdsks that are smaller than the compiled in size of the code (currently 129 by 129 in R,Z)

```
c  namelist mepcinput:
c  time_start             Use only those eqdsks that fall in this time range
c  time_end                default is (0.0, 1.e30)

c  rho_calc_method         0 or 1, default 1.
c                          0 means use q to get rho
c                          1 means use toroidal flux to get rho
c  fit_rhomax_type
c  fit_psi_type           these are fitting specifiers for
c                         rhomax and psi
c                         possible values are
c                                "none"    eg no fit to data
c                                "linear"  eq do a linear fit
c                                "spline_icsvku"  !spline fit
c                                "spline_icsscv"  !spline fit
c                                see the IMSL manuals for the
c                                description of the spline fits
c
c
c                          note that dpsidt_const_zeta(j,i),
c                          dpsidt_const_rho(j,i),and
c                          drhodt_const_zeta = rho_12(j)**drhomaxdt(i)
c                          are not calculated if fit_psi_type.eq. "none"
c                          It is then up to Onetwo to decide what will
c                          be done  to determine these values.
c
c
c
c  nk_rhomax               if fit_rhomax_type .eq. "spline_icsvku"
c                          then nk_rhomax is the number of knots
c
c  nk_psi                  if fit_psi_type .eq. "spline_icsvku"
c                          then nk_psi  is the number of knots


c  icsscv_ijob_psi          fit type parameter used only if
                        fit_psi_type ="spline_icsscv"
c  icsscv_ijob_rhomax        fit type parameter used only if
c                         fit_rhomax_type ="spline_icsscv"
c                          both of these are IJOB parameters:
c                         IJOB  parameter in icsscv call
c                         from IMSL description:
c                       - JOB SELECTION PARAMETER. (INPUT)
c                        IJOB = 1 SHOULD BE SELECTED WHEN
c                          NX IS SMALL (LESS THAN ABOUT 20)
c                          OR WHEN UNEQUALLY SPACED ABSCISSAE
c                          (X(1),X(2),...) ARE USED.
c                        IJOB = 2 SHOULD BE SELECTED WHEN
c                          NX IS LARGE AND THE ABSCISSAE ARE
c                          EQUALLY SPACED.

c  pol_flux lim           normalized  value of poloidal flux to use
c                         for plasma boundary

c output_file_name        Name for netcdf file to be created


c plot_file_name          Name to use for cgm plot file output


c input_file_name          name of file that contains list of eqdsks to
c                          process
c
c dump_file_name          All the data that was caluated will be dumped to this
c dum_values              ASCII file if dump_values = .true.
c
c
c
c---------------------------------
```

c analysis_check if =1 (which is the default) the c profiles run in analysis mode are included in the c check for the relative maximum change, see relmax. c This means that the time step might be cut back c even if we are not solving the corresponding c diffusion equation. This is useful for checking the c consistency of the time dependent profiles that are c specified in inone. If this effect is not wanted c then set analysis_check = 0

H Glf23 modeling

The two previous versions of GLF23 were removed from Onetwo and replaced with one new routine. As uasual an input description of the available options that can be set in the Onetwo input file,inone, is given in file cray102.f . There are several new options as follows:

glf23_iglf Determines if old (=0) or new (=1) version will be run. The "old" version is the one prior to Feb. 2003 which had some problems with reversed shear discharges. The new one is version 1.6 which is an intermin fix for reversed shear cases. (It is expected that version 1.6 will eventualy be replaced by a more refined model) The code defaults to using version 1.6.

write_glf_namelist Valid settings are 0,1,2,3. 0 means do not write a namelist (in file glf23_namelist) . 1 means do write the namelist. This namelist is a convenient way of getting a single time slice of information from onetwo into the glf23 stand alone code (as released to the NTCC). In that stand alone code the executable testglf can read this namelist (after changeing the name from "glf_23_namelist" to "in" and setting lprint = 1). In this mode only the results from the most current time are saved. Note that the file glf_23_namelist only contains input for Glf23. The output from Glf23 is obtained from the file witten by the testglf code by setting lprint =1. (There is also a debug option, see the switch glf_debug in cray102.f, that includes output from Glf23) write_glf_namelist = 2 means write the namelist and terminate Onetwo right after the very first time that glf is called. write_glf_namelist = 3 write the namelist and terminate Onetwo at the start of the inital time step. This differs from the write_glf_namelist =2 option in that all sources (rf,beam) are called at the initial time first, which changes some input into glf such as the electron density due to beam efects.

glf23_ncpus The number of cpus to use in Glf23 calculations. Valid only on multiple cpu machines with a proper intallation of mpi. At this time the messsage passing overhead is too great (see the table below) for this option to be used.

The stiff non linear behavior of the GLF23 confinment model (and others sucah as Wiland) requires much more computational effort than previous models. To deal with this problem two additional solution methods were created in Onetwo. The first is an adaptive method of lines approach whereby only the spatial derivatives are replaced by finite difference forms. The resulting set of coupled equations are then ordinary differentail equations in time and can be solved by "black box" ode solvers . Onetwo uses the Radau5 package whcih allows variable variable coefficients for the time derivatives. It was found that this approach

The second new solution method introduced into Onetwo is a globally convergent combiantion of steepest descent,trust region, and Newton type solvers using line search, dog leg and hook step methods (familiar from the numerical solution of sets of non linear equations). This arsenal is required to solve the resulting set of non linear coupled algebraic equations that are obtained by finite differencing both the space and time derivatives in the diffusion equations. One major advantage of this techmique is that it allows us to run both time dependent and time indepenent cases.In he time indepedent case we solve the diffusion equations with the time derivative explictely zeroed. The resulting set of non linear algebraic equations may not have a solution, reflecting the physical fact taht the current set of sources do not lead to a time indepdent solution. But our solution method is capabe of handling such situations by using descent methods whereby the residula sum of ssquares of all the equations (one eqaution for each variable,and for each grid point) is minimized. Hence even if we do not find a solution converged to the usuall degree (typically ¡ 1.e-8),we will still get the best approximate solution and a measure of how much this solution deviates from the expected true solution. Quite often this is enough information to allow manual modification of the problem in order to generate an acceptable result. An obvious application is the determiantion of steady state current drive situations. Rather than evolve the system for 100 sec or more to relax the current density sufficiently (for Diii-D) we instead step directly to the equilibirum solution. (Of course it may take a signifcant number of iterations to solve even the time indepednet equations)

The interested reader can find more information and examples regarding these solution techniques in the detailed Onetwo writeup (
u
stjohn
)

Note that Glf23 assums that the diffusion equations are of the form

with no convection. In Onetwo the corresponding equation is

and hence we must apply the correction factor. This is done by solveing the equations with an effective chi given by

in Onetwo. To compare diffusivities with Xptor we must keep this effect in mind. The output from Onetwo prints and plots the effective chi which will differ from the Xptor chi by the indicated factor.

No. cpus	Taurus	Luna	Lohan1
	sec	sec	sec
1	.06	0.21	
2	.14	0.21	
4		.13	
6		.13	
8		.11	
10		.10	
12		.11	
14		.10	
16		.10	

Table 2: CPU Time for 51 Point GLF23 Calculation

H.1 Parallel Glf23 Computations

A parallel version of the stand alone GLF23 test code was created to evaluate the feasibility of using the distributed memory message passing interface (MPI), or the shared memory Open MP scheme to speed up the calculations. Open MP has the huge advantage over MPI of allowing incremental code parallelization whereas MPI requires that the entire code be parallelized before it can be used. Unfortunately our local computing environment is limited to 2 cpus for shared memory situations and hence there is little to be gained in using Open MP. Open Mp directives are included in some relevant parts of Onetwo but no signifcant gain in execution time can be achieved using only 2 processors. (The 8 processor GS80 shared memory machine ,gaws21.sd.gat.com, is the exception but it is not generally available for production Onetwo runs).

To determine what could be done with MPI a version of the stand alone Glf23 code was run with MPI parallelization done over the 51 point rho grid internal to subroutine callglf2d. The results for various combinations of machines and processors is given in Table I.

The message from this table is pretty convincing. By comparing the single cpu results across machines we see that commidity Athlon and Xeon processors are operating at speeds that make it difficult for the slower multiple cpu machines to compete. The bottleneck is in communications amongst the processors. In this example there are two places where communication is required. First, the master proces reads the input data and then broadcast the data to all the slave processors. The alternative would be to have copies of the input file on each slave machine and let each slave read the data directly. The reason that we dont do this is that it takes much longer to ship the file to each processor and read it than it does to broadcast it from a single processor. Second, after each processor has done its part of the calculations (in our case each processor has calculated the diffusivities on some subset of the rho grid) the results have to be collected and combined into a single set of arrays on the master process. The combined communications costs of these two bottlenecks is so great in this example that even 16 processors on Luna cannot beat the single processor results on Taurus. It was found that the broadcast of the input data to all the processors was very roughly constant at about 25 msec for 8 or more processors. The 2 cpu case, where communication is localized, takes much less time of course (5 msec) . The elapsed time to do the computations on the 51 point grid is about 100msec for 2 processors and decreases to a constant 20 msec for 10 or more processors. The final reduction of the data back to a single cpu (done with multiple calls to MPI_REDUCE) take the majority of the time and increases slowly from 75 msec for 2 cpus to about 120 msec for the 16 cpu case. All of these timmings are on Luna and they fluctuate substantially as the system load changes. However more precise averages are not needed to support the conclusions reached here. Based on these observations we see that even if Luna had faster processors the results would not improve greatly since it is the data reduction phase(using MPI_REDUCE) that is the primary problem. Hence the Glf23 computations are in fact not very well suited for parallelization on Luna at least. A faster communication mechanism is needed in order to improve this situation. Alternatively we have to do more computations on each individual processor in order to increase the ratio of computation to communication times. In the present example this would be achieved by using 201 instead of 51 grid points. This would make the computational time comparable to the data collection time.

Based on these observations there is little incentive to pursue parallel computations of the Glf23 confinement model in Onetwo. Much larger sections of the code will have to be run in parallel before using MPI makes any sense. As far as Onetwo is concerned, the MPI approach seems limited to Monte Carlo beam calculations and rf ray tracing, both massive,embarassingly parallel problems.

H.2 Problems

Unfortunately there are hardware/compiler,software dependent problems that necessitate this section.

Known problems are:
(1) Preplt will not run on Katze due to insufficient memory. Onetwo will run but you must select run_preplt = .false. in the first namelist of inone. Note that if you do not do this Onetwo will take an error exit but the files will still be created. Hence you can move trpltfil to another machine and run

preplt there. (2) The MEPC code uses Display for plotting which limits the places you can run it to Hydra and ?? (3) The fastwave code current drive calculations fail when nzrffw = 2 and nzrffw ¿ 6 . (4) Currently the current drive calculations in Toray v1.4 do not agree when Toray is run on Hydra and Cardea

I Vloop Boundary Condition

A boundary condition that allows specification of the plasma loop voltage as a function of time instead of the total current was introduced into Onetwo starting with version 3.4 . The relevant input parameters are input in the first namelist of inone. The options are;

vloop_bc(1,..,kbctim),volts : Is an array of loop voltages(at the plasma boundary). If only one value is given then it is assumed that vloop_bc is constant in time. Vloop_bc takes precedence over totcur!! Note that as the resistivity changes vloop will drive different amounts of ohmic current and hence the total current will float. To use this option specify at least one value of vloop_bc in the first namelist of inone.

vloop_bc_time(1,...kbctim) ,sec: vloop_bc can be given in bctime(1,..kbctim) or it can be given on a separate time base, vloop_bc_time. vloop_bc_time has the same maximum length as bctime (ie kbctim) the code will detect how many valid entries there are in vloop_bc_time. The vloop_bc_time array must be in 1:1 correspondence with the vloop_bc array but the two arrays do not have to be in any special order. (Onetwo will time sort both of the arrays according to monotonically increasing values in vloop_bc_time) If vloop_bc_time is not used then vloop_bc must correspond to the elements in bctime and bctime itself must be ordered monotonically increasing in time (bctime is not time sorted). Obviously vloop_bc_time must be a superset of the start and end times of the analysis (eg. time0,timmax).

vloopvb set to 1 to get special monitoring output to file vloop_monitor.txt. This is used primarily as a debugging aid. (this file could get to be quite lengthy for a long time run). The file can be read/plotted with readvloop.py. But note that many of the local machines do not have the Biggles plot package that readvloop.py uses. You can do one of three things:

1. Install Biggles (very easy if Python is available)
2. rewrite the plot calls in readvloop.py to your graphics package
3. just delete the lower half of vloop.py and use it only to read the data (and then pass it into IDL for example)

The method used to introduce vloop_bc into the boundary condtions is as follows. This method was used because it is consistent with other requirements in the code where the boundary is not necessarily at $\boldsymbol{\rho} = \mathbf{1}$. At the initial time the ohmic current (or parallel electric field) must be known throughout the plasma. In Onetwo we continue to obtain this profile in the standad manner. That is, the total current, usually taken from the eqdsk, is used as the boundary condition for the startup guess to generate an ohmic current profile consistent with the current drive models that are active at this time. This supplies the starting ohmic current profile for all grid points except the point at the very edge of the plasma. To get curohm at the edge we use the condition

$$\boldsymbol{V_l = 2\pi R_0 E_0 = \eta H < \frac{\vec{J}_{ohmic} \cdot \vec{B}}{B_{T0}} > |_{rho=1,new}} \tag{265}$$

where $\boldsymbol{V_l}$ is the input supplied loop voltage (ie vloop_bc) at this time. The total current is then adjusted to reflect this new value of the edge ohmic current by adjusting the total current density:

$$\boldsymbol{< \frac{J_\phi R_0}{R} > |_{rho=1,new} =< \frac{J_\phi R_0}{R} > |_{rho=1,prev} - < \frac{\vec{J}_{ohmic} \cdot \vec{B}}{B_{T0}} > |_{\rho=1,prev}} \\ \boldsymbol{+ < \frac{\vec{J}_{ohmic} \cdot \vec{B}}{B_{T0}} > |_{rho=1,new}} \tag{266}$$

The new current density,EQ[266], (which differs from the old current density only in the edge value) is then integratd to obtain a new total current:

$$\boldsymbol{I = 2\pi R_0 \int_0^1 < \frac{J_\phi R_0}{R} > H\rho d\rho} \tag{267}$$

Finally this value of I is used as the actual boundary condition for the $\boldsymbol{\rho F G H B_{P0}}$ profile to advance Farady's law in time. Some iteration is required to achieve consistency. In the above formulae we have used $\boldsymbol{\rho = 1}$ symbolically to indicate the plasma edge ($\boldsymbol{\rho}$ actually has units of cm and ranges from 0 to some maximum value depending on the enclosed toroidal flux).

