UCRL-TR-204783

SYMTRAN – A Time-dependent Symmetric Tandem Mirror Transport Code

D.D. Hua, T.K. Fowler

June 14, 2004

DISCLAIMER

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

SYMTRAN -- A Time-dependent Symmetric Tandem Mirror Transport Code

D. D. Hua* and T. K. Fowler
Department of Nuclear Engineering
University of California, Berkeley
June 14, 2004

*The work of Dr. Hua was funded by LLNL Grant B538660

1. Introduction

A time-dependent version of the steady-state radial transport model in symmetric tandem mirrors in Ref. [1] has been coded up and first tests performed. Our code, named SYMTRAN, is an adaptation of the earlier SPHERE code for spheromaks, now modified for tandem mirror physics. Motivated by Post's new concept of kinetic stabilization of symmetric mirrors, it is an extension of the earlier TAMRAC rate-equation code omitting radial transport [2], which successfully accounted for experimental results in TMX [3, 4]. The SYMTRAN code differs from the earlier tandem mirror radial transport code TMT [5] in that our code is focused on axisymmetric tandem mirrors and classical diffusion, whereas TMT emphasized non-ambipolar transport in TMX and MFTF-B due to yin-yang plugs and non-symmetric transitions between the plugs and axisymmetric center cell. Both codes exhibit interesting but different non-linear behavior.

While pulsed operating scenarios may be of interest, we have first focused on achieving a steady state reactor. We have also focused on classical diffusion which is possibly achievable in tandem mirrors as noted below. Examples of ignited steady states with classical radial diffusion are given in Sections 3 and 7. The highest fusion power gain achieved so far, not yet optimized, is Q ≈ 10.

Since it turns out that our highly non-linear tandem mirror model does not always yield a steady state, a study has been undertaken to determine conditions for a steady state. First, we note that convergence problems in our earlier steady state code [1] appear to be solved in the time-dependent version. Secondly, we sometimes encounter thermal instability common to all fusion reactors [6]. Criteria for thermal stability are developed that explain the apparent stability of TMX and predict stable regimes for reactors. Finally, and most importantly, we find that achieving steady state can require auxiliary heating that limits the fusion power gain Q.

The requirement for auxiliary heating appears to arise from the properties of classical transport near a cold boundary, usually at the plasma edge analogous to the "scrapeoff" in a tokamak. However, unlike a tokamak in which the scrapeoff occurs at a well-defined location outside the magnetic separatrix, in a tandem mirror scrapeoff-like

zones can also occur internally, yielding multiple hot cylinders separated by thin cold zones where temperatures plunge and end plugging is lost. With further work, it may be possible to provide the stabilizing power only by the end plugs, with externally controlled feedback to prevent instability.

We have also considered design requirements to avoid plasma microinstabilities in the end plugs and in the center cell, complimentary to other ongoing studies of MHD stability employing kinetic stabilization of the end plugs [7]. Maintaining plug stability sets requirements on the magnetic field and radius in the plug. For the center cell, the most prominent microinstability known from tokamaks -- the ion temperature gradient (ITG) mode -- is expected to be stabilized by the tandem mirror potential. The code includes a model of electron thermal gradient (ETG) transport thought to persist in tokamaks free of ITG.

This report is organized as follows. Equations solved by the code are given in Section 2. Test runs calculating the evolution of temperatures with prescribed density profiles are presented in Section 3 and analyzed in Section 4. Thermal stability, developed in Appendix A, is discussed in Section 5 and applied to TMX in Section 6. Test runs with the full set of equations, including an ignited steady state with Q ≈ 10, are discussed in Section 7. Plasma stability requirements are discussed in Section 8 and buildup to ignition in Section 9. Experimental tests of key issues are discussed in Section 10 and 11. Results are summarized in Section 12. The code and use of the code is discussed in Appendix B.

2. Equations

The SYMTRAN code solves the equations of Ref. [1] with the addition of time derivatives:

$$3/2\partial(n_C T_i/\partial t) + [n_C(\lambda_i + T_i)/\lambda_{||}] - r^{-1}\,\partial/\partial r\,(r n_C \chi_i \partial T_i/\partial r) = p_\alpha f_\alpha + P_{ION} + (n_C/\tau_e)(T_e - T_i) \equiv P_i \quad (1)$$

$$3/2\partial(n_C T_e/\partial t) + [n_C(\lambda_e + T_e)/\lambda_{||}] - r^{-1}\partial/\partial r\,(r\, n_C \chi_e \partial T_e/\partial r) = p_\alpha (1 - f_\alpha) + P_{ECH} + P_{PLUG} - (n_C/\tau_e)(T_e - T_i) - p_{Brem} \equiv P_e \quad (2)$$

$$\partial(n_C/\partial t) + (n_C/\lambda_{||}) - r^{-1}\partial/\partial r\,(rD\partial n_C/\partial r) = S_N = n_C n_o \sigma v \quad (3)$$

$$\partial(n_p/\partial t) + (n_p/\tau_p) = S_p = n_p N_b \sigma v \quad (4)$$

$$n_p \tau_p = (n_C \tau_{ie})[1 - (\tau_i + \tau_e)/E_B] \quad (5)$$

$$\tau_i = T_e \ln(n_p/n_C) \quad (6)$$

$$(\tau_e/T_e)\exp(\tau_e/T_e) = \tau_{||}/\tau_{ee}\{(2L_p/R_M L_C)(n_p^2/n_C) + n_C\}\{(2L_p/R_M L_C)n_p(\tau_{||}/\tau_p) + n_C\}^{-1} \cdot (m_i/m_e)^{1/2}(T_i/T_e)^{3/2}(\tau_i/T_i)\exp(\tau_i/T_i) \quad (7)$$

$$p_\alpha = 1/4\, n_C^2 (\sigma v)_{DT} E_\alpha \quad (8)$$

$$f_\alpha = \tau_{ie}(\langle E_\alpha \rangle - T_i)\{\tau_\alpha(\langle E_\alpha \rangle - T_e) + \tau_e(\langle E_\alpha \rangle - T_i)\}^{-1} \quad (9)$$

$$n_C \tau_\alpha = 1.6 \times 10^{16} \langle E_\alpha \rangle^{3/2} \quad (10)$$

$$\tau_{||} = \exp(\tau_i/T_i)\{\tau_i A(\tau_i/T_i) + \tau_s\} \quad (11)$$

$$\tau_s = 3.6 \times 10^{-6} R_M L_C T_i^{-1/2} \quad (12)$$

$$A = 1/4\, \pi^{1/2}[\ln(2R_M + 2)][2R_M/(2R_M + 2)]^{-1} \quad (13)$$

$$n_C \tau_i = 1.6 \times 10^{16} T_i^{3/2} = (m_i/m_e)^{1/2}(T_i/T_e)^{3/2} n_C \tau_{ee} \quad (14)$$

$$n_C \tau_e = 10^{18} T_e^{3/2} \quad (15)$$

$$p_{Brem} = Z_{eff}\ 5 \times 10^{-37} n_C^2 T_e^{1/2} \quad (\text{watts/m}^{-3}) \quad (16)$$

$$\tau_i = r_{Li}^2 \tau_i^{-1} + \tau_{ETG} \quad (17)$$

$$r_{Li} = 6.5 \times 10^{-3} T_i^{1/2} B_c^{-1} \quad (18)$$

$$\tau_e = (m_e T_i/m_i T_e)^{1/2} r_{Li}^2 \tau_i^{-1} + \tau_{ETG} \quad (19)$$

$$D = (m_e T_i/m_i T_e)^{1/2} r_{Li}^2 \tau_i^{-1} + 1/4\, \tau_{ETG} \quad (20)$$

$$\tau_{ETG} = C_{ETG}(T_e^{3/2}/B^2)[|dT_e/dr|/T_e - 1/L_{CRIT}] \quad \text{zero if} < 0 \quad (21)$$

$$P_{PLUG} = 2(n_p^2/n_c \tau_{ie}) R_M^{-1}(L_p/L_c) E_B \quad (22)$$

$$R_M = (B_p/B_c) \quad (23)$$

Subscript p denotes plug quantities and c denotes central cell quantities. The rationale for these equations is discussed in Ref. [1]. New features not in Ref. [1] include direct heating of the ions by alphas by the fraction f_α in Eq. (9) with mean alpha energy $\langle E_\alpha \rangle \approx 2000$ KeV ($>> T_e$ and T_i). Quasi-neutrality allows us to calculate a single center cell particle density equation, Eq. (3). We also include Eq. (4) for the plug density. In Eq. (5), we neglect plug ion scattering included in Ref. [2] and assume that the plug lifetime is determined by electron drag at the center cell temperature. We do not include an energy equation for the plug ions, but the factor $[1 - (\phi_i + \phi_e)/E_B]$ in Eq. (5) accounts for plug ion escape when their energy falls to the ambipolar hole energy $(\phi_i + \phi_e)$ [2]. Also in Eq. (5), N_B is the neutral beam density proportional to beam current; and σv is the ionization rate (ions and electrons). In Eq. (3) n_o is the gas feed density, treated more exactly in Ref. [2]. Unlike Ref. [2], we do not yet include a separate equation for the alphas, so helium ash buildup is not calculated.

Four power inputs not included in Ref. [1] are the auxiliary ion heating P_{ION} (which can be turned on and off in feedback response to changes in the ion temperature) and ECH heating of electrons P_{ECH} , which can be turned on or off, and bremstrahlung p_{Brem} , and P_{PLUG} representing heating of center cell electrons close-coupled to heating of electrons in the two plugs by beam ions (diluted by $2R_M^{-1}$ (L_p/L_c), the ratio of the volume of the two plugs of length L_p to that of the center cell L_C). Typically the plug length is $L_p = R_{PLUG} = R_C/ R_M$ where R_{PLUG} is the plug radius projected along field lines into the center cell (the plug "shadow"). The auxiliary powers P_i and P_{ECH} can also be turned on temporarily to achieve ignition if plug heating P_{PLUG} is insufficient to do this.

End loss formulas in SYMTRAN have been updated to be approximately those in the TAMRAC code [2] and TMT code [5]. Eq. (11) is a composite formula, the first term being the Pastukhov formula applicable for mean free path greater than L_c and the second term being that for short mean free path, proportional to the free flow lifetime τ_s. When the mean free path is small but the end plug potential is large, the only ions experiencing free flow are those in the combined loss cone due to R_M and ϕ_i [2, 9], giving the factor R_M in τ_s and an additional factor $\exp \phi_i /T_i$ out front that also applies to the Pastukhov term. The formula is still valid in a cold "scrapeoff" zone where $\exp \phi_i /T_i = 1$ and endplugging is ineffective, in which case the mirror ratio factor in τ_s still applies, since the useful plug radius is smaller than the mirror throat so that scrapeoff power flows through the mirror. To

include such scrapeoff zones, in the code we set $\chi_i = 0$ if $n_C \geq n_p$. Then Eq. (11) covers all regimes.