An example of the output is given in Fig(12). In part a of the figure the total current is plotted as a function of time. The current increases even though the edge current density, part b , follows the input loop voltage(part d). The edge ohmic current density(as given by Eq.(265) is shown in part c .

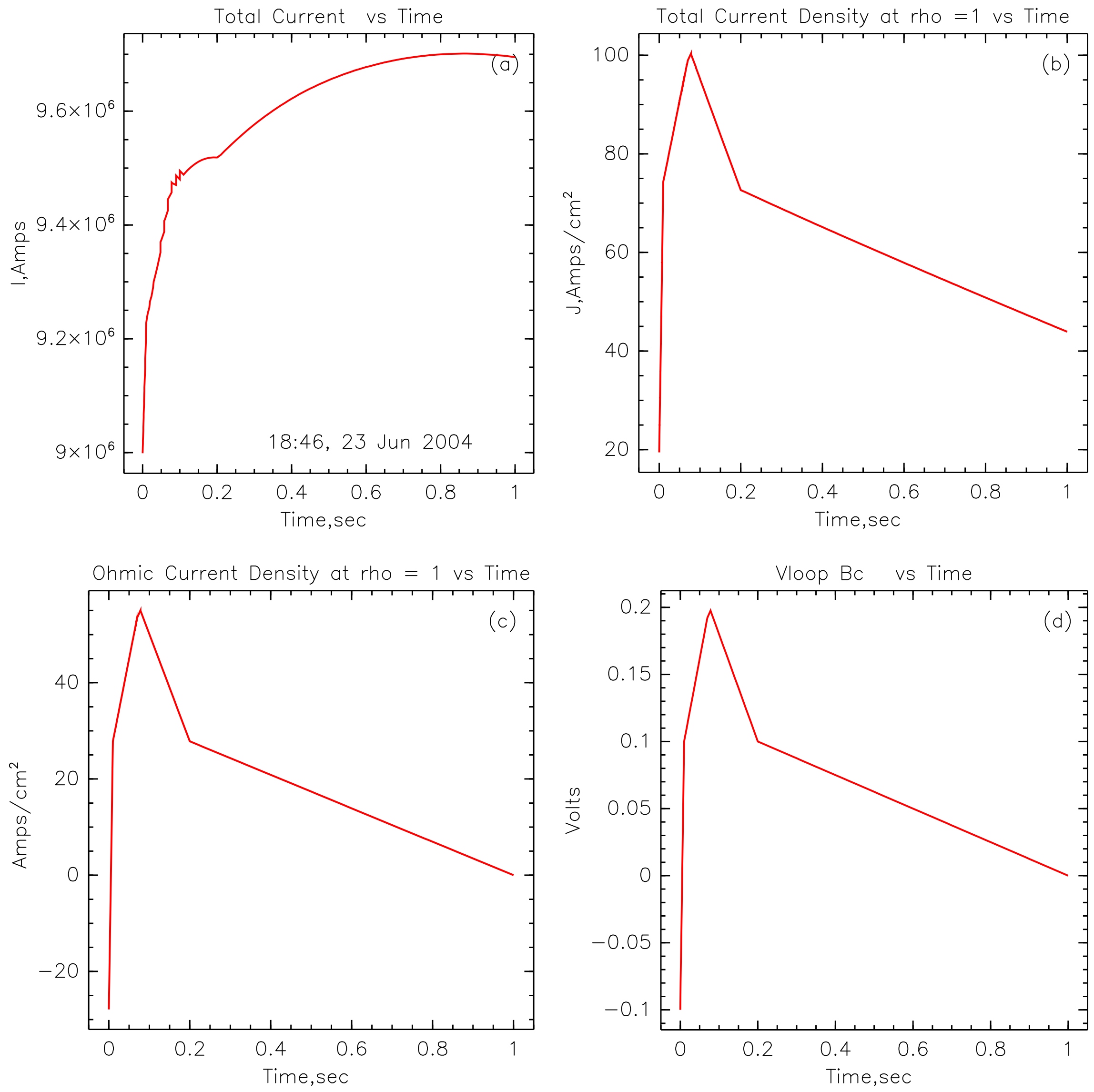


Figure 12: Example of using the vloop boundary condition for an ITER AT type discharge

J Running Curray in Onetwo

Funding provided by the National Transport Code Collaboratory (NTCC) enabled us to install and review the ray tracing code Curray at General Atomics. The project consisted of building an interface to the Curray code in Onetwo and subsequently testing the Curray code with L and H mode DIII-D discharges.

We found that there were two different ways that profile information is passed to Curray from Onetwo. The first method, exemplified by the NTCC version of the code, used input from an (undocumented) file onetwo.out. The updated version Curray obtained from TK Mau uses files masanori.in, bdens and ebeam_d to pass thermal and fast ion profiles to Curray. Furthermore, for other transport codes (eg Transp) a different file, trxpl.out, is used for this purpose. Since the information that is passed to Curray from a transport code is not specific to a transport code we decided to use this opportunity to unify the transport code interface. Consequently the interface described below is intended to be general enough so that other transport codes wishing to couple to Curray can use this method. Accordingly we have incorporated the existing Transp interface (using file trxpl.out) in our scheme. The method of profile input using files masanori.in, etc. was retained (as an option) in the version of Curray now installed at GA. We did not create a code to write files in the format required for onetwo.out. This means that the existing NTCC version of Curray can not currently run the DIII-D test cases presented below. Note that other (non Onetwo), users of the code will not notice any change in the interface since it was created in such a way as to be transparent.

The interface to the Onetwo transport code required that some minor changes and new features were added to the Curray code itself. The changes in Curray are clearly delineated by the "!HSJ" sentinel present in some sections of the code. To eliminate confusion and for easy reference we present here an itemized list of the changes made.

- Curray was moved to our CVS repository /c/cvsroot and all future changes to the code should be derived from that repository. As mentioned above the starting point for this repository was updated version of Curray obtained directly from TK Mau.

- Curray will now accept the name of an input file on the execution line. This allows proper manipulation of the Curray run while being controlled by a transport code. The default input file name remains curray_in and hence if the name is not supplied at the command prompt then the old behavior is recovered.

- The name of the eqdsk to read can now be supplied as input in the namelist data in the input_file. The default name of this eqdsk remains eqdsk.hb so that the old behavior is recovered if no explicit name is given in the namelist input. This feature was added to allow simulations where mhd evolution is taking place.

- The default eqdsk size was made 129x129 instead of 65x65 since the larger size is what is routinely used at DIII-D for mhd coupled transport calculations. The file param.f90 must be changed and the code recompiled to affect this change since dynamic sizing of eqdsks is not implemented at present. This also means that two versions of the Curray executable have to be maintained to accommodate the mhd grids. The transport grid passed to Curray is governed by namelist input parameter nprof and as long as the grid size is less than nprof (= 133) no change is required for it.

- To allow investigation of the sensitivity to various fast ion components the number of ions (parameter nions in param.f90), was increased to 12 to accommodate fast ion species. Each beam line and injection energy is assumed to be a separate fast ion species. For the typical DIII-D modeling in Onetwo this means that the two injectors and 3 energies give rise to six fast ion species. In addition fast alphas can be present as well. At the present time each fast species is assumed to have a density and temperature profile given by two thirds of the stored energy density divided by the fast ion density. This feature is available only by creating the Curray input files using the Onetwo code. An option in Onetwo will collapse all the fast ion components into one effective component thereby recovering the more usual mode of operation of Curray.

- The new Onetwo interface eliminates the second eqdsk file, eqdskex, and the spectrum input information file, raytrin. Both of these files were intended for use only with Onetwo and hence will not affect other users. The information that used to reside in raytrin is duplicated in curray_in and the information in eqdskex is duplicated in the TRANSP interface file trxpl.out. The meaning of the namelist switch ioread in Curray is thus modified accordingly. Ioread =1 now means create raytrout only. Its previous association with raytrin is no longer meaningful. Onetwo was changed to create file curray_in (including the spectrum information) instead of file raytrin. File trxpl.out, also created by Onetwo now contains all the kinetic information (eg Te,Ti,ne,ni,nfast,etc). The format of trxpl.out was not changed so that it will still work with Transp as well. The meaning of the Curray namelist input switch iprof was changed to accommodate the trxpl.out file input from Onetwo. Previously iprof =1 meant use Transp input file trxpl.out. Now iprof=1 means use file trxpl.out, no matter what the source of that file is (eg Transp or Onetwo).

- The GA version of Curray was modified to “correctly” handle a problem with EFIT type eqdsks. An unfortunate convention in these eqdsks causes a sign problem in the driven current. This modification currently applies to all eqdsks and needs to be generalized for NSTX and other machines.

On the Onetwo side of the interface three files, curray_in,eqdsk,and trxpl.out are created by Onetwo each time Curray is run out of Onetwo. File curray_in is the namelist input file, required for all types of Curray runs, which includes the necessary antenna spectrum information (the curray_in file is documented in file curray_input, which is part of the Curray distribution). The actual name of the curray_in file is determined at run time by Onetwo. File eqdsk is a standard EFIT type equilibrium file whose actual name is also determined at run time.

The spectrum and power density information for Curray is assumed to be present entirely in the Onetwo input file inone. An example of this input for DIII-D is given in the appendix B for 60,83 and 117 MHz operation. This information is basically identical to the information normally present in file curray_in (for a single time slice ,input power,and spectrum) and the description of the namelist parameters in curray_in serves as the input description in file inone. However the Onetwo input file inone is actually a superset of the Curray data to allow for changes in time in the spectrum and input power and to allow for several different antenna specifications simultaneously. This is the purpose of the extra indices that will be found in the parameters in inone(see appendix B). Finally a file that contains the thermal and fast ions densities and temperatures, trxpl.out, is created and then Curray is spawned. This process may be repeated many times as the transport simulation proceeds. An option in inone allows the saving of the curay_in, eqdsk and trxpl.out files so that a stand alone Curray run can be repeated at a later time. Since Onetwo will write the curray_in file it is never necessary to create that file by hand. Instead the information should be placed into inone directly. At any given time Onetwo will spawn Curray repeatedly until all Curray cases have been done. The results are accumulated internally in Onetwo. The Curray models can be freely intermixed with other heating and current drive models. For example in appendix B three Curray cases are defined based on frequency, power input, etc. These three cases could be made active simultaneously by specifying equal times (rfon) for which the models are active. Alternatively any combination of models (fast wave, ech, etc.) is selected using this method.

Curray creates the files currayout and raytrout. The former is a standard Curray output file meant for user inspection. The latter file,raytrout,is read directly by Onetwo. We note that one of the planned enhancements of Curray mentioned on the NTCC web site is “Enhance code capabilities to handle more than one antenna spectrum either by source modification or by a shell script to do multiple CURRAY calls”. The Onetwo interface described above does this already and we expect that most other transport codes (eg TRANSP) also have machinery in place to account for multiple Curray cases.

J.1 Curray Benchmarking

We ran both the modified local version of Curray and the NTCC version on the ITER and NSTX test cases supplied with the NTCC Curray distribution. Additionally two new DIII-D test cases were added to broaden the range over which Curray has been successfully applied.

The first two test cases examined are the ones supplied with the NTCC distribution of Curray. The object of this testing was to verify that the GA version of Curray yields results comparable to those obtained with the NTCC version. The NSTX test case given on the NTCC web site [8] has to use the adjoint method of determining current drive efficiency (which required a change in the supplied input file). It was observed that the calculations take significantly longer when this option is used. Since execution time becomes an issue when Curray is called repeatedly from a transport code guidelines should be evolved to automatically determine when the added complication of the adjoint equations is required and how often the tables have to be recalculated during a transport simulation. At present the Onetwo interface does not take these matters into consideration.

The results are show in Fig(13) for both the low aspect ratio NSTX case and the ITER case. For ITER the driven current should be calculable using either the Karney adjoint formulation of current drive efficiency or the Ehst-Karney small inverse aspect ratio approximation. As indicated in Fig(13 b) for the latter case the total driven current is about 51 kA. When the calculation is repeated using the adjoint method this result drops to about 44 kA. This is perhaps a larger discrepancy than one would expect for this machine. As is evident from the figure the results for the GA version of Curray and the NTCC version are indistinguishable. Hence we may confidently carry out further testing using the more recent GA version only.

As previously remarked we were not able to test DIII-D cases against the Curray version on the NTCC web site due to differences in the way Onetwo data is read by these two versions of the code. This situation is independent of the new Onetwo interface and exists due to the development of curray that has taken place since the NTCC version was released. To test Curray with DIII-D discharges an H mode (shot 111221) and an L mode

Shot 84293
(L mode)

FW MHz	$\boldsymbol{P_e, kW}$	$\boldsymbol{P_i, kW}$	$\boldsymbol{I_{CD}, kA}$	$\boldsymbol{P_a}$
60	501	499	56	41 %
60	491	509	51	80 %
83	744/607	256/398	80/71	31%
83	728	272?	74	80%
117	950	50	111	29%
117	895	105	99	82%

Shot 111221
(H mode)

FW MHz	$\boldsymbol{P_e, kW}$	$\boldsymbol{P_i, kW}$	$\boldsymbol{I_{CD}, kA}$	$\boldsymbol{P_a}$
60	340/287	660/773	22/16	95%
83	420	580	30	99%
117	395/212	605/788	30/18	99%

Table 3: Heating and Current drive results for DIII-D L and H mode cases.The second number for electron,and ion absorbed power and current drive indicates results obtained using Transp profiles. $\boldsymbol{P_a}$ is the percent of injected power absorbed. For the L mode shot results are quoted for 6 and 100 edge reflections (with the larger value of $\boldsymbol{P_a}$ corresponding to 100 edge reflections and the smaller value to 6 reflections)

(shot 84293) case were examined. The kinetic data for these two shows is given in Fig.(14). The spectrum for 60,83 and 117 MHZ was obtained from TK Mau (H mode) and from Craig Petty (L mode). The total electron and ion absorbed power and current drive values for these discharges using the six lobe antenna spectra given in appendix B are summarized in Table I. It was previously established [9] that six edge reflections lead to rf power absorption values greater than 90% for high $\boldsymbol{\beta_e}$ discharges. Thus that number has become a de-facto standard. However as is seen in Table I for the L mode case 6 edge reflections are not sufficient. For the L mode shot we ran a second series of cases where 100 edge reflections were allowed. This boosted the absorbed power to about 80%. A further increase in allowed reflections only slowly increases the absorbed power as it asymptotically approaches 100%)

The cases shown in the table and in the figures below all used 1MW of fast wave input power,an antenna location of 1 degree, and 66 rays in six spectral power lobes (see appendix B). It was found that allowing six edge reflections in the ray tracing caused most of the power to be absorbed for the H mode case. However as indicated in the table the absorption for the L mode case tends to be weak unless the number of edge reflections is increased significantly (maxref = 100). $\boldsymbol{P_e}$,$\boldsymbol{P_i}$, $\boldsymbol{I_{CD}}$ are all quoted on the basis of 100% power absorption. Hence $\boldsymbol{P_e + P_i = 1MW}$ for all cases shown.

Some of the profiles associated with the results given in Table I are presented below. To begin with we examine in Fig(15) results obtained using Transp and Onetwo as the transport code drivers for Curray. To generate these results the profiles of densities and temperatures obtained from Transp were used in the curves corresponding to Onetwo. Hence we find that both the electron and ion heating and current drive are very similar. The small observed differences are most likely due to the difference in equilibrium files used. For Onetwo an eqdsk of size 129 x 129 was required. The Transp results use a 65 x 65 eqdsk. As is shown below in practice, due to different transport grids, and especially treatment of the fast ions, the difference between Transp and Onetwo simulations will be larger.

The actual heating and current drive profiles determined by using Onetwo derived fast ion density and effective temperature profiles for the case given in Fig(15) is shown in Fig(17). In addition to the usual single effective beam model this figure also shows the results of treating each individual beam component as a separate species. Since each beam component will have a different fast ion stored energy distribution the effect on Curray was of interest. As shown in the figure however the results are only slightly different when the multiple beam model is used. Due to the non linear nature of the dispersion relation that Curray must solve we find that there is a 10% difference in electron heating and current drive but only about a 5% difference in ion heating (which includes the fast ions). In parts (c) and (d) of the figure a comparison of fast wave injection at 60 and 83 MHz is made. The higher injection frequency favors electrons at the expense of ions and increases the driven current. Of particular interest in Fig(17c) is the change in the ion heating profile in moving from 60 to 83 MHz fast wave injection. For both frequencies most of the absorption in the ion channel is due to the fast ion contribution (at 60 MHz we have 543 out of 660 Kw of ion heating due to the fast ions and at 83 MHz the corresponding numbers are 492 out of 580 Kw). The harmonics that contribute to the results are shown in Fig(16a,b). At 83 MHz it is the 6'th harmonic of deuterium at rho =0.06 (high field side) that causes the high $\boldsymbol{(0.79 w/cm^3)}$ absorption peak. A smaller secondary peak at rho = $\boldsymbol{0.5}$ is due to a combination of 5'th (high field) and 7'th (low field) harmonics for this case. At 60MHz the 4'th harmonic of D at rho = 0.26,(high field side) leads to the sharp peak in the fast ion heating profile (see FIG(17a,b). A second peak at rho = 0.43 is due to the 5'th harmonic of D (low field side). This explains the broad absorption profile compared to the 83 MHz case as the waves must sample larger volumes of the plasma in order to encounter both the low and high field side resonances. For the 83MHz case the high fast ion beam temperature near the axis is more than able to overcome this effect. In both cases there is little qualitative change in the electron heating profile due to Landau damping and TTMP effects.

As was shown in Fig(15) the Transp and Onetwo results track closely when identical profiles are used.In Fig(18) we show results when this is not the case. Here the fast ion distribution from Transp is sufficiently different that the ion heating and current drive are significantly different. It is seen in the figure that Onetwo puts more energy into the electrons and less into the ions. This also increases the driven electron current. The ion absorption in Onetwo is less due to the lower effective temperature of the fast ions species. In Fig(18c) the thermal ion (D and minority species H) and fast (D) ion absorption for the two cases is shown. Most of the ion absorbed power in fact appears in the fast ion species (dashed lines, Fig(18a) inside of rho = 0.2. As seen in the figure the primary and minority ion species contribute relatively little. In Fig(18d) the effective fast ion pressure is shown for the two cases. The difference inside of rho =0.2 is substantial and is responsible for the higher ion absorbed power (788 Kw) obtained using the Transp derived profiles compared to the Onetwo result (605 KW). Due to the relatively small volume over which the effective fast ion temperature is significantly different in the two codes the overall fast ion energy content is almost the same. Hence we see that the Curray results can be quite sensitive to details of the fast ion distribution if fast wave absorption is localized.

Our second DIII-D test case uses the L mode discharge, shot 84293@2110 msec. For 60MHz fast wave injection the results using Onetwo as the driver for Curray are shown in Fig.(7). The corresponding resonance surfaces for this frequency as well as for 83 MHz is given in Fig.(8). Examination of Fig(8a) shows that there is a strong fourth harmonic resonance (4D) for deuterium near the magnetic axis at 60 MHz injection. This causes the sharp peak in fast ion absorption observed near $\boldsymbol{\rho = 0.16}$ in Fig(7a). The 3D and 5D contributions are also visible as small peaks at $\boldsymbol{\rho = .3}$ and $\mathbf{0.45}$ respectively.

Our final example consists of the L mode discharge at 83 MHz. In Fig.(21) the results between Transp and Onetwo driven Curray are compared. The situation is quite similar to the H mode case shown in Fig.(18). Due to the larger stored energy density for fast ions near the magnetic axis Curray predicts higher ion heating and lower electron heating using the Transp Monte Carlo derived beam profiles. The 5D beam ion resonance near $\boldsymbol{\rho = 0.07}$ is clearly visible but the electron heating due to Landau damping and TTMP effects is quite significant and leads to a higher driven current than was observed in the previous cases. The decrease in electron power absorption near the magnetic axis, clearly visible in Fig.(7a) has also been observed in full wave calculations [7]. But the full wave calculations are preliminary in nature so we can not include them in this report.

When this shot is run at 117MHz the results are as indicated in Fig(22). In this figure we have also included the heating and current drive profiles allowing for a maximum of 100 reflections of the wave. As is seen in the figure most of the power is absorbed by the electrons which increases the driven current. Even though the wave direction can change significantly with 100 reflections the current drive decreased only by about 10%.

Curray follows the recommended standards given in refb4. In particular we note that the necessary source code together with drivers for two test cases with input and output documentation are currently available.

J.2 Standards, problems and recommendations

Curray follows the recommended standards given in **??**. The necessary source code together with drivers for two test cases with input and output documentation are currently available. We found that the code can be built under OSF F90 5.4, HP UX11.0 f90, and Linux(RH7.2,8.0,Pgf90). However Curray will fail with a floating point error on the OSF machines for some cases which work under Linux and HPUX. Further investigation will be required to determine the cause of this problem. The Makefile was modified to allow building of the code on all local platforms. We did not change the basic structure of the Makefile that allows the build to continue even when errors have occurred. However we find this approach makes building the code more obscure than it needs to be since errors encountered during the build do not terminate the build process. This forces the user to backtrack through many obscure lines of output generated by the makefile to find out where the error occurred. It is much easier to work through the build process one error at a time until the build succeeds. One possible workaround is to at least indicate to the user how the makefile may be used to write to std error piped to a disk file. Then working through the std error file from the top down will achieve the same result. The unused "customization " files present in the distribution should be incorporated into the mainline code (and then selected through input switches) or removed from the distribution altogether. Otherwise the concept of a single, tested version, of Curray becomes less obvious. We recommend that version numbers be attached to the code so that reference to a given version is uniform across all users of the code.

A readme file (curray_readme) together with CurrayDoc.pdf and curray_input provides the necessary documentation. As is required, no graphics is embedded in Curray. A file suitable for graphics generation is written (rayop) and may be processed using an open source graphics library (pgplot) together with the driver curplot. The functionality of curplot was found to be adequate but no extensive testing was undertaken.

In building and running Curray we encountered some minor problems. Interpretation of the validity of results and proper input settings for the code also initially caused us some difficulty. Below we detail some of these issues and offer remedies as appropriate.

Using the files masanori.in, bdens and ebeam_d is awkward and error prone in our opinion. The reading of these files is initiated by the switch gamene =0 and iprof =0. This is confusing since iprof =0 implies that analytic profiles will be read but then gamene.ne. 0 is used to read in tables of profiles from the above files instead. The GA version of the code now allows this input method to be used optionally but we recommend that the new Onetwo interface described above is used in the future. This will require a new identification scheme for labeling fast ions in file currayout. The file curray_input which documents the namelist input parameters in curray_in should be added to the NTCC distribution.

Curray currently has a nice f90 style to it. However the modules are using a mixture of f90 and f77 syntax. Not all compilers have option flags to accept such a mixture. Since modules are an f90 construct it would appear to us that using f90 only syntax in modules makes sense. We think that this should be a standard that NTCC contributed codes should try to achieve for sake of cross platform compatibility.

We find that the recompilation of the code for different eqdsk sizes is awkward. Although this feature is present in most codes that we know about, Curray is in a better position to ameliorate this problem. Using the present version of Curray it is possible to specify the name of the eqdsk in the namelist of the very first file (eg curray_in) that is read (or the default name eqdsk.hb is used). Hence the eqdsk file can be opened at that point, the size of the eqdsk extracted, and then the arrays can be allocated dynamically at run time. All of this can be done at the very start of the program in (rfdrive.f90).

Limits of arrays in the curray_in namelist inputs must be given in the documentation. For example, what is the maximum allowed number of ion species nspec? One may get the mistaken impression that dynamic arrays are used to accommodate user input. For most variables that is not the case however. Also there are subtleties involved in the input of the antenna spectrum. We find that most users (including ourselves) have insufficient experience to supply this data without further help from a more detailed description of the required input variables.

Certain features to improve the smoothness such as ray launch randomization using a Gaussian ray generator or other appropriate means should be included. Importance sampling could be used to better map out regions of high absorption. In Fig(13a) for example we find spikes in the ion heating and current drive profiles. Such features become particularly troublesome when non inductive current drive studies are performed using a transport code driver.

There is redundancy of information across input files. For example trxpl.out and curray_in, have variables nprim,nimp,nspec,etc. either the redundancy should be removed or a clear indication of precedence rules must be given in the input instructions. Some switches,eg icurdr, actually are used to create files (eg adjin). These switches and the name of the files they create should be spelled out in the input instructions. Finally assignment of Fortran io unit numbers in the namelist input file does not seem reasonable (its purpose is not clear).

When used within a transport code the definition of the driven current as some sort of flux surface averaged quantity is important for accurate current drive calculations. The definition of this quantity needs to be made precise in Curray. The electron driven current in currayout is incorrectly labelled as Ma. The actual value given is in amps/watt of injected power. We should point out that for the DIII-D discharges presented above Curray is quite sensitive to the initial launch conditions employed for the rays (in addition to the edge reflections problem). We found that both the number of poloidal starting points per toroidal refractive index,nthin, and the radial starting location of the ray in terms of poloidal flux, psi0, influenced the results significantly. We would recommend that features are added to the code to take care of these issues,perhaps by doing a preliminary scan to set parameters if time dependent transport run is to be done. Most of the cases run encountered some sort of (non fatal) error during execution. For example "no solution of dispersion close to raypoint" is one such error but there are at least several others. It is not reasonable to expect a non specialist user to cope with such errors. We need to have an automatic assessment and fix for such failures in the code. A code such as Curray which is intelligent enough to trap these errors can most likely be made to take remedial action as well(eg perhaps a more robust but more time consuming non linear method must be invoked when an error condition is sensed). Relatively simple errors such as "starting point too close to the p-cutoff" or absorbed power too small should be eliminated altogether since checking for such errors during a transport run is not feasible.

Although not explicitly shown in this report we did scan the effect that different equilibrium file grid sizes have on Curray. The ray tracing is found to be slightly different in such cases but the effect is not great for DIII-D provided reasonable grid sizes are used (eq. 65x65 or greater). However we did encounter situations where changing eqdsk size caused Curray to fail. We did not determine the specific cause of these failures.