We will assume that the end plugs are stable to mirror loss cone instabilities, applicable only if scenarios yield ratios of plug radius to plug ion Larmor radius > X to avoid DCLC instability, where X is of order 40 in a reactor [10]. (Stream stabilization by end losses from the center cell is inconsequential for reactor parameters.) We assume the Alfven Ion Cyclotron (AIC) mode is stabilized by aiming the neutral beams to produce "sloshing ions" [8].

We generally assume classical radial transport, with classical particle transport in D equal to the classical electron thermal diffusivity in χ_e. However, we have provided an option to include turbulent transport, χ_{ETG}, due to electron temperature gradient (ETG) instability. The MHD-stable symmetric tandem mirror with no internal currents is not likely to suffer magnetic turbulence, and the strong ion temperature gradient (ITG) mode is probably stabilized by the tandem mirror potential profile, as it can be in tokamaks in the H mode; H-mode-like behavior has been reported in tandem mirrors [11]. However, evidence in tokamaks suggests that the electrostatic ETG mode may persist. The term χ_{ETG}, with a threshold at a critical temperature gradient $L_{CRIT} \approx 3/2 \left| (n/(\partial n/\partial r) \right|$ [12], accounts for this possibility, with a coefficient C_{ETG} to turn it on or off. Consistent with experimental evidence in DIIID, we apply the same χ_{ETG} to electrons and to ions and 1/4 χ_{ETG} to the particle diffusion coefficient D [13]. In Eqs. (1) and (2), we have neglected $\partial n/\partial r$'s in classical heat flow, taken as $q = - n\chi\partial T/\partial r$ (correct for ETG).

The potentials are calculated assuming ambipolarity, in contrast with the careful attention to non-ambipolarity for tandem mirrors with yin-yang plugs in Ref. [5]. We justify this in the symmetric tandem mirror since, aside from transport by neutrals which we neglect, collisional (Coulomb) transport in a symmetric tandem mirror is inherently ambipolar (as is ETG transport mentioned above). The ambipolar potential on each flux surface is calculated using Eqs. (6) and (7), where Eq. (7) balances end losses of ions and electrons (in the limit of long mean free path). In assuming ambipolarity, we are neglecting weak non-ambipolar currents due to ionization in the presence of a radial electric field. These weak currents may be essential in shaping the plasma potential to avoid rotational instability (see Section 8).

Early in our work, we decided to approximate χ_e by the expression farthest to the right in Eq. (7), appropriate for the core of a long tandem mirror reactor dominated by the center cell. We will make a different approximation in the scrapeoff, where the Pastukhov term is small, and in Section 11, where the plugs dominate.

The boundary conditions employed are standard, with zero derivatives at the origin and temperatures and densities set at low values at an outer boundary that defines the plug shadow.

3. Test Runs with Fixed Density Profiles

In this section we discuss test runs of the code, first using only Eqs. (1) and (2) with specified density profiles for the plug and center cell. Initial conditions on $T(r/R_C)$ for both ions and electrons have the form:

$$T(r/R_C) = T(1) + [T(0) - T(1)][1 - (r/R_C)^2]$$

where $T(0) = 0.002$ and $T(1) = 0.001$.

Table 1. Test Runs with Prescribed Density Profiles

Run	P_{ECH}	n_{po}	C_{Te}	$f_{\square}$	Stable
1	10	10	0	0	No
2	0	10	7	0	No
3	50	10	7	Eq.9	No
4	70	10	7	Eq.9	Q = 2.3
5	50	25	7	Eq.9	Q = 5.4
6	70	40	7	Eq.9	Q = 2.3

Parameters for six test runs are listed in Table 1. In all runs, $B = 1$, $R_M = 9.2$, $E_B = 1000$, $L_C = 30$, $L_p = R_C \quad R_M$ and $R_C = 1.5$. Also, in all runs ignition is initiated by a short power burst $P_e = 1000$ added to Eq. (2), until T_e at $r = 0$ reaches 40 KeV. This is followed by steady P_{ECH} where indicated in the table (and $P_{PLUG} = 0$). Here and hereafter, powers are in code units, wherein energies and temperatures are in KeV and all terms of Eqs. (1) - (4) are divided by 10^{20} , giving densities in units 10^{20} m^{-3}. Other units are MKS. The radial variable r is scaled to R_C by dividing diffusion coefficients by R_C^2.

The center cell density n_C is constant in Runs 1 - 5. For these cases, the scrapeoff is defined by setting $\square_i = 0$ and $\square_e = 4T_e$ in a fixed zone $0.98 < r/R_C < 1$. This $\square_e$ is typical of free flow.

For Run 6, both n_p and n_C have parabolic form with $n_C = 1$ at $r = 0$:

$$n_p = n_{po}\,[1 - C_{Te}T_e/E_B][1 - (r/R_C)^2] \quad , \quad n_C = [1 - (r/R_C)^2]$$

For this case, the scrapeoff is defined as in the general case in Section 2, by setting $\Phi_i = 0$ if $n_C \geq 1$. Having approximated Φ_e in the Pastukhov limit in Eq. (7), setting $\Phi_i = 0$ incorrectly gives $\Phi_e = 0$ by Eq. (7) and hence an underestimate of electron heat loss.

For Runs 1 -5, n_p is given by:

$$n_p = n_{po} [1 - C_{Te} T_e / E_B]$$

Here the factor in brackets is obtained from the steady state of Eq. (4) with potentials in Eq. (5) approximated by $\Phi_i + \Phi_e = C_{te} T_e$. This factor applies stabilizing feedback as we shall see.

Runs that did not reach steady state are indicated by "No" under the heading "Stable" in Table 1. Stable runs are identified by the fusion power gain Q obtained in steady state, where:

$$Q = [\int rdr\, 5P_\alpha / \int rdr\, P_{ECH}] \qquad (24)$$

The alpha power was applied only to electrons in Runs 1 and 2, but split between electrons and ions using f_α from Eq. (9) in Runs 3 - 6.

Figure 1 shows results for Run 1 in which plug feedback is negated by setting $C_{Te} = 0$. Weak ECH power $P_{ECH} = 10$ is applied after the initial kick to ignition. Ignition is in full swing ($T > 10$ KeV) by $t = 1$ s. However, without plug feedback, T_e runs away to extreme values, causing decoupling from the ions, so that T_i begins to drop around $t = 8$ s.

Figures 2 - 5 show results from Run 2 in which plug feedback is applied, with $C_{Te} = 7$ (but no ECH power after the initial "kick" to ignition). Now runaway of the electron temperature is prevented, as seen in Figure 2. Nonetheless, core temperatures crash at $t = 2$ s. The progression of the crash is seen in Figure 3, two cold boundaries having developed by $t = 2.7$ yielding two scrapeoff zones in Figure 4. Figure 5 shows results of an attempt to prevent the crash by supplying extra ECH to maintain the scrapeoff in its initial location. However, the crash persists.

Figures 6 - 7 show results of Run 3, which differs from Run 2 only by restoring constant $P_{ECH} = 50$ after the initial kick. A crash still occurs, now with a scrapeoff-like zone appearing internally, around $r/R_C = 0.3$ as seen in Figure 6, with a corresponding local increase in end losses, as seen in Figure 7.

Figures 8 - 12 show results of Run 4, which reaches a stable steady state, as seen in Figure 8. This run differs from Run 3 only in that the auxiliary power is increased to $P_{ECH} =$

70, which proves to be sufficient to stabilize the system. Profiles of temperatures and potentials are smooth (except at the edge) as seen in Figures 9 and 10. Figure 12 gives the instantaneous fusion power gain Q. The steady state value, Q = 2.3, is to be compared with Q = 10 to 20 for cases in Ref. [1] in which radial transport was omitted and alpha power alone was sufficient to balance end losses, so that Q increases with reactor length. The lower Q here arises from the spatial average of $P_{\perp}$ and, most importantly, the requirement for a minimum power density P_{ECH} to maintain a stable state.

Continuing with our examples, Figures 13 - 16 show results of Run 5, which is similar to Run 4 except that the ratio of end plug and center cell densities is higher and the power required for stability is less, giving a higher Q = 5.4.

Figures 17 - 20 show results of Run 6, which is similar to Run 5 except that now the density profiles are parabolic. Parabolic averaging and higher auxiliary power reduces the Q back to Q = 2.3, as seen in Figure 20.

4. Discussion of Test Runs: Tandem Mirror Edge Physics

Test Runs 1, 2 and 3 indicate that sometimes there is no steady state for our model equations, or, if there is a steady state, this state is unstable. These results concern two physical phenomena. First, like all fusion reactors, tandem mirror reactors are subject to thermal instability in the burning core, due to runaway alpha production as the ion temperature increases [6]. In tandem mirrors this alpha-driven thermal instability is usually overshadowed by end losses, destabilizing at fixed plug density (as found in Run 1) but stabilized by plug feedback as discussed in Section 5.

The unexpected effect is thermal collapse at the edge or in the interior, present even with plug feedback in Runs 2 and 3 and many other runs. There appear to be two main points illustrated by the test runs.

First, auxiliary power is required to sustain the edge in steady state. While this power would not be excessive applied only at the edge, the fact that it must be applied over an appreciable portion of the volume to prevent collapse greatly exaggerates its influence on Q, as found in the test runs. This appears to be a different kind of thermal instability arising from the following. Unlike a tokamak, for which our boundary conditions would firmly fix the scrapeoff region in space outside the magnetic separatrix, in the tandem mirror the large end losses at the free flow rate characteristic of a scrapeoff region can occur wherever temperatures fall low enough so that $\tau_{||}$ in Eq.(11) goes over to the free flow value τ_s applicable to the scrapeoff. This appears to be what is happening in Run 3, Figures 6 and 7, where as temperatures drop in the interior, end losses rise, causing a further drop in temperature and eventual collapse. Something similar occurs in Run 2, where the edge

recedes inward and temperatures finally collapse in the interior as seen in Figure 3. In Figure 5, the local application of power in the nominal scrapeoff region does not prevent the collapse, indicating a tendency to recreate collapse in the interior.

Secondly, in interesting regimes that exploit the potential of tandem mirrors to operate stably with classical diffusion, alpha power cannot supply the power required to sustain the edge, so that plug heating or auxiliary heating must supply this power. This is seen in all test runs shown in Section 3, evidenced by an outward rise in electron temperature until it finally reaches a peak value near the edge where the nearby presence of a cold boundary causes electron diffusion to become large. It is this peak in electron temperature that blocks the outward flow of alpha heating in the interior to sustain the edge, and, since $T_e > T_i$ ions cannot heat the electrons. Hence the power to drive electron diffusion near the edge must be supplied locally, either by electron heating by the end plugs, or by auxiliary heating. This qualitative feature occurs in all test runs.

The following provides a qualitative understanding of these points.