J.3 Curray input

In this appendix we present the actual antenna spectrum and other curray related inputs used for the cases presented in this review. Note that 60 ,83 and 117 MHz cases are included. The correct case is selected by selecting the second index equal to 1,2, or 3 respectively.

```
!--------------------start curray specific input ---------------------

! curray_path = 'SOME PATH TO YOUR FAVORITE CURRAY' DON'T SET TO USE DEFAULT
save_curray_input = 0 ! =1 SAVES ALL CURRAY INPUT FILES
 psistep=0.025       !step in psi for ray tracing
 pkexpnt=0.001
 igraph=1            ! 0 NO GRAPHS FROM CURRAY ,1 CREATE GRAPHICAL OUTPUT FILE
 iprint=0            ! -1 CONDENSED,0= NONE,1=NORMAL,2=MORE , 3 = WAY TOO MUCH
 incrt = 1           ! number of harmonics in ion  damping is 2*incrt +1

epserr=0.02          !ERROR LIMIT FOR DISPERSION RELATION
epser1=0.004         !SECOND ERROR LIMIT FOR DISPERSION RELATION
idcur=1              !1 ANALYTIC CUR DRIVE EFFICIENCY, 3 = ADJOINT CALC,2 = DONT USE THIS
nminor=0             !NUMBER OF MINORIY SPECIES
kalfa=0              !0 IGNORE FAST ION DAMPING (APPARENTLY RELATES TO SLOWING DOWN DISTIRIBUTION)


!nspect=6       !NUMBER OF TOROIDAL WAVE NUMBER BINS- SET BY  POWERSRT
nrayptrt=6000

psi_startrt(1) =  3*0.985   !starting value of psi for rays
thgrilrt(1)=6*1.0   !approximate central location of antenna in degrees
                     !w.r.t. outer equatorial plane
powersrt(1,1)= 4.8830e-3, 1.4940e-2, 3.7920e-2, 1.4220e-1, 3.6820e-1, 2.21852e-1
powersrt(1,2)= 2.2948e-1, 4.9065e-1, 2.0648e-1, 4.8111e-2, 1.8340e-2, 6.9367e-3
powersrt(1,3)= 2.2948e-1, 4.9065e-1, 2.0648e-1, 4.8111e-2, 1.8340e-2, 6.9367e-3

anzinfrt(1,1)=    13.488,     8.093,     2.298,    -2.360,    -4.721,    -7.081
anzinfrt(1,2)=   -2.1937,   -4.3875,   -6.5813,    2.4376,    7.3127,   11.7001
anzinfrt(1,3)=   -1.5563,   -3.1126,   -4.6689,    1.7292,    5.1876,    8.3000

anzsuprt(1,1)=    13.489,     8.094,     2.299,    -2.359,    -4.720,    -7.080
anzsuprt(1,2)=   -2.1938,   -4.3876,   -6.5814,    2.4375,    7.3126,   11.7000
anzsuprt(1,3)=   -1.5562,   -3.1125,   -4.6688,    1.7293,    5.1877,    8.3001

nnkparrt(1,1)=   6*1
nnkparrt(1,2)=   6*1
nnkparrt(1,3)=   6*1

!NUMBER OF N POLOIDALS IN EACH BIN:
nnkpolrt(1,1)=        1,        1,        1,        1,        1,        1
nnkpolrt(1,2)=        1,        1,        1,        1,        1,        1
nnkpolrt(1,3)=        1,        1,        1,        1,        1,        1

!UPPER LIMIT POLOIDAL REFRACTIVE INDEX:
anpsuprt(1,1)= 6*-1.730
anpsuprt(1,2)= 6*-1.250
anpsuprt(1,3)= 6*-0.887

!LOWER LIMIT POLOIDAL REFRACTIVE INDEX:
anpinfrt(1,1)= 6*-1.731
anpinfrt(1,2)= 6*-1.251
anpinfrt(1,3)= 6*-0.887


 nthinrt(1) =    11,    11,    11               ! NUMBER OF STARTING LOCATIONS
 maxrefrt(1) =    6,    10,    10               ! MAX NUMBER EDGE REFLECTIOSN
 islofart(1) =   -1,    -1,    -1               ! -1 FAST WAVE, 1 SLOW WAVE
 heightrt(1) = 120.,  120.,  120.               ! ANTENNA HEIGHT , CM


 indvar=1     ! 1 USE TOTAL PHASE AS INDEP VARIABLE
 ichois=2     ! 1 HIGH FREQ DISP. , 2 NO FREQ LIMIT
 modcd=0      ! 0 EHSR KARNEY MODEL, 1 CHIU-KARNEY-MAU MODEL FOR J,P CALCS
 igrill=-3    ! -3 READ IN SPECTRUM, -1 ANALYTIC SPECTRUM
 bmaxrt=30    ! max Bessel function order
 idmpsw=1     ! 0 NO ION DAMPING, ‘ MAGNETIZED ION DAMPING,
              ! 2 UNMAGNETIZED ION DAMPING
 irayiort=0   ! details of data output to file rayiop
 beam_spec =-1,1,1,0 ! BEAM_SPEC(I) ,I =1 FULL ENERGY , 2 =HALF ,3 = THIRD 4 = FAST ALPHA
                    ! =1 MEANS TREAT THIS COMPONENT OF THE FAST IONS AS A SEPARATE SPECIES
                    ! =0 MEANS OMIT THIS COMPONENT. USED FOR EACH INJECTOR SEPARATELY
                    ! THUS TO GET CUR DRIVE DUE TO FAST ALPHAS ONLY
                    ! SET BEAM_SPEC = 0,0,0,1 FOR EXAMPLE.
                    ! TO USE ONLY THE FIRST ENERGY COMPONENT (OF ALL INJECTORS) SET
                    ! BEAM_SPEC = 1,0,0,0 , ETC.
                    ! DENSITIES AND 2/3 FAST ION STORED ENERGY
                    ! DENSITY DIVIDED BY FAST ION DENSITY IS OUTPUT TO TRXPL.OUT
                    ! NUMBER OF FAST SPECIES IS (NO. INJECTORS ) * (SUM FROM 1 TO 3 OF { (BEAMSPEC(I))})
                    ! PLUS BEAM_SPEC(4)
!--------------------end curray specific input --------------------------------------
```

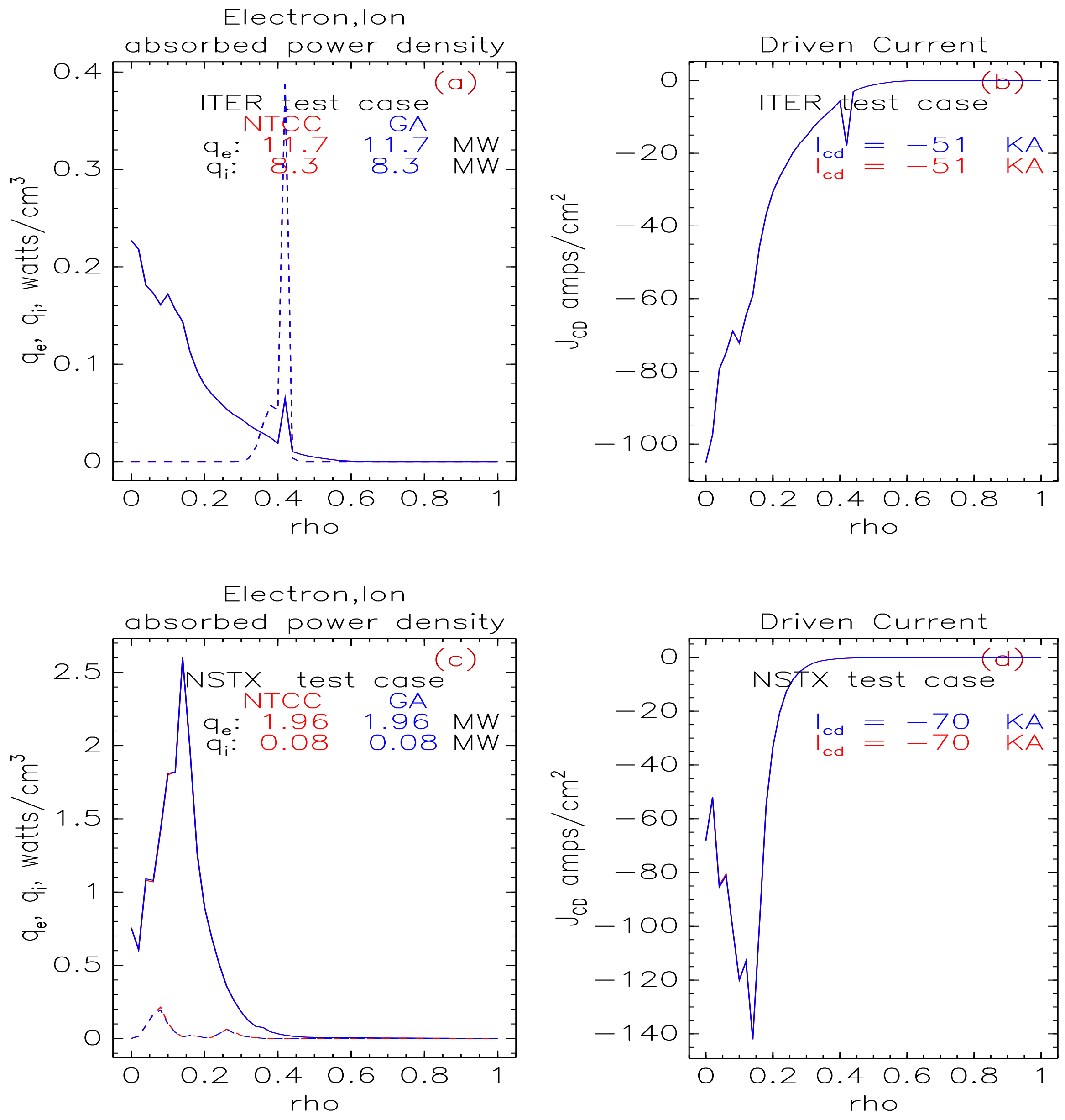


Figure 13: (a) Electron and ion (dashed line) absorbed power density for 20 MW of input power for the example ITER discharge. The corresponding driven current profiles are given in (b). Figures (c) and (d) give the same results for the NSTX case with 2.1 MW of injected power. The NTCC and GA versions of Curray produce identical profiles for these cases.

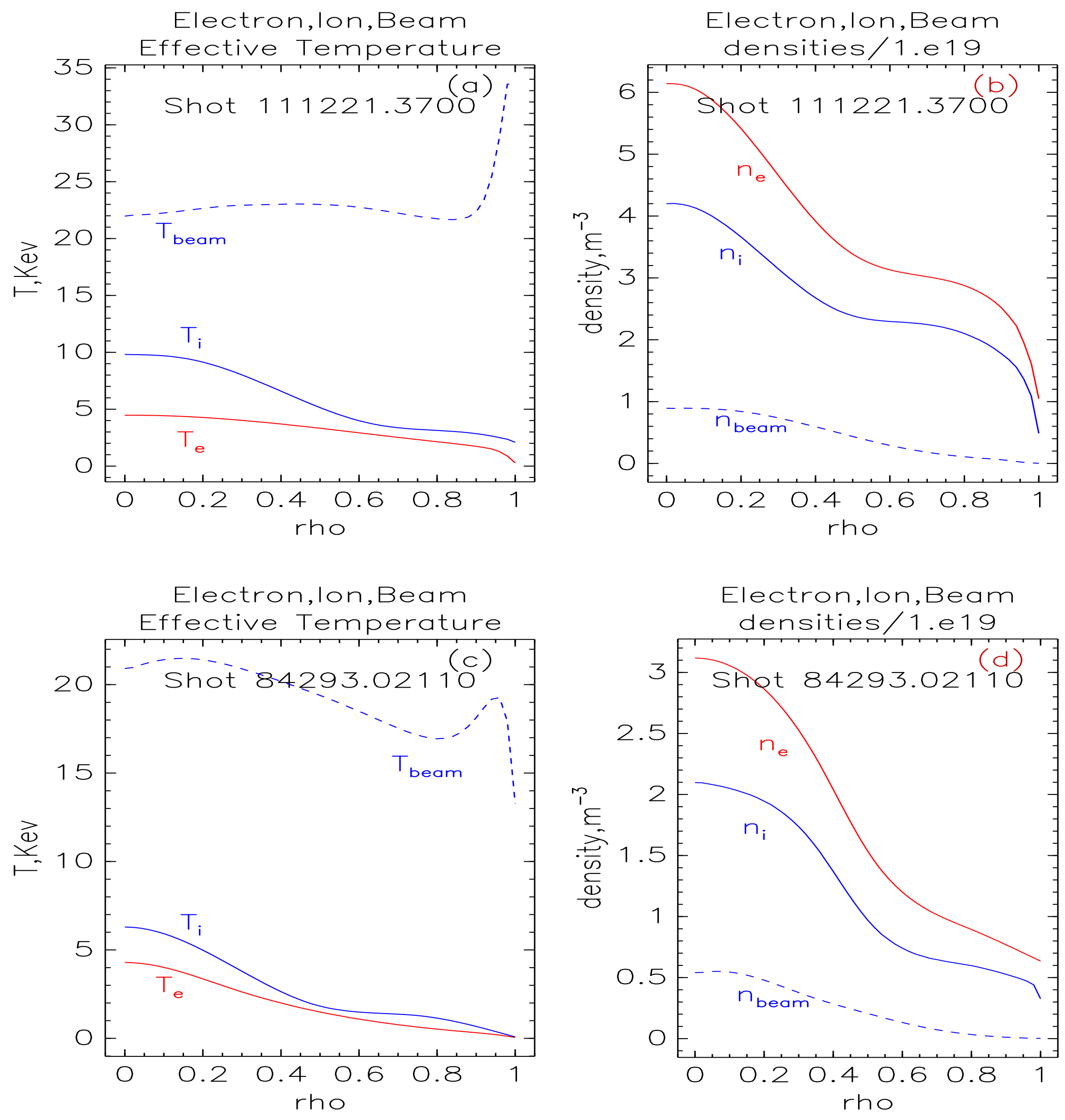


Figure 14: (a,b): Temperatures and densities for the H mode shot 111221 at 3700 msec. The beam temperature is taken as $\frac{2}{3}$ of the average fast ion energy. $\boldsymbol{\beta = 3.23\%, B_T = 1.86T, I_p = 1.18MA, \overline{n} = 4.03\ 10^{13}cm^{-3}}$. (c,d): The corresponding profiles for L mode shot 84293 at 2110 msec.$\boldsymbol{\beta = 0.7\%, B_T = 2.08T, I_p = 1.38MA, \overline{n} = 1.95x10^{13}cm^{-3}}$

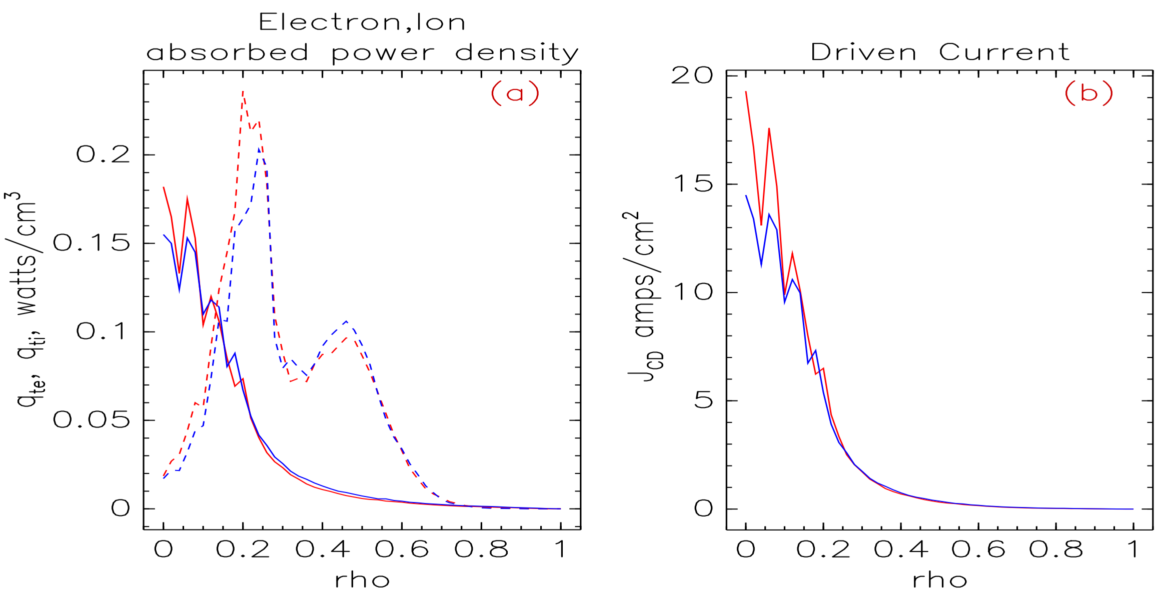


Figure 15: (a) Electron and ion absorbed power density for 1 MW of input power at 60MHz for DIII-D discharge 111221 during H mode phase. The blue curves are the Curray result when run out of Onetwo. The red curves are the results obtained from Transp coupled to the NTCC version of Curray. The dashed lines represent the power density given to the ions. (b) The corresponding driven current profiles. shot 111221 @ 3710 msec

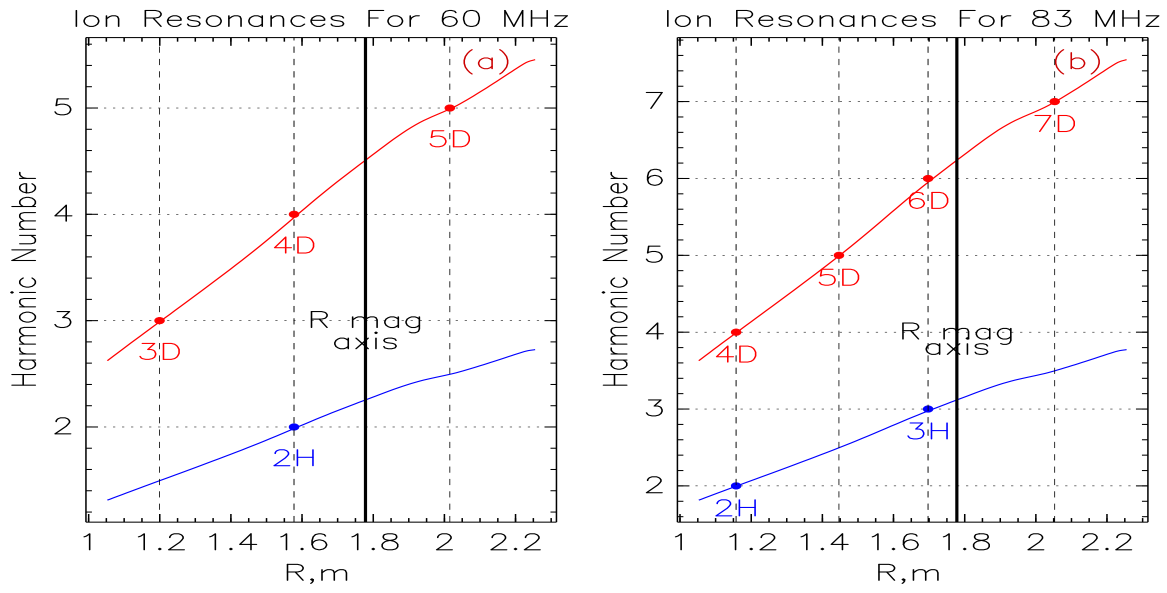


Figure 16: (a) Major radius location of hydrogen and deuterium resonances for H mode shot 111221.03700 at 60MHz. (b) The same result for 83MHz. The magnetic axis is at 1.778 m

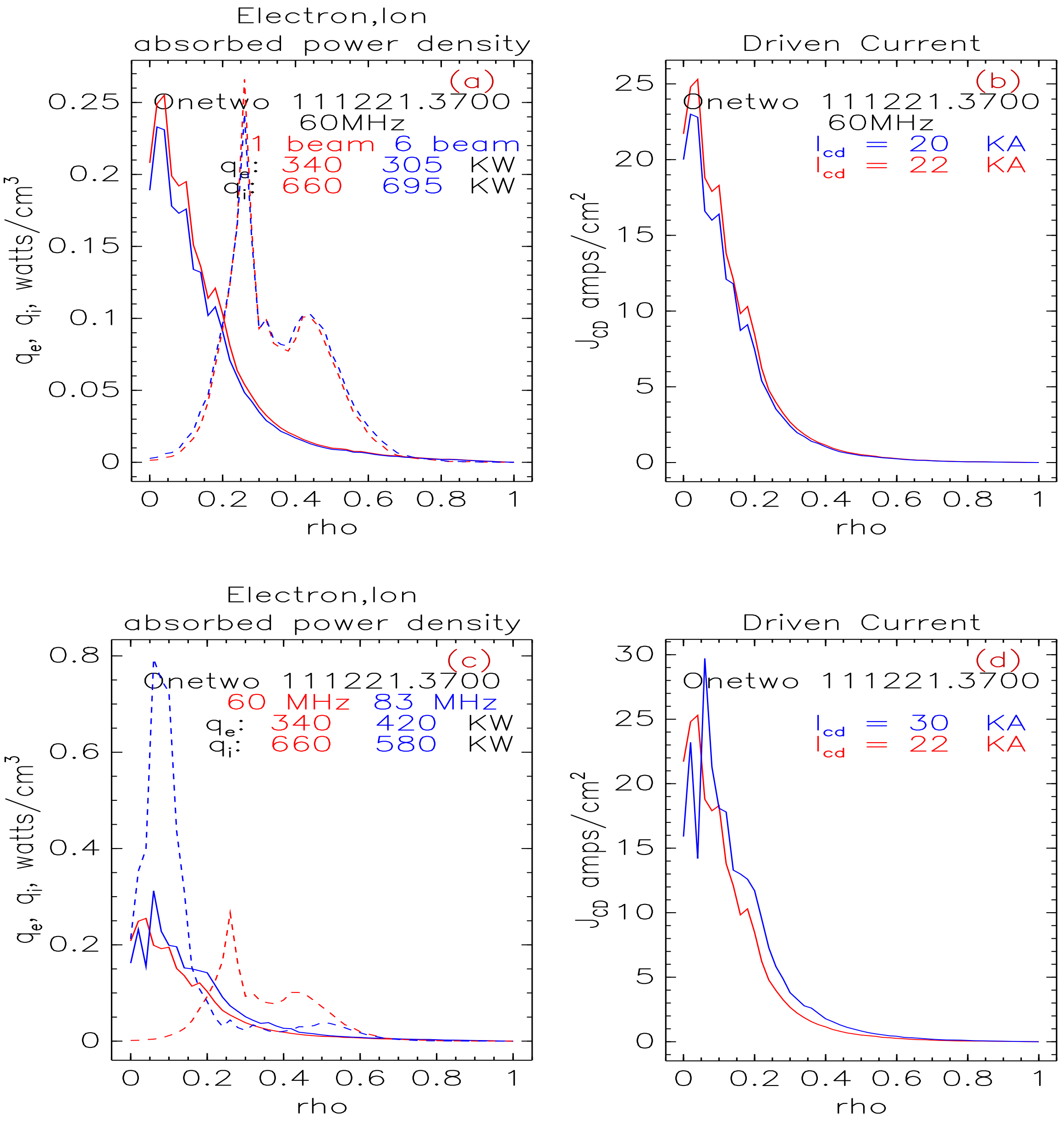


Figure 17: (a,b) The heating and current drive profiles from Curray obtained using Onetwo. Both effective single beam and multiple beam results are shown. (c,d) Comparison of the 60 and 83 MHz fast wave heating and current drive profiles. The electron heating has increased while the resonance absorption of the (fast) ions has moved closer to the magnetic axis and become somewhat less efficient.

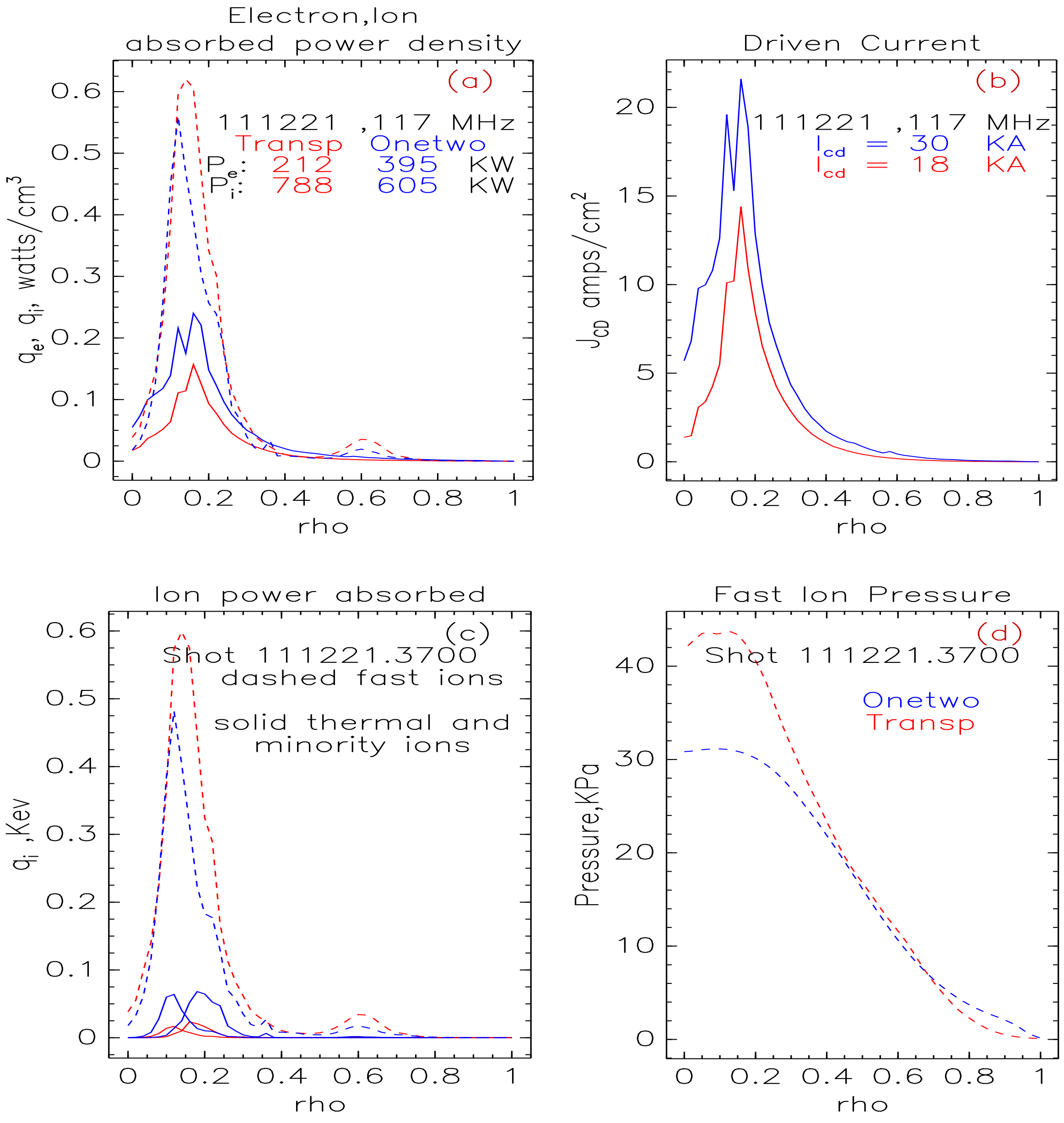


Figure 18: Results for the H mode discharge at 117 MHz compared with Transp. As explained in the text the larger ion power absorption in Transp drives the observed differences

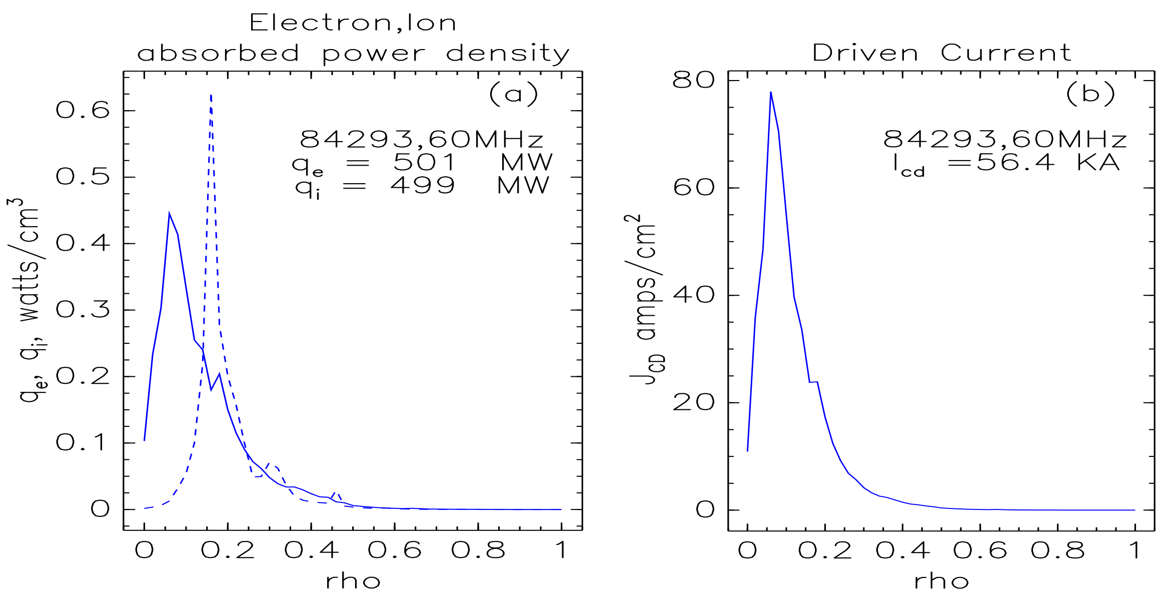


Figure 19: (a,b) The heating and current drive results for the L mode discharge 84293.2110 at 60 MHz. In (a) the solid line represents the electron heating and the dashed curve is the total ion heating. In (b) the electron driven current is shown.