First, we calculate the heat flux required to sustain the scrapeoff. End loss at the free flow rate characteristic of the scrapeoff is triggered by a combination of low ion temperature and strong coupling between ions and electrons giving small confinement enhancement by the ion potential. This occurs not at a fixed position but roughly at the temperature T_S obtained by setting $T_e = T_i$ and $\tau_s = \tau_e$ (from Eqs. (12) and (15)), giving:

$$T_S = 0.019 (n_C R_M L_c)^{1/2} \qquad (25)$$

For classical ion heat transport, giving a scrapeoff width $\Delta \sim \Delta_s$, the heat flux q_S to the scrapeoff is fixed at:

$$q_S \sim (nT/\tau_s) \sim 5 (n_C^2/B_C) \qquad (26)$$

where temperatures are evaluated at T_S. We will use this value of the heat flux to calculate the P_{ECH} required to sustain the stable example in Run 4. All quantities are in code units, discussed in Section 3.

Secondly, we can demonstrate why alpha heating and classical diffusion usually yield the "hollow" T_e profiles characteristic of all of the test runs in Section 3, in which the peak of T_e (r) occurs off-axis near the edge. An example is shown in Table 2.

Table 2. Approximate Analytical Temperature Profiles for an Ignited State

T_e/T_i	n_p/n_C	T_i	T_e	χ_i/T_i	χ_e/T_e	$(\partial\chi_i/\partial r)/S_C T_i$	$(\chi_i/T_i - \chi_e/T_e)$	χ_p/χ_{Ei}
3	5.4	22	66	5.0	7.2	− 0	2.2	0.39
4	4.3	17	68	5.8	7.6	+ 6	1.8	0.35
9	2.4	9	80	7.8	8.5	+ 11	0.7	0.25
20	1.6	7	140	9.6	9.4	+ 560	- 0.2	0.20

Table 2 is calculated as follows. We omit both electron diffusion and particle diffusion in the hot interior, where the Pastukhov term is dominant in Eq. (11). Then, for given beam energy and density E_B and N_B and given gas feed density n_o (or fixed S_p and S_C, or fixed n_p/n_C), the steady state versions of Eqs.(3) - (7) uniquely determine the ratios χ_i/T_i, χ_e/T_e and n_p/n_C for given T_e/T_i if we approximate $f_\chi = 0$. Using these ratios in the steady state of Eq. (2) determines T_i, from which T_e and the potentials follow. Finally, inserting these quantities into the steady state of Eq. (1) determines the ion heat flux $\chi_i = - n_C \chi_i \partial T_i /\partial r$, from which profiles can be developed. When plug heating is included, there are two solutions, one dominated by P_{ECH} and another by p_χ. We choose the latter, indicative of an ignited state. In Table 2 below, we give numbers for this solution for fixed $S_p/S_C = 80$ and Pastukhov parameter A = 1.5. We only list values yielding outward diffusion ($\partial\chi_i/\partial r > 0$), from which a radial profile of T_i and the other quantities could be constructed.

The top line of Table 2 is similar to the interesting reactor case of Ref. [1] yielding, in the absence of diffusion, a 250 MWe reactor 30 m in length with Q = 10; and higher Q at longer length and higher power. These values represent temperatures in the hot plasma core, while those at successively lower T_i represent entry into the pedestal. As can be seen from the table, given our assumptions a T_i decreasing outward requires an increasing T_e. This appears to be a characteristic of classical diffusion, evident in all test runs discussed in Section 3. The last two columns in the table, relevant to thermal stability, will be discussed in Section 5.

These results provide a semi-quantitative explanation of the stable test Run 4. For this test run, $n_C = 1$, $R_C = 1.5$, B = 1 and $\chi_S = 5$. As can be seen in Figure 11, end losses are negligible where the electron temperature falls at the edge and electron-ion heat transfer is small there also. Then we can ignore electron end losses and heat exchange other than P_{ECH} and solve Eq. (2) analytically:

$$T_e(r) = \{(T_{e\,MAX})^{1/2} - (P_{ECH}/4n_C G)(r - (R - \chi))^2\}^2 \qquad (27)$$

Here we used $\chi_e = G/\ T_e$ with $G = 0.0035\ n_C/B^2$, and Δ is the width of the narrow region where T_e falls to T_S . Also, we have applied the boundary conditions $T_e = T_{e\,MAX}$ and $\chi_e = -n_C\chi_e\partial T_e/\partial r = 0$ at $r = R - \Delta$ where T_e is maximum. Both Δ and P_{ECH} are as yet undetermined. We can determine Δ in terms of P_{ECH} by the condition $\chi_e \le \chi_S$ (= 5) at r = R, giving:

$$P_{ECH}\ \Delta \qquad \le \qquad \chi_S \tag{28}$$

Finally, the minimum P_{ECH} is obtained in terms of $T_{e\,MAX}$ by introducing Δ from Eq. (28) into Eq. (27) and requiring $T_e(R) \to 0$, giving:

$$(P_{ECH})_{MIN} \qquad \le \qquad (\chi_S^2/4n_C G\ T_{e\,MAX}) \ = \ 1800(n_C^2/\ T_{e\,MAX}) \tag{29}$$

In principle $T_{e\,MAX}$ could be determined by matching T_e (r) in Eq. (27), in which electron diffusion dominates, to that from Table 2, where electron diffusion was neglected. Here we will take the peak value $T_{e\,MAX}$ = 130 KeV from the code results that agree qualitatively with the Table. Then Eq. (29) gives a $(P_{ECH})_{MIN} \le 157$ versus $(P_{ECH})_{MIN} = 70$ found by the code. The width Δ is also determined, giving $\Delta/R \ge 0.03$ similar to Figure 9.

5. Thermal Stability in Tandem Mirrors

Even when a steady state exists, our code (and a reactor) will fail to find it if this state is thermally unstable [6]. Physically, stability of a state P = L (power = loss) requires that any departure from steady state causes P - L to vary so as to push the system back to steady state. For example, focusing on temperature, stability requires $\partial(P - L)/\partial T < 0$ so that an increase in temperature increases losses compared to power, thereby causing the temperature to fall again, while a decrease in T gives excess power to make T rise.

A. Necessary Conditions for Thermal Stability of the Burning Core

Stability analysis can be formalized by linearizing the transport equations, Eqs. (1) - (4), again resorting to partial derivatives with respect to temperatures and densities to explore around a steady state. This is most easily carried out ignoring radial diffusion, applicable in the burning core of a tandem mirror reactor where we expect temperature profiles to be relatively flat (equivalent to the TAMRAC rate equations of Ref. [2]). In the spirit of WKB approximations, the stability criteria without diffusion can also be applied locally to see effects of perturbing end losses at constant diffusion rates. The complete set

of linearized equations and our analysis is given in Appendix A. Here we summarize the results.

Our analysis of the resulting dispersion relation (quartic in p for quantities varying as μ exp pt) is carried out in terms of A_3, the p^3 coefficient equal to the negative sum of the roots. Physically the real part of A_3 is the negative of the sum of growth rates and decay rates of the densities and temperatures as if there were no coupling between them. A necessary condition for stability is shown to be $A_3 > 0$ (since otherwise some root has a positive real part). In the reactor regime, this gives:

$$A_3 \quad \mu \; -(1/\tau_{Ei})(\tau_e/T_e - \tau_i/T_i) + (1/\tau_p) \quad > \quad 0 \qquad (30)$$

Here τ_p is the plug ion lifetime and τ_{Ei} is the ion energy confinement time in the center cell given by:

$$\tau_{Ei} \quad = \quad \tau_{\parallel}\,(\tau_i/T_i + 1)^{-1} \qquad (31)$$

We can also say that $A_3 < 0$ guarantees instability:

$$A_3 \quad < \quad 0 \qquad \text{sufficient condition for thermal instability} \qquad (32)$$

The quantity A_3 includes the main source of thermal instability in tandem mirrors, coming from the electron heat loss through the plugs (the term $\mu\ \tau_e$). That the timescale is numerically equal to the ion energy confinement time is an artifact of the Pastukhov formula for $\tau_{\parallel}$.

Electron end loss is destabilizing because $\tau_{\parallel}$ increases and the end loss decreases as $\tau_e \mu\ T_e$ increases. With one exception, other physical effects identified in Appendix A -- including particle end losses (as distinct from energy losses) and both plug heating and electron-ion energy exchange -- are all stabilizing. (The exception is alpha heating, which is destabilizing in all fusion reactors for desirable ion temperatures < 50 KeV [6], but which is roughly offset by stabilizing plug heating in a tandem mirror reactor.) However, despite the many stabilizing effects, in the reactor regime the large value of τ_e required for good confinement of electron energy dominates.

The dominant stabilizing effect is end plug losses (the $1/\tau_p$ term in Eq. (30)), which appears to be the main stabilizing effect useful in reactor design. Plug losses are stabilizing because reducing n_p reduces $\tau_i \mu \ln n_p/n_c$.

Electron destabilization is partly offset by ion stabilization (the Δ_i/T_i term in Eq. (30)), typically yielding $(\Delta_e/T_e - \Delta_i/T_i) \approx \ln[(m_i/m_e)(T_i/T_e)^{3/2}] \approx 2.5$. Then, Eq. (30) becomes, using Eq. (5):

$$\Delta_p \equiv \Delta_{Ie}(n_C/n_p)[1 - (\Delta_i + \Delta_e)/E_B] < 0.4\,\Delta_{Ei} \qquad (33)$$

Though Eq. (30) is actually only a necessary condition for stability, we will apply it as our rule of thumb to find stable solutions and reactor designs. This criterion, based on stabilization by the end plugs, is advantageous to design. Whereas center cell losses have a fixed ratio to alpha heating, energy losses in the plug can be offset by increasing the center cell length L_C to produce more alpha power. Thus thermal stabilization by the plugs only sets the scale of tandem mirror reactors, not their ultimate feasibility.

B. Stability of the Pedestal

We have not attempted to carry out a compete thermal stability analysis including the pedestal and scrapeoff, which requires finding both steady state profiles and spatial eigenstates of the linearized equations including diffusion. Some light can be shed on stability in the pedestal by returning to the example of Table 2. The last two columns in Table 2 give quantities appearing in our thermal stability criterion, Eq.(30). Applying the criterion locally, we see that Δ_p/Δ_{Ei} decreases and also $(\Delta_i/T_i - \Delta_e/T_e)$ decreases as ion temperatures drop in the pedestal, indicating thermal stability. Thus if Eq. (30) is satisfied in the core, it is satisfied locally in the pedestal. However, our stability analysis applies only in the Pasthukov limit of $\Delta_{||}$, leaving open the possibility of instability around a scraepoff region where $\Delta_{||} \approx \Delta_s$ and ion end losses might become destabilizing. Also, the fact that a constant P_{ECH} $(\partial P/\partial T = 0)$ is "stabilizing" may only indicate lack of equilibrium.