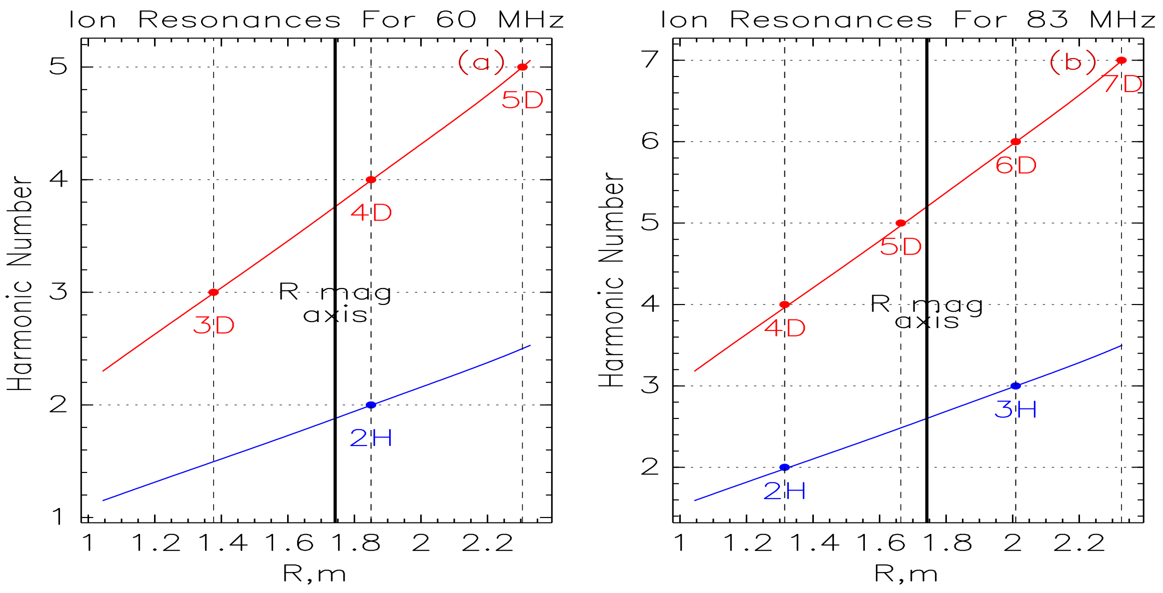
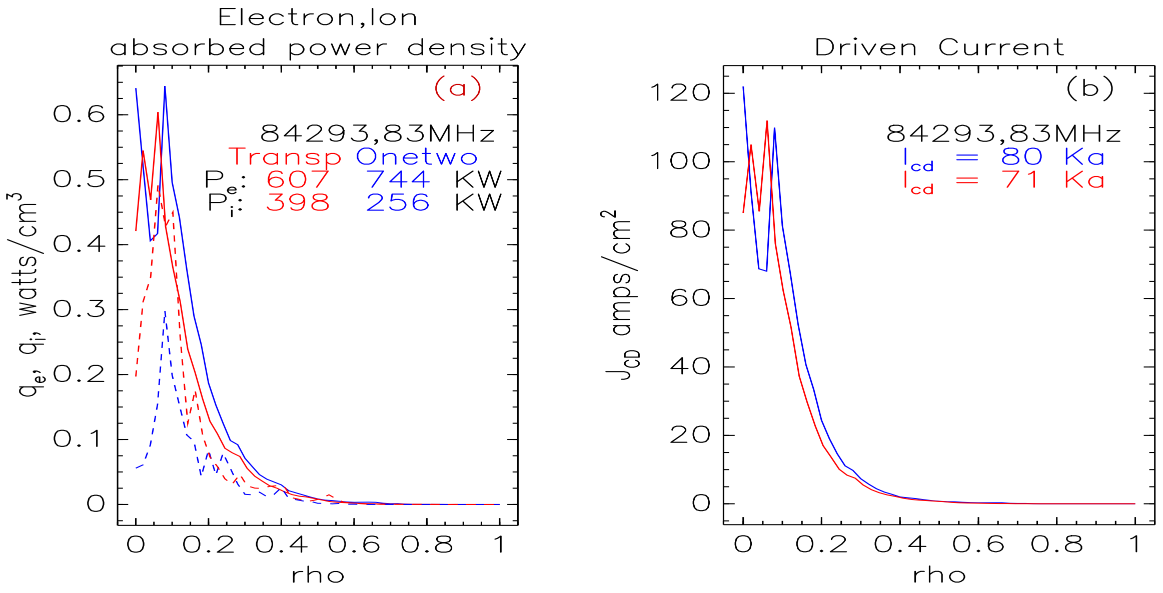


Figure 20: (a,b) Major radius location of hydrogen and deuterium resonances for L mode shot 84293.02110 at 60 and 83 MHz



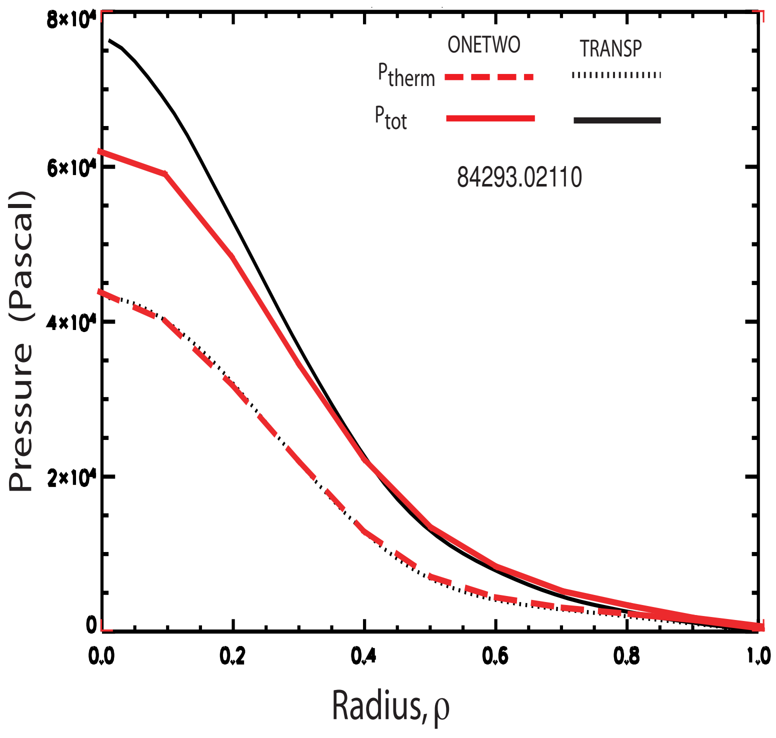


Figure 21: L mode discharge results for 83 MHz, fast wave coupled input power. Both the Transp and Onetwo results are given for the heating and current drive profiles. Transp has significantly more power absorption by the fast ions. The total and thermal pressures determined by Transp and Onetwo are given in the lower figure.

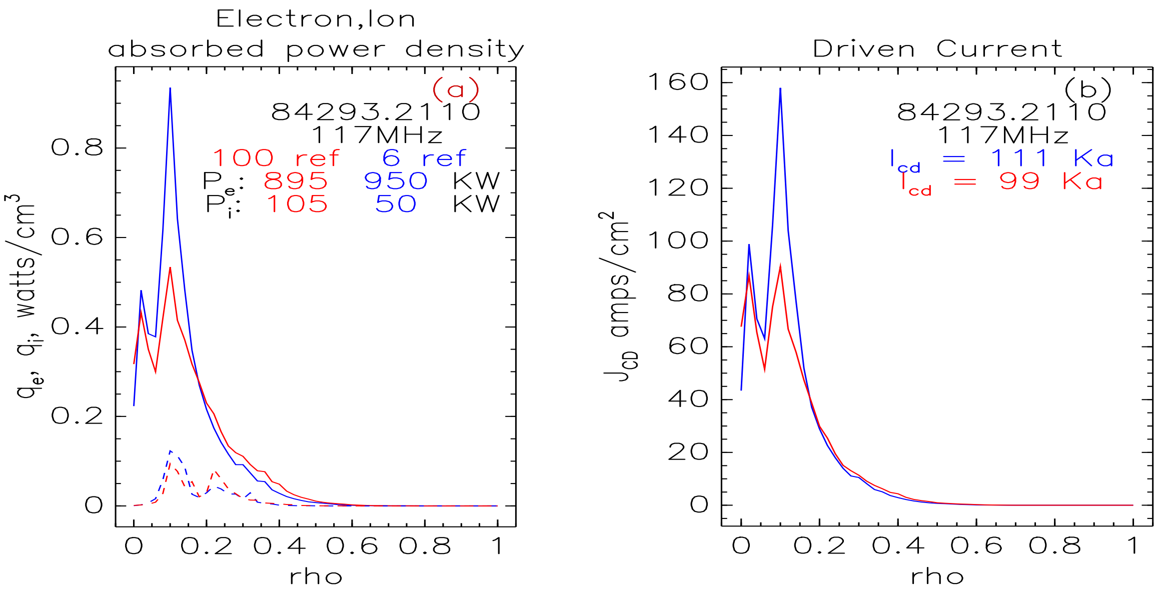


Figure 22: Results for the L mode shot with Onetwo used as the driver for Curray. The red curves correspond to allowing 100 reflections of the rays.The blue curves are for 6 reflections

K Running Nubeam in Onetwo

N.B. : This appendix was part of a submission to the NTCC. Consequently some observations/remarks/details are not intended for the users of the module,rather they were directed at developers.

Funding provided by the National Transport Code Collaboratory (NTCC) enabled us to install and test the Monte Carlo fast ion physics package, Nubeam [12], at General Atomics. The project consisted of building an interface to the Nubeam code in Onetwo and subsequently testing the Nubeam code with DIII-D and Iter discharges. We found that the Nubeam code gave results for all physical parameters tested that were supported by more primitive analytic fast ion slowing down calculations. Please note that it was not our job to actually perform a formal review of the Nubeam code. Consequently a reviewers sheet is not attached to this report.

The most recent version of Nubeam uses C++ features that are not currently supported by our local computational facilities and consequently we are forced to remain with the older version of the modules dated Sept. 2003. We note that locally encountered difficulties in building the Nubeam code have generally been caused by the coupling of the C++ code with Fortran and we consequently urge the developers of the code to rewrite the Preact module in Fortran. Even the older (Sept. 2003) version of the code will not build on our local HP architecture which further restricts our use of this module. At this time we satisfactorily run Onetwo coupled with Nubeam only on Linux based X86 machines.

Our original attempt at interfacing the Nubeam module with Onetwo by linking the Nubeam libraries directly with Onetwo indicated that, at least in our implementation, a memory leak of unknown origin was responsible for our inability to get successful runs. This difficulty was ultimately resolved by changing the interface to externally call Nubeam at desired times and shutting the Nubeam program down in between these calls. This latter method also gave us the ability, through judicious use of save files, to run Nubeam in a stand alone fashion on any subset of Onetwo generated input files.

K.1 Onetwo Interface

The interface to Onetwo is described in some detail in this section to act as a guide for other institutions contemplating a similar project as well as to provide written documentation of the use of the module in Onetwo. As remarked above we now run the Onetwo Nubeam combination separately,which requires that Onetwo generates suitable input files for Nubeam any time a call to the fast ion physics package is made. Since equilibrium evolution is also part of the general transport scheme appropriate time dependent equilibrium information must also be available. Finally, since the Nubeam package solves the instantaneous deposition as well as fast ion slowing down problem (including fusion originated alpha particles) some form of time step control for Nubeam calls must be provided. The time stepping itself must be aware of possibly arbitrarily complicated beam pulse waveforms (see Fig(27a) for example).

The Nubeam driver called by Onetwo is a modified version of the recent nbdrive program made available by PPPL.(I have not yet produced a patch file that will generate these changes automatically).

Interfacing to Nubeam is complicated by the fact that as many as three separate files must be read in order to collect the information required to setup a suitable Onetwo/Nubeam coupled run. The first file is the standard Onetwo input file called inone. For Onetwo runs that do not use Nubeam the inone file contains all of the necessary beam information. That case is not considered further in this paper. If a Nubeam type Onetwo run is selected in inone by an appropriate choice of input parameters then additional information taken from one or two other input files is required. The arbitrary names of these additional input files,beam_data_namelist and beam_data_ufile are specified in inone. The file pointed to by beam_data_ufile is optional(currently not used) and is intended to provide an automatic,MDS Plus driven interface to the beam pulse data. This file is part of the interface work still required to couple Onetwo to Nubeam when time dependent waveforms are to be used . The last file,pointed to by beam_data_namelist, is the primary Nubeam input file and is required whenever Nubeam is run. This is the file traditionally written by nblist.for (a Transp routine that creates the nbdrive_naml namelist). It has in it the beam geometry and the order of the beam data in the ufile, as well as the the beam on and off times, etc. Onetwo will attempt to read this file as a standard namelist input file (which means generally that the file must first be edited by hand because nblist does not properly terminate namelists ??)

The actual driver code called by Onetwo when Nubeam is started is called Nubeam_driver. (This is also the name of the code to execute if Nubeam is run in stand alone mode). When Nubeam_driver is started it expects to read a single file that contains all necessary information to run Nubeam. Nubeam_driver is set up to take the first part of this name from the command line used to start it. Thus for example the Nubeam package is started by executing the line

```
nubeam_driver nubeam_12
```

Where the complete input file name is actually nubeam_12_namelist.dat. Onetwo dynamically constructs a command line which includes the name of the input file that Nubeam_driver will read. nubeam_12_namelist.dat is created by the Onetwo code before Nubeam_driver is called. Once this file exists Nubeam can be run in stand alone mode if desired. But the normal usage is to have Onetwo repeatedly run Nubeam through Nubeam_driver as a sub process with automatically generated nubeam_12_namelist.dat files that reflect the plasma state at the current time.