6. Thermal Stability in TMX

Plasmas in TMX experiments, discussed in Refs. [3], [4] and [8], appeared to be thermally stable, with enough power to sustain the scrapeoff layer (referred to as the "halo" in Ref. [8]). Also, since alpha heating was not involved, the solution for TMX analogous to Table 2 is that dominated by P_{ECH}, which turns out not to exhibit the hollow electron temperature profile that played a prominent role in our discussion of ignited test runs in Sections 4 and 5.

With an available power typically > 100 kW in excess of losses by charge exchange, ionization and radiation, and a surface area of order 4 m^2, there was an available heat flux up

to 25 kW/m^2, versus < 1 kW/m^2 into the scrapeoff by Eq. (26) (times 16 to obtain kW with $n_C = 0.03$, $B_C = 0.2$). However, Eq. (26) applies to classical transport in axisymmetric geometry. The actual radial heat flow in TMX was more typically 15 - 30 kW due to asymmetry-induced "neo-classical" transport [8].

Concerning thermal stability, we first consider the following data from the most detailed published power balance in TMX in Ref. [8]:

V_C	=	0.4 m^3	center cell volume
V_p	=	0.003 m^3	plug volume
L_p	=	0.4	plug length
R_C	=	0.2	center cell radius
B_C	=	0.2	center cell field
n_p	=	0.15	
n_C	=	0.03	($n_p / n_C = 5$)
T_e	=	0.05 KeV	center cell (0.09 KeV in plugs)
T_i	=	0.06 KeV	
I	=	200 A	end plug neutral beam current
I_{TRAP}	=	15 A	fraction trapped
E_B	=	13 KeV	mean energy of trapped ions, 9 KeV

This gives energy stored in the center cell = 30 J. Power balance accounting for 85% of the power showed end losses ≅ 35 kW (1/3 of the net power excluding charge exchange, ionization and radiation). Hence:

$$\tau_{Ei} \geq \tau_E = (30\ J/35\ kW) = 1\ ms$$

To calculate the plug ion lifetime, we note that of the 15 A trapped, 5 A is lost to charge exchange, leaving a net 10 A yielding a plug ion charge 0.007 Coulomb, hence:

$$\tau_p = (0.007C/10A) = 0.7\ ms$$

This τ_p is comparable to the calculated value from Eq. (5) giving $\tau_p \cong (n_C \tau_e)/n_p = 0.7$ms at the center cell electron temperature. Thus our simplified thermal stability criterion, Eq. (33), was only marginally satisfied. The full criterion, Eq. (A24) in Appendix A, does indicate stability, giving a destabilizing end loss term $\cong -4/\tau_{||}$ offset by a stabilizing plug heating term $3/2\ (P_{PLUG}/n_C T_e) \cong +9/\tau_{||}$ in addition to the term $1/\tau_p$.

For this data, with $n_p/n_C = 5$, ion cyclotron noise was evident (adding to the heating of center cell ions). However, the highest values of $n_C\tau_{\parallel}$ were obtained when $n_p/n_C \leq 3$ which suppressed ion cyclotron noise [4]. This is the regime to which our code model applies.

Results in TMX were reproduced fairly well by the TAMRAC code [2] and degradation of confinement when cyclotron noise was present was also accounted for based on a theoretical model described in Ref. [14]. The TAMRAC rate equation code, to which the analysis of Appendix A applies, initially encountered thermal instability of solutions but final results including more physics were stable [15].

One feature of TAMRAC simulations of TMX not included in SYMTRAN was stream stabilization of the DCLC mode. Stream stabilization was simulated by requiring that the current escaping the center cell (plus any externally supplied stream) equal that required to stabilize the DCLC mode, together with associated plug heating losses that perhaps produced additional stabilizing plug-feedback.

Studies of alpha heating using TAMRAC also found stable solutions, one difference being a focus on helium ash buildup in TAMRAC but not yet included in our code [2].

7. Test Runs with the Full SYMTRAN Code (Transport Eqs. (1) - (4))

While test runs discussed above used only Eqs. (1) and (2), the SYMTRAN code is operative with the full set of equations that simultaneously calculate the time evolution of the electron temperature, the ion temperature, the center cell density and the plug density.

Parameters for three test runs with the full code are listed in Table 3. Initial conditions for temperatures are the same as those in Section 3, and initial conditions on densities have the same form with $n(0) = 0.02$ and $n(1) = 0.01$ for n_C and 1.2 times these values for n_p. Other parameters for these runs are $B = 0.5$, $R_C = 1.5$, $E_B = 1000$, and, in Eqs. (3) and (4), $S_{nc} \equiv n_o\tau_v = 5$ and $C_{np} \equiv N_B\tau_v = 100$. Also, now P_{PLUG} is activated in Eq. (2). Using Eq. (22), we take P_{PLUG} in the form:

$$P_{PLUG} = X\,(n_p^{\,2}/n_c\tau_e)E_B$$

where for convenience X was a prescribed constant in Runs 7 and 8, but in Run 9 it is self-consistently calculated as $X = 2\,(V_p/V_C) = R_M^{-1}\,(L_p/L_c)$ as in Eq. (22). Here we took $L_p = R_C/\,R_M$. Including P_{PLUG} also gives a revised form for Q:

$$Q = \{\int rdr\; 5P_\tau / [\int rdr[P_{PLUG}\,(1 - (\tau_i + \tau_e)/E_B)^{-1} + P_{ECH}]\} \qquad (34)$$

Here the factor $(1 - (\phi_i + \phi_e)/E_B)^{-1}$ accounts for the loss of plug ions at energy $(\phi_i + \phi_e)$ as in Eq. (5), giving for Q the total fusion power divided by the sum of ECH power and the total input power to the plug.

Table 3. Parameters for test runs of the full code, Eqs. (1) - (4)

Run	P_{ECH}	R_M	L_C	X	Stable
7	0	20	100	1.7×10^{-3}	No
8	2×10^4	20	100	1.7×10^{-3}	$Q \approx 10$
9	2×10^4	9	80	Eq. (22)	$Q \approx 9.5$

Figures 21 - 23 show results for Run 7 with plug heating but no auxiliary heating. This run, which crashed around 9.5 seconds, is to be compared with the unstable Runs 2 and 3 with plug feedback and prescribed density profiles. The additional equations for plug and central cell densities have added richness in non-linear feedback giving results reminiscent of Run 3, but with even more regions of temperature collapse in the interior as seen in Figures 22 and 23.

As in the runs of Section 3 with prescribed density profiles, adding sufficient auxiliary heating P_{ECH} was found to stabilize the system. Best success was obtained if this heating is delayed. In the examples below, ECH is turned on when the electron temperature reaches $T_e = 60$ KeV.

Figures 24 - 27 show results of Run 8 in which auxiliary heating added to Run 7 does stabilize the system. Here the final density ratio is $n_p/n_C \approx 6$ at $r = 0$, similar to reactor cases in Ref. [1] giving Q = 10 - 20. As seen in Figure 27, Run 8 gives an average Q = 10. In this run, P_{PLUG} happened to be given a value inconsistent with the length and mirror ratio used in calculating ϕ_s (giving too low end loss at short mean free path). The stabilizing power for this run is $P_{ECH} = 20{,}000$, versus $(P_{ECH})_{MIN} \leq 200{,}000$ by the crude criterion of Eq. (29) with $n_C \approx 40$ and $T_{eMAX} \approx 140$ for this run.

In contrast with Run 7, for the stable Run 8 the density, temperature and potential profiles are fairly smooth, as seen in Figures 25 and 26. However, note the persisting temporal oscillations in Figure 24. The amplitude of oscillations is reduced as P_{ECH} is increased, at some sacrifice in Q. Though the oscillating state appears to be stable through the 20-second duration of the run, longer runs would be required to determine whether oscillations eventually grow or damp (perhaps a moot point, since the burn cycle may require periodic flushing of helium ash).

A different example with greater oscillation amplitudes but about the same average Q was obtained in Run 9, shown in Figures 28 - 30. In Figure 28, note the limit-cycle oscillation of n_p versus T_e, 180° out of phase. Aside from the densities, this run has reactor parameters similar to those in Reference [1], with all quantities calculated consistently for $R_M = 9$, $L_c = 80$, $R_C = 1.5$ and $B_C = 0.5$. However, the product $B_C R_C = 0.75$ is not yet large enough for plug stability (see discussion below).

Figures 31 and 32 show results of Run 10, which is identical with Run 9 except that Ω_{ETG} has been applied to Ω_e only, with a coefficient $C_{ETG} = 0.1$. Also, to stabilize this run, the ECH power is higher ($P_{ECH} = 5 \times 10^4$), giving a lower $Q \approx 4$, and ECH had to be applied earlier, when T_e first reaches 40 KeV. A lower $C_{ETG} = 0.01$ still gave $Q \approx 10$ with the lower P_{ECH} of Run 9, applied earlier. Temperatures in Figure 31 are similar to those of Run 9, but the electron temperature profile is quite different, as seen in Figure 32. More work is needed to understand ETG transport in tandem mirrors, both concerning the threshold for such transport to occur (if any) and also the consequences of ETG transport if it does occur.

In all runs discussed here, achieving buildup to an ignited state required density sources in Eqs. (3) and (4) that give densities an order of magnitude above those for reactor scenarios in Ref. [1]. However, the quasi-steady states achieved in Runs 8 and 9 can be scaled to lower density since, in steady state, all terms in Eqs. (1) - (4) scale as n_C^2 in the Pastukhov limit, including the diffusion term for classical diffusion and also the required auxiliary heating to prevent temperature collapse by Eq. (29). More realistic buildup strategies are discussed in Section 9.

8. Plasma Stability Requirements

In addition to a favorable energy balance and thermal stability, suitable reactor steady states must be stable to the DCLC mode in the end plugs and to MHD, rotational and ITG modes in the center cell.

Stability to ion cyclotron modes must be maintained for the end plugs. Stream stabilization utilized in TMX is not feasibile for reactors because streaming out of the center cell is negligible compared to requirements for stream stabilization, and because external streams would extract unacceptable power from the plugs or stream guns. Thus, in a reactor, the DCLC mode must be stabilized, the condition being a plug radius $R_p \approx 40\ r_{Lp}$ at reactor densities (r_{Lp} = plug ion Larmor radius) [10]. This places conditions on the end plug field as follows:

$$B_p R_p = 4.6 \times 10^{-3} (R_p/r_{Lp})(ME_i)^{1/2} \qquad \text{DCLC stability} \qquad (35)$$

where M is the ion mass in ratio to hydrogen (M = 2.5 for DT) and in a reactor $E_i \equiv E_B$, the neutral beam injection energy.