Nubeam_driver creates output files based on the first part the input file name given on the command line. Thus for example the files, nubeam_12_details.dat and nubeam_12_summary.dat are created. The file nubeam_12_summary.dat is intended for visual perusal and is not used further by Onetwo. The other file, nubeam_12_summary.dat, contains all of the fast ion deposition and slowing down data that Onetwo will process. This file is parsed by Onetwo but I do not recommend that approach (instead the rather lengthy output should be reformatted explicitly for machine readability and an appropriate read routine should be created - I have not done this at this time). Nubeam_driver will also write two netcdf formated restart files, nubeam_12_xplasma_state.cdf and nubeam_12_nubeam_state.cdf. These files are read by Nubeam on the next call to Nubeam and serve as initial conditions for the next time step. Onetwo does not read these netcdf files at all. Obviously on the first call to Nubeam these restart files will not exist. This forces us into the situation that we must start the analysis before the beams (or other fast ions) exist. However once restart files exist they will be used to restart the analysis at suitable times so this restriction is not very severe. A complication is the fact that Onetwo profiles are also changing as a function of time and hence another file, nubeam_12_restart_profs.txt, is required to complete a restart problem specification. All of these files are overwritten each time Nubeam is executed! The user must make sure that old *.cdf files are removed form the working directory before starting a new case. Otherwise Nubeam will pick up those old files, most likely resulting in unwanted results. On option the restart files for Nubeam can be saved at a particular time using the inone parameter wrt_restart_file_time.

Since Nubeam does both the instantaneous deposition of neutral beam ions as well as the slowing down of fusion products and beam ions a finite time step, $\boldsymbol{\Delta t_{nb}}$ (greater than about $\boldsymbol{10^{-5} sec}$) must be supplied to the Nubeam code. The fast ion distribution related quantities are then evolved from $\boldsymbol{t}$ to $\boldsymbol{t + \Delta t_{nb}}$. At the final time, $\boldsymbol{t + \Delta t_{nb}}$ the physical quantities are dumped to file nubeam_12_details.dat. But Nubeam was called with profile that exist in the transport code at time $\boldsymbol{t}$. Since Nubeam does not evolve (thermal) transport related quantities such as the thermal electron and ion densities and temperatures these and other profiles are held constant during the Nubeam time step $\boldsymbol{\Delta t_{nb}}$. Thus there is an implied iteration necessary to fully bring the fast and thermal distributions into agreement.

Given the current processing power available to us such iteration is however not feasible and hence no attempt to accommodate it currently exists in Onetwo. The minimum time step allowed by Nubeam ($\approx \boldsymbol{10^{-5} sec}$) is much smaller than what is practically possible. Although stiff confinement models may force Onetwo to take sub millisecond time steps in the thermal transport modeling, typically we use 5 to 20 percent of the fast ion lifetime for $\boldsymbol{\Delta t_{nb}}$. To accommodate this disparity in time steps linear interpolation is used. That is, once Nubeam returns with the fast ion related quantities at time $\boldsymbol{t + \Delta t_{nb}}$ we return to the thermal transport model in Onetwo at time $\boldsymbol{t}$ and evolve the thermal quantities from time $\boldsymbol{t}$ to time $\boldsymbol{t + \Delta t_{nb}}$ using linear interpolation of the nubeam quantities which are now known both at the beginning and at the end of the time interval $\boldsymbol{\Delta t_{nb}}$. I believe that such a non iterative approach is benign in the this application but not attempt to justify it has been made.

K.2 Using Nubeam with Onetwo in restart mode

As explained above the Onetwo/Nubeam combination must be initially set up to run from a start time where none of the beams are on. Once such a run exists however it is possible to use the files generated by Nubeam and Onetwo as restart files. The three required restart files provide fast ion information such as beam density, slowing rates,etc., at the time the restart files were written. The restart files supplement the information in the file nubeam_data_namelist and all four files are required in a restart run. The restart information can be used as initial conditions for a new run, eliminating the requirement that the beam power is zero at the start time of the Onetwo run. It is the users responsibility to make sure that the information in the restart files is compatible with the inone file that will be used to run Onetwo. The restart file,nubeam_12_restart_profs.txt (or similar name -see above), has the time of creation of the file in it on the first line,nubeam_restart_time. The second line contains the number of beams that are on at this time,nbeams, the following 4*nbeams lines contain the beam power, energy, full and half energy fractions for each beamline. Recall that time0 is the given in inone at which the transport simulation is run. Suppose that we set things up so that time0 ¡ nubeam_restart_time. Then as Onetwo steps forward in time we eventually encounter the nubeam_restart_time and there would be a sudden jump in such quantities as the fast ion density when the information in the restart file was added to the current information in Onetwo. This is non physical and hence we do not allow nubeam_restart_time ¿ time0 as input (the code will check for this and exit if it is found). Hence we must have time0 $\leq$ numbeam_restart_time. The code applies the initial conditions present in the nubeam_12_restart_profs.txt at time0, even if the nubeam_restart_time in that file is less than time0 ! The times at which the individual beams are turned on and off, (tbona and tboffa in file beam_data_namelist) must be consistent with this (as long as the user does not modify nubeam_12_namelist.dat from what it was when the restart files were created this will be the case). The problem that can arise here is that an individual beam, say beam number 2, is off at time nubeam_restart_time but the user sets time0 in inone such that beam 2 is on at time0. This is an inconsistency since the restart file does not contain any information about beam 2 but the value of time0 in inone assumes that beam 2 is on at the beginning. If it is found that at time0 beam 2 is on but beam 2 is off at time nubeam_restart time in file nubeam_12_restart_profs.txt the code will exit.

An obvious application of the restart method is to supply a consistent electric field at time0 when beam driven current is present. By setting up an initial run which assumes that the beam is turned on 3 slowing down times before time0 and running from that time up to time 0 to get an equilibrated beam at time0.

K.3 File Usage Summary

As is evident from the discussion above the file structure associated with running Nubeam is somewhat complex. A new set of these files is created each time Nubeam is called. Normally the previous versions of these file is simply overwritten by the new ones at the current time. The following summary should help in keeping things in perspective:

inone The primary Onetwo input file. The Nubeam related input in this file is all given in the second namelist. It is highly recommended (to avoid confusion) that only the following beam related quantities are specified in inone when the Nubeam option is selected:

- **use_nubeam** logical variable if true indicates that Nubeam is to be used. Default is false.
- **beam_data_ufile** Name of the ufile to be used to get some beam input quantities. Not well defined at this time (and hence not used)
- **nubeam_restart** Integer, = 1 use existing restart files on first time step,
  =0 restart files don't exist, code will create them.
  if nubeam_restart = 1 supply the following:
  - **nubeam_state_path** Fully qualified name of Nubeam state restart file
  - **nubeam_xplasma_path** Fully qualified name of xplasma state restart file

profile restart file This item is NOT input in inone. It is here as a reminder that the file ***_restart_profs.txt is also required. Here *** is the value of nubeam_state_path. In other words the profile restart file is assumed to reside in the same place as the state restart file.

wrt_restart_file_time A single time at which the restart files are to be saved. If left blank then only the last set of restart files at the final time will be available.

save_nubeam_input Integer, defaulted to 0. set to 1 to save all input files to nubeam (one such file is created each time nubeam is called) Note that Nubeam must be called on a regular basis even after the beam is off since fast ion slowing down is part of the Nubeam calculations. Hence use this option cautiously.

beam_data_namelist Character variable, default is 'nubeam2_namelist.dat' It is the name of the beam input data file used by Nubeam. if use_nubeam = .false. and beam_data_namelist points to a valid file, that file will be used to generate beam input for the standard Onetwo Nfreya package.

A detailed input description to run Nubeam as a sub process of Onetwo is given in file cray102.f

K.4 Nubeam - Onetwo Comparison

In Onetwo the initial fast ion deposition is also done using Monte Carlo methods. This aspect of the problem is thus expected to be identical in the two codes and it is only the subsequent orbiting (or lack thereof) during the fast ions lifetime that drives differences between the codes.

In Fig(23) we show a simple time dependent example of the type normally treated by Onetwo. A single beam pulse is turned on at 1800 msec and remains on for the duration of the 200 msec transport time. The total number of fast ions in the plasma approaches a steady state value as indicated in part (b) of the figure. In this and all subsequent figures we show the analytic and Nfreya based results generated by Onetwo in red and the Nubeam results for the identical case in blue. For Nubeam two curves are show, the choppier,dashed curve, corresponds to using 1000 "ions" in Nubeam. The smoother,solid, blue curves refer to Nubeam results using 10000 Monte Carlo particles. As indicated in Fig(23(b)) the assumed analytic slowing down distribution in Onetwo leads to larger number of fast ions, due in part to fewer charge exchange losses. The actual equilibrium fast ion density is shown Fig(24a) where we see that the higher,co-injected, fast ion density in Onetwo is on the outer half of the plasma. Onetwo uses a prompt orbit model,based on conservation of canonical angular momentum, to spread the fast ions out over an orbit width at the time of birth. Obviously such a prompt model can only approximate the spatial diffusion of the fast ions during the slowing down process.

In Fig(24b) we show the various combinations of fast and thermal $\boldsymbol{d(d,n)he^4}$ neutron rates as a function of time. The thermal neutron rate determined by Nubeam is not distinguishable from the rate determined by Onetwo in the figure. The slight drop in the thermal rate as a function of time is due to the buildup of fast ions. Since zeff and the electron density are held constant during the simulation the thermal ion density decreases slightly as the steady state fast ion density approaches equilibrium. However the beam thermal rate is significantly less in Onetwo. The beam-thermal neutron rate in Onetwo is calculated by using a Maxwellian distribution for the thermal ions and a classical slowing down distribution for the fast ions. Consistent with this the beam-beam rate in Onetwo has the value of $\boldsymbol{2.71x10^{13}/sec}$ compared to $\boldsymbol{3.11x10^{13}}$ for Nubeam. Even though the beam density is higher in Onetwo the beam associated neutron rates are less. Assuming that the Monte Carlo determined rates in Nubeam are more applicable this indicates that the assumed slowing down distribution in Onetwo can lead to beam related neutron production rates that are in error by $\boldsymbol{\approx 30\%}$.

The preferential heating of the thermal ions after 200 msec of beam evolution is shown in Fig(25). Here the the Nubeam determined spatial distribution of the neutral beam heating profile is somewhat noisy, even when 10000 Monte Carlo particles are used in the simulation. Near the magnetic axis a simulation with 1000 particles can be quite far off due to the lack of sufficient sampling in the small volume near the magnetic axis. Such a result would tend to cause havoc in a transport simulation (at the next time step the void area might be filled in ,etc.) and hence we expect that significantly more than 1000 ions should be used as a matter of routine. Overall the tracking with the Onetwo results for both the spatial power density (Fig(25a)) and integrated power (Fig(25b)) is reasonable.

Two other features which are of paramount importance to the analysis and simulation of thermal plasmas are the beam driven current ad the beam supplied torque density that drives the toroidal rotation of the thermal ion species. These quantities are compared with Onetwo in Fig(26a,b) respectively. For Onetwo the total (shielded) beam driven current is 75 KA, while for Nubeam the number is about 90 KA. The current profiles track reasonably well and the difference can be ascribed to the analytic formulation of the beam driven current used in in Onetwo ([11]) versus the Monte Carlo simulation of the collision operators used in Nubeam ([12]).

The torque density, Fig(26b), from Nubeam is quite noisy even with 10000 ions (the 1000 ion result is not shown because it is too noisy). Because the torque density enters only as a source term in the toroidal momentum equation the effect of this noise is probably not severe. However we should keep in mind that the actual torque density used at any given time and at any given radial grid point, will be a linear combination of two profiles of the type shown, separated in time by $\boldsymbol{\delta_{nb}}$ as explained above. Since this time interval can be large we might expect some discrepancies due to this effect, particularly with stiff confinement models that depend on the rotational shear.

We do not explicitly discuss the final set of figures,Fig(27,28). They are included here to demonstrate that arbitrarily complex beam pulse waveforms can be handled by Nubeam.

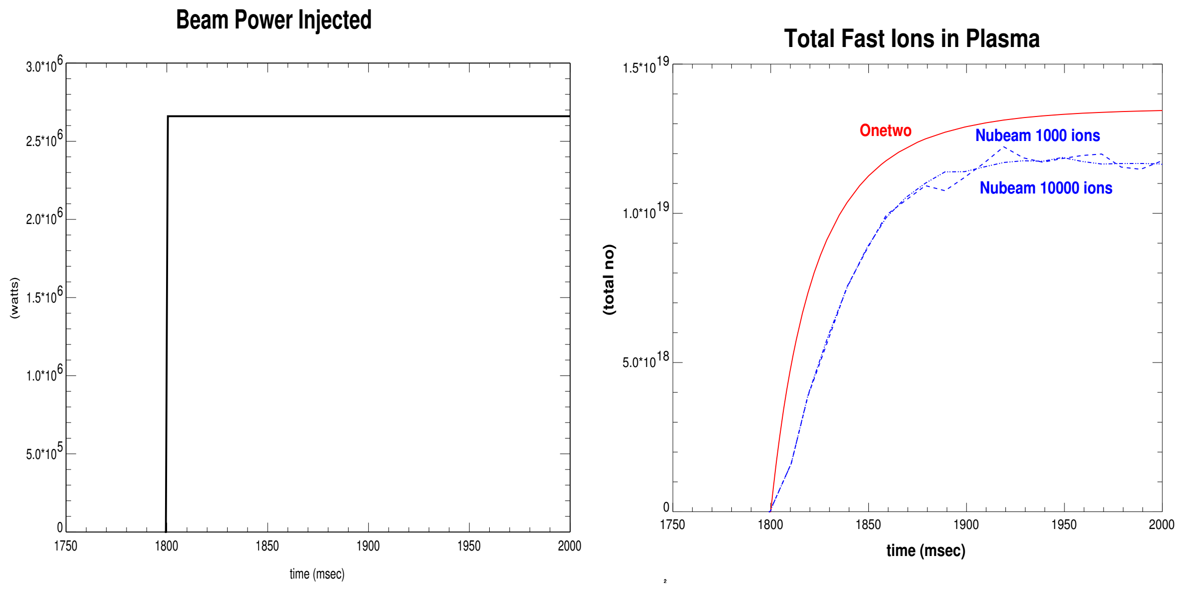


Figure 23: (a) A simple rectangular beam pulse used to drive Nubeam and the Onetwo fast ion calculations. (b) The resulting stored fast ion beam density showing the asymptotic approach to steady state