Turning to center cell stability, other studies indicate that kinetic stabilization allows a center cell beta up to 50% [7], which, for $T_e/T_i \geq 2$, gives for the ion beta:

$$\beta_i \quad \approx \quad 1/3\,\beta \quad \leq \quad 0.15 \qquad \text{MHD stability} \qquad (36)$$

We can use this result to estimate requirements for stability against rotation due to the plasma potential, giving stability if:

$$\omega_{ROT} \quad = \quad [(E_r/B_C)^2/R_C\rho_p]^{1/2} \quad \leq \alpha\,(v_A/L_C) \quad \equiv \quad \alpha\,(v_i/\rho_i^{\,1/2}L_C) \qquad (37)$$

where ρ_p is a scale length for pressure. We estimate the electric field E_r from $\phi_e \mu\ T_e$ with T_e approximated by Eqs. (27) - (29):

$$E_r \quad = \quad 4(\phi_e/\rho_\phi^{\,2})\, x\{1 - (x/\rho_\phi)^2\} \quad \leq \quad 770(E_B/\rho_\phi) \qquad (38)$$

where on the far right we take ϕ_e (in volts) $\leq 1000(1/2\ E_B(\text{KeV}))$ and the extremum at $x \equiv r - (R - \rho_\phi) = \rho_\phi/\ 3$, where ρ_ϕ is the potential scale length. Substituting Eqs. (35), (36) and (38) into Eq. (37) with $v_i = 2.7 \times 10^5\ \ T_i$ for DT fuel gives a rough criterion for rotational stability consistent with DCLC stability and MHD stability:

$$L_C/R_C \quad \leq \quad 2800(\rho_p/R_C)^{1/2}(\rho_\phi/R_C)(T_i/E_B R_M)^{1/2} \qquad \text{Rotational stability} \qquad (39)$$

In Sections 1 and 2 we mentioned the possibility that the ITG mode prominent in tokamak radial transport analysis is prevented by the tandem mirror potential. A crude criterion for electric shear suppression of the ITG turbulence is:

$$|\partial(E_r/B_C)/\partial r| \quad \geq \quad \omega_{ROT} \quad = \quad \alpha_{ITG}\omega_T^* \quad = \quad \alpha_{ITG}\,(v_i/\rho_T) \qquad (40)$$

where $\alpha_{ITG} < 1$ and ρ_T is the ion temperature gradient length. Using Eq. (38) and v_i above and taking the extremum value for $\partial E/\partial r$, Eq.(40) gives:

$$(\rho_\phi^{\,2}/\rho_T R_C) \quad \leq \quad (0.05/\alpha_{ITG})\,(T_i/E_B R_M)^{-1/2} \qquad \text{ITG stability} \qquad (41)$$

Let us evaluate the stability criteria of Eqs. (35), (36), (39) and (41) for typical reactor parameters of Section 7 ($T_i = 40$, $E_B = 1000$, $R_M = 9$) and $R_p/r_{Lp} = 40$. The results are:

$B_C R_C$	>	3	DCLC stability
β	<	0.5	MHD stability
$(\nabla_\perp^2 / \nabla_r R_C)$	<	$0.8/\eta_{ITG}$	ITG stability
L_C/R_C	<	$190\ (\beta_p/R_C)^{1/2}(\beta_\perp/R_C)$	Rotational stability

The main constraint, in order to obtain interesting values of L_C/R_C, is the criterion for rotational stability, which requires relatively smooth potential and pressure profiles ($\beta/R_C \ll 1$) whereas profiles for the runs of Section 7 are characterized by relatively flat regions bounded by steep gradients. The pressure profile can be modified by aiming beams and injecting gas (or pellets), represented in the model by spatial dependence of N_B and n_o not yet exercised in the code.

The potential profile can be modified to accomplish rotational stability by means of bias fields, an example being the biased limiter of Ref. [11]. In principle, by segmenting the end wall into concentric rings and applying different bias voltages to each segment, the potential $\Phi(r)$ in the center cell can be modifield arbitrarily, at some cost in additional power drawn from the bias power supplies that appears as ion heating. Approximately, $\Phi(r) = V(r) + \Phi_e(r)$ where V(r) is the bias applied to the segmented end wall and $\Phi_e(r)$ is the potential as currently calculated by SYMTRAN. The bias power can be estimated from the non-ambipolar ion mobility current due to radial charge separation at ionization (neglected in SYMTRAN) of order [16]:

$$j_r = en_C(E_r/B)/(\nu_{Ci}\tau_{||}) = (S_C m_i/B^2)E_r \qquad (42)$$

from which the bias power is much less than the end loss power:

$$P_{BIAS} \sim V(j_r \pi R_C L_C) \sim P_{LOSS}\, 10(r_{Li}/R_C)^2 \sim 0.003\, P_{LOSS} \qquad (43)$$

where $P_{LOSS} = S_C \pi R_C^2 L_C \Phi_e$ is the approximate end loss (neglecting Φ_i). In estimating numbers we took $E_r = \Phi_e/R_C$, $V = \Phi_e = 10T_i$ and $R_C/r_{Li} = 60$ (consistent with $R_p/r_{Lp} = 40$ for DCLC stability). For a 250KWe reactor, the bias power would be < 1 MW.

The feasibility of potential profile control by a segmented end wall depends on the necessity to maintain sufficient plasma conduction to the wall versus engineering requirements on heat loads, secondary emission, sputtering and other materials issues. Similar considerations apply to ion sources to provide kinetic stabilization, possibly allowing a design in which electrons and guns can be co-located. The engineering issues suggest fanning the escaping field lines to dilute power loads, while conduction requirements probably require gas feeds in the end region to supplement the low density plasma escaping from the center cell of a high Q device. These issues await analysis beyond the scope of this paper.

Another important constraint is $B_C R_C > 3$ to achieve DCLC stability. This is to be compared with BR = 15 or so in ITER with H-mode confinement, thus implying the necessity for near-classical confinement in the tandem mirror. There is some margin, in that purely classical ion radial heat diffusion gives ignition at BR ≈ 0.3 over a range of ion temperatures. It is this margin whereby end losses, greater than radial losses, established the nearly flat ion temperature profile in test Run 9 that contributes to the high Q of that run. Also, it is the dominance of end losses and end plug feedback that can make the tandem mirror a thermally stable system.

Finally, given stability to DCLC, MHD, rotational and ITG modes, we must deal with the possibility of ETG transport and trapped particle modes. First tests of the consequences of ETG transport were explored in Run 10 of Section 7. Trapped particle modes in tandem mirrors were treated in Reference [17] but require more careful study for the kinetically stabilized symmetric tandem mirror.

9. Buildup

In Section 7, buildup to ignition was accomplished by plug heating, with simultaneous buildup of plug density and center cell density. This strategy leads to a minimum requirement on plug beam current, expressed through the parameter $N_B \langle \sigma v \rangle$, in order that the plugs be able to increase the electron temperature of a target plasma of initial density n_C. To calculate the minimum current, we take $n_p = (N_B \langle \sigma v \rangle) n_C \tau_{pe}$ from the steady state of Eq. (4) (for $\tau_{\alpha s} << E_B$) and substitute this into Eq. (2) with no alpha or ECH heating and neglecting the electron-ion transfer term. Then, at startup when $\tau_{||}$ ≈ exp $(\tau_i / T_i) \tau_s$ ≈ τ_s, we obtain:

$$3/2\ \partial(n_C T_e)/\partial t > [2R_M^{-1}(L_p/L_C)E_B](N_B \langle \sigma v \rangle)^2\ n_C \tau_{pe} - n_C (\tau_e + T_e)/\tau_s \qquad (44)$$

where on the right hand side L_p is the plug plasma length and E_B the beam energy, coming from P_{PLUG} in Eq. (22). Using Eq. (12), Eq. (44) says that achieving $\partial(n_C T_e)/\partial t > 0$ so that build up of T_e can proceed requires:

$$N_B\sigma v \quad > \quad 7300(n_C/L_p E_B)^{1/2} \tag{45}$$

This condition for buildup by plug heating alone is satisfied by TMX and by the runs in Section 8. Using the TMX parameters from Section 6, Eq. (45) gives $N_B\sigma v > 500$, while the actual TMX beam current gives $N_B\sigma v = 1390$ (obtained from a trapped current of 10 A, plug volume $V_p = 0.003$ and $n_p = 1.5 \times 10^{19}$ giving $N_B\sigma v = (10/en_pV_p)$). For reactor cases in Section 7, with initial $n_C = 0.02$, $L_p = R_C/$ $R_M = 1/2$, $R_C = 1.5$, $R_M = 9$ and $E_B = 1000$, Eq. (45) gives $N_B\sigma v > 46$ compared to $N_B\sigma v = 100$ for these runs. Other requirements, to insure that the electrons heat the ions, are generally less strenuous.

An alternative buildup strategy may be simultaneous plug injection and auxiliary heating of the electrons in an initial target plasma of fixed density, followed by gas feed after ignition. Constant auxiliary heating giving $n_C T_e \mu P_t$ cannot increase the electron temperature in competition with an exponentially increasing density, as we found in Section 7 in which it was necessary to delay the ECH heating eventually needed for thermal stability. However, moderate ECH heating at fixed n_C can reach ignition. The auxiliary heating required is:

$$P_{ECH} \quad > \quad (n_C/\tau_{\|})\,(\epsilon_e + T_e) \qquad \text{for buildup} \tag{46}$$

This is greatest initially when $\tau_{\|} = \exp(\epsilon_i/T_i)\,\tau_s \approx \tau_s > 0.001$ and $(\epsilon_e + T_e) < 1$ for typical reactor parameters. Then Eq. (46) gives $P_{ECH} > 50$ for a target density $n_C = 0.05$, comparable to steady state $P_{ECH} = 50$ required for thermal stability in Runs 8 and 9.

Having first ignited at low density, the density can be increased by gas fueling. Then alpha heating μn_C^2 can stay ahead of end losses even as n_C grows, given by:

$$3/2\ \partial(n_C T_e)/\partial t \approx \quad n_C^2\,\{1/4\,(1 - f_\alpha)\,\sigma v_{DT} E_\alpha - (\epsilon_e + T_e)/(n_C\tau_{\|})\} \tag{47}$$

and similarly for the ions. Ignition temperatures will be sustained during density buildup if fueling is increased slow enough to maintain the following restriction on the rate of density rise:

$$\partial n_C/\partial t \quad \ll \quad 2/3\ (n_C^2/T_e)\{1/4\ (1 - f_\square)\ \square v_{DT} E_\square\} \qquad (48)$$

or:

$$n_C(t) \quad \ll \quad n_{Co}\{1 - t\ n_{Co}\ (2/3T_e)(1/4)(1 - f_\square)\ \square v_{DT}\ E_\square\}^{-1} \qquad (49)$$

For $n_{Co} = 0.05$ and $T_e = 80$ typical of runs in Section 7, the allowed rise time is of order $\square$ 30 seconds, compared to $(n_o \square v)^{-1} \square$ 1 s to maintain reactor-level densities in steady state. Thus the gas feed must be turned up gradually over a 30s or so startup time, during which plug and auxiliary power (< 100 MW) must be supplied from the grid.

10. Experimental Tests

Many of the physics issues discussed above could be tested on neutral-beam injection experiments, such as Gamma 10 reconfigured with kinetically-stabilized symmetric end plugs. Pulsed experiments with timescales of many milliseconds would suffice.

Progress beyond earlier TMX and Gamma 10 experiments requires primarily access to electron temperatures higher than the maximum 100 eV range achieved to date in all mirror devices. The highest electron temperature one can expect in a single mirror or end plug, even with purely classical processes, is < 10% of the ion energy due to the high ambipolar potential required to confine the electrons in a mirror [18]. The tandem mirror examples of Section 7 gave $T_e \square$ 8% of $E_i \equiv E_B$ for classical end loss and radial transport. By contrast, the electron temperatures obtained in devices like 2XIIB and TMX were $\square$ 1% E_i , known to be limited by requirements of stream stabilization of ion cyclotron modes. By Eq. (35), theoretical requirements to stabilize the DCLC mode without streaming depend on the plug radius in ratio to the plug ion Larmor radius R_p/r_{Lp} [10]. In 2XIIB, with $R_p/r_{Lp} \square$ 1, DCLC stability required streams essentially filling the ambipolar hole, and similarly for TMX due to AIC modes [4]. In Ref. [19], with $R_p/r_{Lp} \square$ 7 and sloshing ions created by trapping, requirements should have been less, and indeed that experiment achieved $T_e = 100$ eV equaling 3% of $E_i = 3.6$ KeV.

Our goal will be $T_e = 1$ KeV, which we find to be attainable at low $\square$ in the plugs, while low fields in the center cell can allow sufficient $\square$ there to test kinetic stabilization. There are several reasons to push T_e to about 1 KeV if possible. First, this would greatly exceed electron temperatures in previous mirror experiments and place tandem mirrors on a footing with other approaches for which 1 KeV has marked a turning point. More

specifically, it could bring into play ETG transport in competition with classical transport as discussed in Section 9. Secondly, achieving T_e = 1 KeV would require attaining regimes of stable mirror operation at densities far above the stable regimes of Baseball II. Thirdly, since axial confinement would be very good giving little axial outflow, circular plugs could not rely on gas-dynamic stabilization but would require the kinetic stabilization that is one of the goals of the experiment.

The key to a feasible experiment is moderate density in the end plugs. The theoretical value of R_p/r_{Lp} required for DCLC stabilization by Eq. (35) depends on $\omega_{pi}^2/\omega_{ci}^2$ in the plug, the lowest occurring when $\omega_{pi}^2/\omega_{ci}^2$ is very large ($> 10^5$) or very small ($< 10^2$). We will choose $\omega_{pi}^2/\omega_{ci}^2 = 100$, giving useful plug densities at high field and $R_p/r_{Lp} = 10$ for stability [10]. With $R_p/r_{Lp} = 10$ and M = 2 for deuterium, Eq. (35) gives:

$$B_p R_p = 0.065\, E_i^{1/2} \quad \text{to achieve } T_e = 1 \text{ KeV} \qquad (50)$$

Providing fields to meet the criterion of Eq. (50) at the ion energies required to heat electrons to 1 KeV should be less costly using kinetically stabilized circular plug coils, compared to the yin-yang plugs of previous experiments. We have calculated parameters for a neutral beam experiment like Gamma 10 equipped with high field circular plugs. Figures 33 - 35 give SYMTRAN results for a run with E_B = 25 (40 KV beams) and beam source $C_{np} = 1750$ which is turned off when $n_p \approx 0.4$, after which the plug decays at constant B. In this run, n_C is held constant during the buildup of n_p, T_i and T_e. Other parameters are L_p = 0.3, L_C = 3, R_C = 0.3, R_M = 10. Electrons are only heated by the plugs ($P_{ECH} = 0$), but edge temperatues were set higher than other runs [$T_e(1) = T_i(1) = 0.035$]. The maximum core electron temperature achieved is T_e = 1.0, which occurs during the decay of the plug. A lower E_B = 13 (20 KV beams) gave T_e = 0.6.

11. Compression Experiment

As an alternative to neutral beams, recently Post and Coensgen have suggested that pulsed plug coils can heat the plug ions by adiabatic compression, along the lines of Ref. [19]. In the following, we will address requirements for a meaningful pulsed symmetric tandem mirror experiment. Again our goal will be T_e = 1 KeV.

The arrangement for a tandem mirror compression experiment would follow that of Ref. [19] for each end plug. As described in that reference, magnetic compression has produced deuterium ion energies of 3.6 KeV and T_e = 100 eV [19]. The target plasmas for compression are produced by deuterium-loaded titanium-washer-stack guns, one set for

each end plug. For the tandem mirror, there would be two sets of guns, one at each end to produce energetic ions in the end plugs starting from a target plasma produced by the guns. As described in Ref. [19], the target plasma is trapped by reflection from the inner plug mirror (partially activated during injection) and a fast gate coil activated in time to prevent escape of the returning plasma. Target plasmas with ion energy 1.3 KeV were obtained by this method. Assuming similar performance in the pulsed tandem mirror experiment, in order to obtain a final ion energy E_B we require a compression ratio C given by:

$$C \equiv B_p\,(\text{final})/B_p\,(\text{initial}) = E_B/1.3 = 0.77\,E_B \qquad \text{to achive } E_B \qquad (51)$$

where here E_B is achieved by compression.

We have written a version of our code called SYMTRAN-C to simulate a compression experiment. For simplicity, we let the center cell density n_C and the magentic compression ratio C(t) be specified, and we neglect plug ion losses, giving for the plug density:

$$n_p(t) \qquad = \qquad n_{po}\,C(T) \qquad (52)$$

The plug ion energy E_B is now a function of time given by:

$$dE_B/dt \qquad = \qquad E_B\,C^{-1}(dC/dt) \quad - E_B/\tau_p \qquad (53)$$

where the first term on the right describes compression. Note that compression requires that C(t) rise faster than $1/\tau_p$ calculated for the electron temperature of the initial target plasma.

The ion temperature in the center cell is still given by Eq. (1), while the electron temperature is given by Eq. (2) with additional terms for compression in the plugs:

$$3/2\;\partial(KT_e + n_C T_e)/\partial t \;+ (K/\tau_p + n_C/\tau_{||})(\tau_e + T_e) - r^{-1}\partial/\partial r\,(r\,n_C\,\tau_e\,\partial T_e/\partial r)$$
$$= \qquad 3/2\;KT_e\,C^{-1}(dC/dt) + P_{PLUG} + P_{ECH} \;- (n_C/\tau_e)(T_e - T_i) \qquad (54)$$

$$K \qquad = \qquad 2(n_p\,L_p/R_M\,L_C) \qquad = \qquad 2(n_{po}\,L_p/R_{Mo}\,L_C) \qquad (55)$$

where in Eq. (54) we take T_e to be the same in the plugs and center cell (close coupling) and we now include a loss term for plug electrons omitted in Eq. (2). In Eq. (55), the factor K, which arises from volume weighting for plug and center cell terms divided by the center cell

volume, is constant in time since C(t)'s cancel in $n_p = n_{po}\, C(t)$ and $R_M = R_{Mo}\, C(t)$. Here we will take:

$$C(t) = 1 + (C_{MAX} - 1)(t/\tau) \qquad (56)$$

where τ is the compression rise time.

If we set $n_C = 0$, these equations represent the single mirror compression experiment in Ref. [19]. If we neglect all loss terms (giving $E_B = E_{Bo}\, C(t)$), Eq. (54) can be integrated, giving at the peak of compression ($C = C_{MAX}$):

$$T_e = C_{MAX}\, T_{eo} \{1 + 10/3\, (E_{Bo}/T_{eo})(\tau_i/t_o)[(1 + t/\tau_i)^{1/2} - 1]\}^{2/5} \qquad (57)$$

where $\tau_i = \tau/(C_{MAX} - 1)$ and $t_o = (n_C \tau_e / n_p)_o$ is the initial drag time. For the initial parameters in Ref. [19] ($E_{Bo} = 1.3$ KeV, $n_{po} = 0.06$, and $T_{eo} = 0.007$) and $C_{MAX} = 3.5$ and $\tau = 5 \times 10^{-5}$ for this experiment, we obtain $t_o = 10^{-4}$ and $T_e = 0.16$ compared with $T_e = 0.1$ measured by microwave noise. This electron temperature is 6 times that due to compression alone, due to rapid heating while electron temperatures are low, even though at the final temperatures the timescale for electron heating by the plugs is much longer than the plug duration.

Figure 36 gives results from SYMTRAN-C for the single mirror compression experiment including the loss terms, for the same initial conditions and $n_C = 10^{-9} \approx 0$ to approximate a single mirror. Also, we do not solve Eq. (7) but instead take $\tau_e = 5T_e$ appropriate for a single mirror [18]. Now $T_e = 0.15$ at the end of compression, slightly below the lossless case and still in fair agreement with the experiment.

Having found satisfactory agreement with experimental results, we now use SYMTRAN-C to calculate parameters for a compression tandem mirror experiment with the goal of $T_e \geq 1$ KeV. Figure 37 shows results for the same initial conditions in the plugs but $C_{MAX} = 15$ and $\tau = 10^{-3}$, with $n_C = 0.025$ and $T_{eo} = T_{io} = 0.007$. Agaain we approximate Eq. (7) by $\tau_e = 5T_e$ (plug dominated). For these parameters, the plug compression formula, Eq. (57), would satisfy $T_e \geq 1$ KeV. However, with density $n_C = 0.025$ in the center cell, the SYMTRAN-C gives only $T_e = 0.5$ KeV.

The reason for lower electron temperatures in the tandem mirror case is the fact that only the plug plasma is compressed, while collisions distribute the heat of compression from the plugs into the center cell, thereby adding to the heat capacity. To increase the electron temperature, we apply pulsed auxiliary electron heating in the center cell at the

same time that compression in the plugs is building the potential barriers that improve the confinement time in the center cell.

Figure 38 shows a SYMTRAN-C tandem mirror result that achives the goal of $T_e \geq$ 1 KeV by applying pulsed $P_{ECH} = 50$ (in code units) in the center cell during the compression cycle. In MKS units, this corresponds to 300 kW for 1 millisecond (300 joules). Note that high temperatures persist for several milliseconds, giving time for transport measurements in the center cell (the magnetic field is assumed constant during this time). The ion temperature (not shown) remains essentially at its initial value in the target plasma (taken as 0.007 KeV). With 1 KeV electrons, the ions are not heated significantly, the transfer time being $\tau_{ie} = 400$ ms.