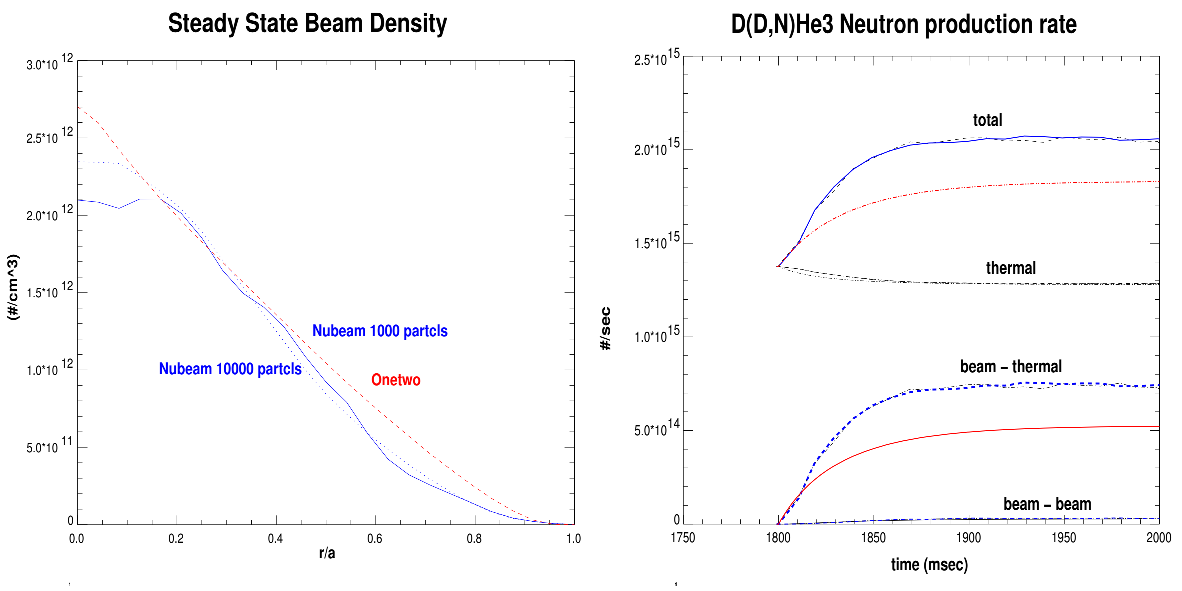


Figure 24: (a)The fast ion density at 2000 msec across the plasma. (b) the neutron rates as a function of time

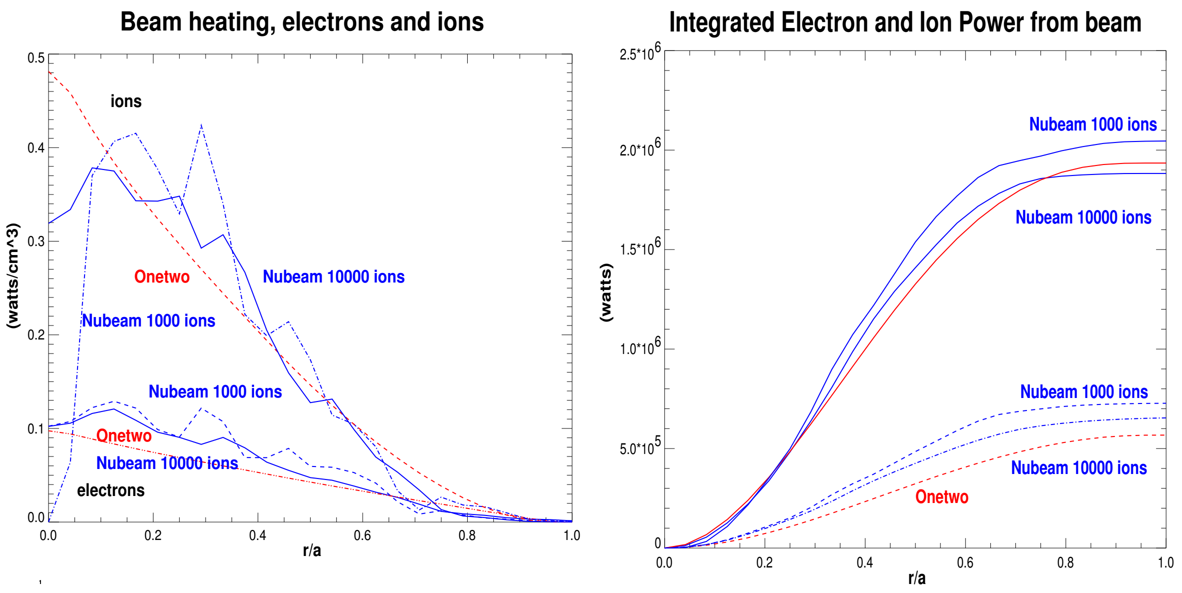
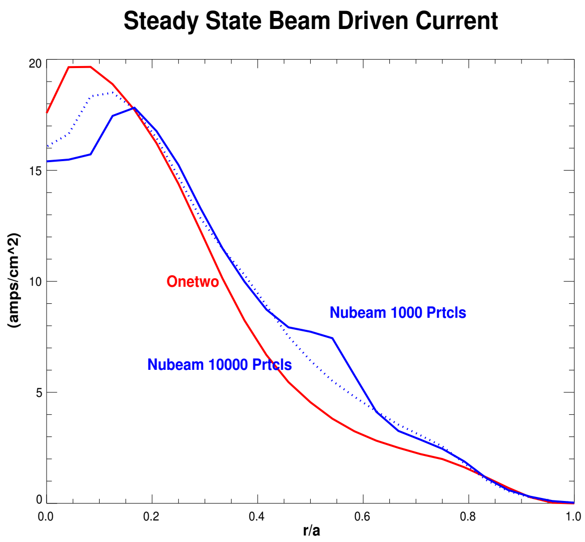


Figure 25: (a)The electron and ion thermal heating profiles. (b) The integrated thermal heating profiles



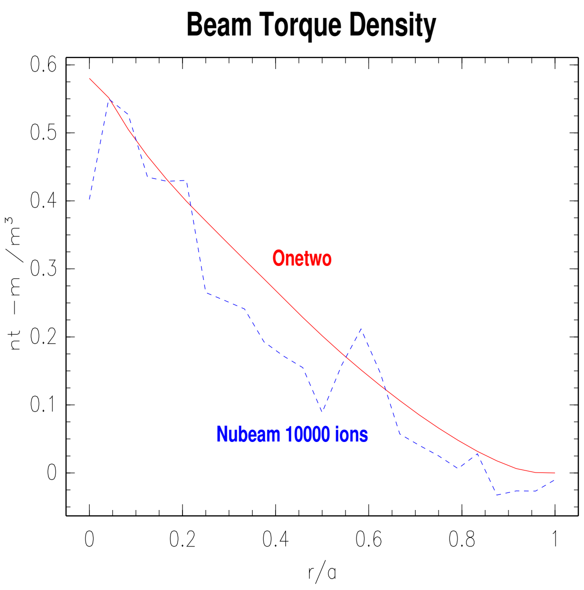
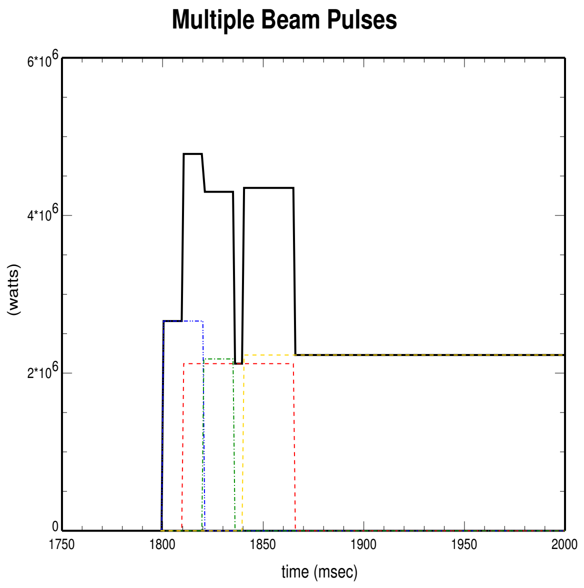


Figure 26: (a)The beam driven current. (b) The torque density acting to spin up the plasma



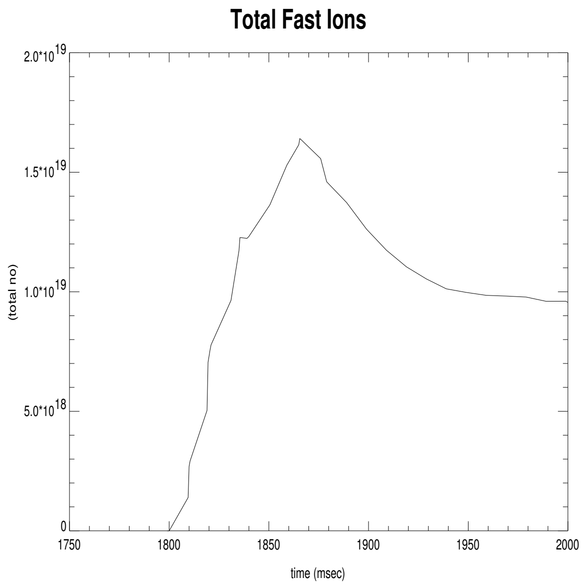
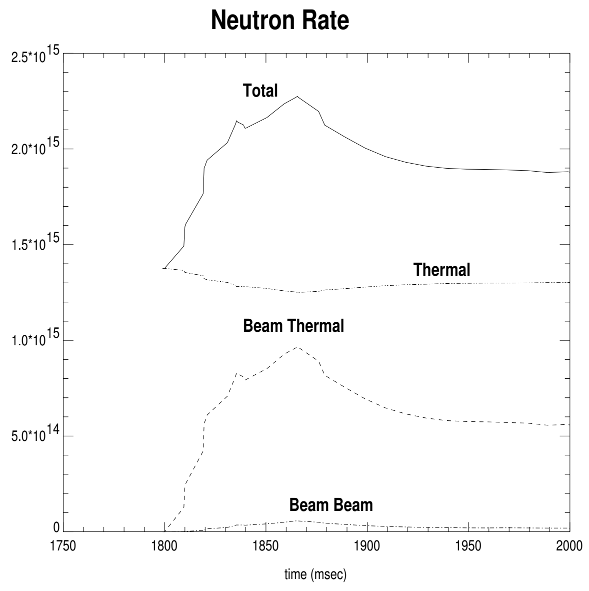


Figure 27: (a)A complicated series of beam pulses (b) The resulting total number of fast ion in the plasma



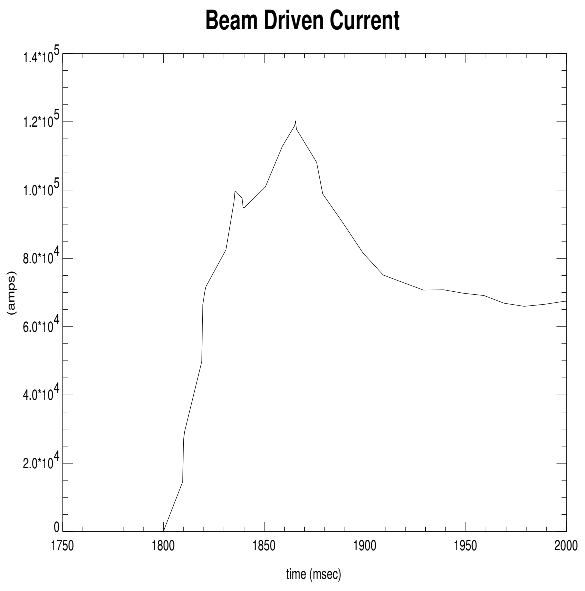


Figure 28: (a)The Nubeam calculated neutron rates and (b) The Nubeam determined beam driven current due to the pulsed power shown in Fig(27a).

References

[1] K.M. Burrel. *J. Comp. Phys.*, 27:88, 1978.

[2] J.D.Callen, R.J.Colchin, R.H.Fowler, D.G.McAlees, and J.A.Rome. Neutral beam injection into tokamaks. In *Proc. Fifth Intl. Conf. on Plasma Phys. and Controlled Nuc. Fusion Research*, volume I, page 645. IAEA, 1974.

[3] R.Freeman and E.Jones. Analytic exp[ressions fpr selected cross sections and maxwellian rate coefficients. Technical Report CLM-R 137, UKAEA,Culham, 1974.

References

[1] Wouwer,a.,editor,"Adaptive Method of Lines", Chapman Hall,Boca Raton,Fl.,2001

[2] Nocedal,J.,Wright,S.,"Numerical Optimization", Springer,N.Y.,Ny.,1999

[3] Dennis,J., and Schnabel,R. "Numerical Methods for Unconstrained optimization and Nonlinear Equations", Prentice-Hall,Engl.Cliffs,N.J.,1983

[4] V. Ganzha and E.V. Vorozhtsov " Computer-Aided Analysis of Difference Schemes For Partial Differential Equations",Wiley,N.Y.,N.Y.1996

[5] Strikwerda,J.,"Finite Difference Schemes and Partial Differential Equations", Wadsworth&Brooks/Cole Pacific Grove,Cal.,1989

[6] von Rosberg,D.U. "Methods For The Numericall Solution of Partial Differential Equations", Am. Elsevier ,Ny.,N.Y.,1975

[7] Jaeger,E.F., et al., Phys. Plasmas 8, 1573 (2001)

[8] http: //aries.ucsd.edu/mau/CURRAY/web/

[9] Petty,C.C.,et. al.,in 'Radio Frequency Power In plasmas', AIP Conference Proceedings 244, (Charleston,1991),p.96

[10] http://w3.pppl.gov/NTCC/standards/standards.html

[11] J.D. Gaffey,Jr.,"Energetic ion distribution resulting from neutral beam injection in tokamaks",J.Plasma Phys. (1976), Vol 16,part 2, 149-169

http:/fusion.gat.com/comp/analysis/grid/onetwo/

[12] A. Pankin, D. McCune, R. Andre et al., "The Tokamak Monte Carlo Fast Ion Module NUBEAM in the National Transport Code Collaboration Library", Computer Physics Communications Vol. 159, No. 3 (2004) 157-184.