The following are example parameters for a compression experiment based on these results:

Device parameters

$L_C = 3$	$R_C = 0.2$	$B_C \geq 0.2$	(adjustable)
$L_p = 0.3$	$R_p = 0.08$	$B_p = 0.67$	(at the onset of compression)
	$R_p = 0.02$	$B_p = 10$	(after compression: C = 15)

Initial target plasma (similar to Ref. [17])

$n_p = 0.06$	$E_B = 1.3$	$T_e = 0.007$

Plasma after compression and pulsed ECH

$n_p = 0.9$	$E_B = 20$	$T_i = 0.007$	$\tau_p = 5$ ms
$n_C = 0.025$	$T_e = 1$	$\tau_i = 3.6$	

With these parameters, end loss from the center cell is negligible, giving opportunity to observe radial transport during the > 10 ms lifetime of the plugs (or the magnetic field). The DCLC stability parameters are $(\omega_{pi}^2/\omega_{ci}^2)_{PLUG} = 170$ for hydrogen and $R_p / r_{Li} = 10$, giving stability with no streaming [18].

To estimate the plug coil energy, we take the plug mirror field to be $1.5\ B_p = 15$ chosen to reduce plug loss by ion scattering as T_e rises. We take as the coil inner radius R = 0.15 and we take the length L = 0.4. Then the stored energy E_{COIL} is, crudely:

$$E_{COIL} \approx (0.4)(0.15)^2[(15)^2/2\mu_o] \approx 2.5 \text{ MJ} \qquad (58)$$

We conclude that a power source yielding a few MJ with a millisecond rise time could drive a pulsed, compression-heated symmetric tandem mirror experiment capable of exploring the main issues for the concept at electron temperatures of 1 KeV, unprecedented in previous mirror experiments.

12. Summary

The time-dependent SYMTRAN code is now ready for Q optimization studies, with various physical knobs discussed in the text. Steady state can be achieved or not, depending on several physical effects included in the code that can cause macroscopic, or thermal, instability of the system, as discussed in Sections 4 and 5. Thus far we have only found stable steady states with constant auxiliary power that reduces Q.

Despite this need for auxiliary power to provide thermal stability to the system, test runs to date show that Q = 10 or more can probably be achieved with classical radial diffusion, similar to Q values shown to produce interesting steady state reactor parameters in Ref. [1] where radial diffusion was neglected. An example is Run 9 (Figures 28 - 30) which represents our most realistic attempt to date (when scaled to densities appropriate for steady state, as discussed in Section 7). Strategies to achieve ignition at useful densities using only the equipment needed to sustain the steady state are discussed in Section 9.

The remaining tasks are (1) to obtain results with SYMTRAN for an overall optimal system, and (2) to do so consistent with plasma stability at the MHD and microscopic levels as discussed in Section 8. In first tests reported in Section 7, some anomalous ETG electron heat transport in addition to classical transport did not appear to be disastrous.

A key physics issue is the need for high electron temperature in order to achieve high Q without recourse to thermal barriers, which in turn requires DCLC stability of the plugs without recourse to stream stabilization. New experiments to test this point and kinetic stabilization are suggested in Sections 10 and 11. A corollary to the requirement for DCLC stability is high magnetic field in the end plugs, now feasible with circular plug coils and kinetic stabilization [7].

References

[1] D. D. Hua and T. K. Fowler, “Calculations of Radial Transport in Symmetric Tandem Mirrors,” Department of Nuclear Engineering, University of California, Berkeley, September 30, 2003.

[2] R. H. Cohen, Nuclear Fusion **19**, 1295 (1979).

[3] F. H. Coensgen et al. Phys. Rev. Letters **44**, 1132 (1980).

[4] T. C. Simonen et al., "Plasma Confinement Experiments in the TMX Tandem Mirror," Brussels IAEA, 1980, paper IAEA-CN-38/F-1.

[5] A. A. Mirin, S. P. Auerbach, R. H. Cohen, J, M. Gilmore, L. D. Pearlstein and M. E. Rensink, Nuclear Fusion **23**, 703 (1983).

[6] R. W. Conn, "Magnetic Fusion Reactors," in *Fusion*, E. Teller, Editor, Academic Press, New York, 1981, Chapter 15, Vol. 1, Part B, Section V.

[7] R. F. Post, Trans. of Fusion Technology **39**, 25 (2001); ongoing MHD studies by J. Byers et al., private communication.

[8] T. C. Simonen et al., "TMX Tandem Mirror Experiments and Thermal-Barrier Theoretical Studies," Baltimore IAEA, 1982, paper IAEA-CN-41/G-1.

[9] T. G. Rognlien and T. A. Cutler, Magnetic Fusion Energy Quarterly Report October-December 1978, Lawrence Livermore Laboratory, Calif., UCRL-50051-78-4 (1978), p. 4.

[10] D. E. Baldwin, Rev. Mod. Phys. **49**, 317 (1977).

[11] O. Sakai, Y. Yasaka and R. Itami, Phys. Rev. Letters **70**, 4071 (1993).

[12] W. Horton, "Electron Transport and the Critical Temperature Gradient," Review Talk F11 4, DPP/APS Albuquerque Oct. 2003.

[13] G. D. Porter and the DIIID Team, Phys. Plasmas **5**, 4311 (1998).

[14] T. D. Rognlien and Y. Matsuda, Nucl. Fusion **21**, 345 (1981).

[15] R. H. Cohen, private communication March, 2004.

[16] A. J. Lichtenberg and M. A. Lieberman, *Principles of Plasma Discharges and Materials Processing*, Interscience, New York, 1994, Chapter 2, Eq. (2.3.14).

[17] H. L. Berk, M. N. Rosenbluth, R. H. Cohen and W. M. Nevins, Phys. Fluids **28**, 2824 (1985).

[18] T. K. Fowler, "Mirror Theory," in *Fusion*, E. Teller, Editor, Academic Press, New York, 1981, Chapter 5.

[19] F. H. Coensgen, W. F. Cummins, W. E. Nexen, Jr. and A. E. Sherman, Nuclear Fusion: 1962 Supplement, Part 1, 125 - 133 (1962). [19] SPHERE -- "A SPHeromak Experiment and Reactor Simulation Code," Department of Nuclear Engineering, University of California, Berkeley, UC-BFE-054. March 1, 2000.

[20] D. D. Hua, T. K. Fowler and E. C. Morse, "SPHERE – A SPHeromak Experiment and Reactor Simulation Code,"Department of Nuclear Engineering, University of California, Berkeley, UC-BFE-054, March 1, 2000.

Appendix A: Thermal Stability in the Tandem Mirror Core

Linearization of Eqs. (1) - (4) without the diffusion terms gives:

$$3/2\, T_{i1}\,[p + n_C^{-1}\partial/\partial T_i\,(L_i - P_i)] = - (n_{C1}/n_C)T_i\, p + n_C^{-1}\sum X_1\, \partial/\partial X\,(P_i - L_i) \quad \text{(A1)}$$

$$3/2 T_{e1}\,[p + n_C^{-1}\partial/\partial T_e\,(L_e - P_e)] = - (n_{C1}/n_C)T_e\, p + n_C^{-1}\sum X_1\, \partial/\partial X\,(P_e - L_e) \quad \text{(A2)}$$

$$n_{C1}[p + \partial/\partial n_C\,(n_C/\tau_{\|} - S_C)] = \sum X_1\, \partial/\partial X\,(S_C - n_C/\tau_{\|}) \quad \text{(A3)}$$

$$n_{p1}[p + \partial/\partial n_p(n_p/\tau_p - S_p)] = \sum X_1\, \partial/\partial X\,(S_p - n_p/\tau_p) \quad \text{(A4)}$$

where all perturbation quantities vary in time as exp pt. The input powers P_i and P_e were defined in Eqs. (1) and (2) and for later convenience the energy end loss terms have now been labeled L_i and L_e defined by:

$$L_i = n_C(\tau_i + T_i)/\tau_{\|} \quad \text{(A5)}$$

$$L_e = n_C(\tau_e + T_e)/\tau_{\|} \quad \text{(A6)}$$

The left hand sides of Eqs. (A1) - (A4) give the response of each variable to its own sources and end losses. The first term on the right hand sides of Eqs. (A1) and (A2) gives the density variation of kinetic energy. The summations on the right hand sides of Eqs. (A1) - (A4) give the other coupling terms with sums $\sum$ running over the three perturbations (labeled X_1) not appearing on the left hand side.

The end loss derivatives, unique to tandem mirrors, are given as follows. With the hot core of reactors in mind, we neglect the τ_s term in Eq. (11), giving:

$$\partial/\partial n_C\,(n_C/\tau_{\|}) = + (1/\tau_{\|})\{2 + (T_e/T_i)[(\tau_i/T_i + 1)/\tau_i/T_i]\} \quad \text{(A7)}$$

$$\partial/\partial T_i\,(n_C/\tau_{\|}) = + (n_C/\tau_{\|})(1/T_i)(\tau_i/T_i - 1/2)\} \quad \text{(A8)}$$

$$\partial/\partial T_e\,(n_C/\tau_{\|}) = - (n_C/\tau_{\|})(1/T_e)(\tau_i/T_i + 1)\} \quad \text{(A9)}$$

$$\partial/\partial n_p\ (n_C/\tau_{\|}) \;=\; -(1/\tau_{\|})(T_e/T_i)[(\tau_i/T_i + 1)/\tau_i/T_i]/(n_p/n_C)\} \qquad \text{(A10)}$$

A. Sufficient Condition for Thermal Instability in the Core

The time constant p is found by eliminating the four perturbation quantities (with subscript 1) from the four equations for p, Eqs. (A1) - (A4) (by inverting a 4x4 matrix). The resulting dispersion equation has the form:

$$p^4 + A_3 p^3 + A_2 p^2 + A_1 p + A_0 = 0 \qquad \text{(A11)}$$

Finding the p's exactly with all the cross couplings is complicated. However, we can learn something by examining only the signs of the coefficients. The coefficient A_3 in the cubic term is the negative sum of the roots a_i ($A_3 = -\sum a_i$, i = 1 to 4). Stability requires $A_3 > 0$ ($\sum a_i < 0$) since otherwise some root must have a positive real part. Similarly, A_2 is a sum of quadratic products of roots ($A_2 = \sum a_i a_j$ all $i < j$ (six terms). Here A_2 is real. Then any complex roots occur as conjugate pairs so that any negative product in A_2 is a product of real parts of the roots. Stability requires $A_2 > 0$ since otherwise some product of real parts is negative indicating that at least one root has a positive real part. Similar rules apply to A_1 and A_0, which are cubic and quartic in the roots. We will restrict our attention to A_3 and A_2. Since both must A_3 and A_2 must be positive for stability, a sufficient condition for instability is:

$$A_3 \text{ or } A_2 < 0 \qquad \text{Sufficient condition for Thermal Instability} \qquad \text{(A12)}$$

Satisfying Eq. (A12) guarantees thermal instability. Violating Eq. (A12) leaves open the possibility of stability, but even then stability is not assured.

To evaluate A_3 and A_2, let us denote derivative terms in Eqs. (A1) - (A4) by A_{jk} equal to the X_k derivative of $(L - P)_j$ for heat terms and the equivalent loss minus source for particles as indicated in Eqs. (A3) and (A4). The subscripts are labeled i for T_i ; e for T_e ; c for n_C ; and p for n_p . With this notation, A_3 and A_2 are given by:

$$A_3 = \{A_{ii} + A_{ee} + A_{pp} + A_{CC}\} - \{A_{ci}(T_i/n_C) + A_{ce}(T_e/n_C)\} \qquad \text{(A13)}$$

$$A_2 = \{A_{ii} A_{CC} + (A_{ee} + A_{pp})(A_{ii} + A_{CC}) + (A_{ee} A_{pp} - A_{ep} A_{pe}\}$$
$$- 4/9\, A_{ie} A_{ei} - 2/3\, A_{ci} A_{ic} - 2/3\, A_{ce} A_{ec} + 2/3\, A_{ei} A_{ce} (T_i/n_C)$$

$$+ 2/3\ A_{ci}\, A_{ie}\, (T_e/n_C) - A_{ci}\, (A_{ee} + A_{pp})(T_i/n_C) - A_{ce}\, (A_{ii} + A_{pp})(T_e/n_C) \quad \text{(A14)}$$

Let us now evaluate A_3. The second {...} bracket coming from the first terms on the right in Eqs. (A1) and (A2) is stabilizing but relatively small. Using end loss derivatives from Eqs. (A8) and (A9), we obtain:

$$-\{ A_{ci}\, (T_i/n_C) + A_{ce}(T_e/n_C)\} \quad = \quad 3/2\ (1/\tau_{\|}) \quad \text{(A15)}$$

Substituting for this and the other A_{jk} 's gives for Eq. (A13):

$$A_3 = \{ n_C^{-1}[\partial/\partial T_i\ (n_C(\tau_i + T_i)/\tau_{\|}) - \partial/\partial T_i\ P_i] + n_C^{-1}[\partial/\partial T_e\ (n_C(\tau_e + T_e)/\tau_{\|}) - \partial/\partial T_e\ P_e]\}_{TEMP}$$
$$+ \{3/2\ (1/\tau_{\|}) - \partial/\partial n_C\ (S_C - n_C/\tau_{\|}) - \partial/\partial n_C\, S_C - \partial/\partial n_p(S_p - n_p/\tau_p)\}_{DENSITY} < 0 \quad \text{(A16)}$$

where we have separated out thermal terms coming from Eqs. (A1) and (A2) from center cell and plug density terms coming from Eqs. (A3), (A4) and (A15).

We can reduce Eq. (A16) to a more explicit expression using Eqs. (A7) - (A10) and other derivatives given exactly or approximately as follows:

$$\partial/\partial T_i\ (\tau_i + T_i) \quad \approx \quad 1 \quad \text{(A17)}$$

$$\partial/\partial T_e\ (\tau_e + T_e) \quad \approx \quad (\tau_e/T_e + 1) \quad \text{(A18)}$$

$$\partial/\partial T_i\ P_{ie} \quad \equiv \quad \partial/\partial T_i\ n_C(T_e - T_i)/\tau_e \quad = \quad - n_C\ (1/\tau_e) \quad \text{(A19)}$$

$$\partial/\partial T_e\ P_{ie} \quad = \quad (1/\tau_e)\ (3/2\ T_i/T_e - 1/2) \quad \text{(A20)}$$

$$\partial/\partial T_e\ P^* \quad = \quad - (1/\tau_e)\ 3/2\ P^*/T_e \quad \text{(A21)}$$

$$\partial/\partial n_C\, S_C \quad = \quad + (1/\tau_{\|}) \quad \text{(A22)}$$

$$\partial/\partial n_p(S_p - n_p/\tau_p) \quad = \quad - (1/\tau_p) \quad \text{(A23)}$$

In Eqs. (A22) and (A23) we use steady state results from Eqs. (3) and (4).

Finally, Eq. (A16) becomes:

$$A_3 = \{ (1/\tau_{\parallel}) [- (\tau_e/T_e + 1)(\tau_i/T_i) + (\tau_i/T_i + 1)(\tau_i/T_i - 1/2) + 1]\}_{THERMAL}$$
$$+ \{ (1/\tau_{\parallel}) [1/2 + (T_e/T_i)((\tau_i/T_i + 1)/\tau_i/T_i)] + (1/\tau_p)\}_{DENSITY}$$
$$+ \{(1/\tau_e)[T_e/T_i + 3/2\, T_i/T_e]\}_{IE} + \{3/2\, (P^*/n_C T_e)\}_{PLUG}$$
$$+ \text{alphas} + \text{aux.}$$
$$< \quad 0 \qquad \text{(A24)}$$

Negative terms are destabilizing while positive terms, which can cause this sufficient condition to fail, would leave open the possibility of a stable system. The effect of heat end losses (labeled Thermal), unique to tandem mirrors, is usually destabilizing while the density terms -- both center cell and plug -- are stabilizing. The electron-ion heat exchange term (labeled IE) and also the plug heating P* are stabilizing. Auxiliary heating effects are case-specific; constant auxiliary power (no derivatives) would only affect stability through its influence on the steady state.

The alpha terms in A_3 are given by:

$$\text{alphas} = - (f_i/n_C)\, \partial P_\alpha/\partial T_i + (P_\alpha/n_C)\, \partial f_i/\partial T_e \qquad \text{(A26)}$$

$$\partial f_i/\partial T_e \approx 3/2\, (1/T_e)\, f_i\, (1 - f_i) \qquad \text{(A27)}$$

where Eq. (A27) follows from Eq. (9) if we drop temperatures compared to $\langle E_\alpha \rangle$. The interesting burn regime occurs around an ion temperature of 15 KeV where power density is maximum for a given beta [4]. In this regime, $(T_i/P_\alpha)\, \partial P_\alpha/\partial T_i \approx 1$, giving:

$$\text{alphas (in } A_3) = - (P_\alpha/n_C T_e)\, f_i\, [T_e/T_i - 3/2(1 - f_i)] \qquad \text{burn regime} \qquad \text{(A28)}$$

Thus alpha heating is destabilizing.

Combining Eq. (A28) with the other heating terms in Eq. (A24) gives:

$$A_{HEATING} = \{(1/\tau_e)[T_e/T_i + 3/2\, T_i/T_e]\}_{IE} + \{3/2\, (P^*/n_C T_e)\}_{PLUG}$$
$$- (P_\alpha/n_C T_e)\, f_i\, [T_e/T_i - 3/2(1 - f_i)] \qquad \text{(A29)}$$

Keeping only dominant terms in the rest of Eq. (A24) gives the main effects of end losses:

$$A_{LOSSES} \approx - (1/\tau_{\parallel})(\tau_e/T_e - \tau_i/T_i)(\tau_i/T_i + 1) + (1/\tau_p) \qquad \text{(A30)}$$

In some experiments the heating term may be important but in reactors it appears that the IE term is small while the alpha destabilization roughly offsets the benefit of plug heating stabilization at high Q, where $P^* \approx (5/Q)\, P_{\alpha}$.

The main destabilizing contribution in A is due to electron end losses $\propto \nu_e$ in Eq. (A30) while the main stabilizing effect is the plug term $\propto 1/\tau_p$, requiring as the approximate condition for stability (only necessary):

$$\tau_p \quad < \quad [(\nu_e/T_e - \nu_i/T_i)(\nu_i/T_i + 1)]^{-1}\, \nu_{\|} \qquad \text{(A31)}$$

Even if Eq. (A31) is satisfied, stability requires also $B > 0$ (again necessary, not sufficient). Dominant terms appear to be:

$$A_2 \approx (A_{ee} A_{ii} - 4/9\, A_{ie} A_{ei}) + A_{pp} A_{ii} + (A_{ee} A_{pp} - A_{ep} A_{pe})$$
$$\approx (1/\tau_p \nu_{\|})\{(\nu_i/T_i)^2 [1 - 5/9\, (\nu_i/T_i)(\nu_e/T_e)(\nu_p/\nu_{\|})] - (\nu_e/T_e)((\nu_i/T_i) - 3/2(T_e/T_i)\} \qquad \text{(A32)}$$

Again stability requires sufficiently small $\nu_p/\nu_{\|}$.

APPENDIX B

1. Introduction

The symtran codes, **symtran2.f**, **symtran4.f** and **symtran.c.f** are adapted from the SPHERE code originally written to simulate spheromaks [20]. The code **symtran2.f** solves Eqs. (1) and (2) with n_c set constant (temporally) and n_p specified as a function of time (but uniform spatially), while the code **symtran4.f** solves Eqs. (1) - (4). Finally, the code **symtran.c.f** solves Eqs. (1), and (52)-(54). All three codes use the parabolic partial differential equation solver **D03PCF**, while **symtran2.f** and **symtran4.f** also use the root solver **C05AGF** (for ϕ_e in Eq. (7)) from the Numerical Algorithm Group (NAG). Detailed informations on both routines are available at NAG's website: **www.nag.com** in the fortran 77 numerical library.

To compile the codes and link them with the NAG library, type

f77 symtran2.f (or **symtran4.f**, or **symtran.c.f**) **-dalign -lnagd**

which produces an executable file **a.out**. Create (output) file names **stat.dat**, **U.dat**, **U0.dat**, **Phi.dat**, **Power.dat** and **Endloss.x.dat** (only need to create these output files just once, not necessary to do so everytime), then type **a.out** to run.

For **symtran4.f**, to conform to specifications in **D03PCF**, Eqs. (1) to (4) are expressed in the form:

$$P(1,1)\frac{\partial T_i(\rho,t)}{\partial t} + P(1,3)\frac{\partial n_c(\rho,t)}{\partial t} + Q(1) = \frac{1}{\rho}\frac{\partial}{\partial \rho}\left[\rho R(1)\right] \quad \text{(B1)}$$

$$P(2,2)\frac{\partial T_e(\rho,t)}{\partial t} + P(2,3)\frac{\partial n_c(\rho,t)}{\partial t} + Q(2) = \frac{1}{\rho}\frac{\partial}{\partial \rho}\left[\rho R(2)\right] \quad \text{(B2)}$$

$$P(3,3)\frac{\partial n_c(\rho,t)}{\partial t} + Q(3) = \frac{1}{\rho}\frac{\partial}{\partial \rho}\left[\rho R(3)\right] \quad \text{(B3)}$$

$$P(4,4)\frac{\partial n_p(\rho,t)}{\partial t} + Q(4) = \frac{1}{\rho}\frac{\partial}{\partial \rho}\left[\rho R(4)\right] \quad \text{(B4)}$$

where the radial coordinate $\rho = (r/a)$ is normalized with respect to the minor radius a. The various coefficients in the PDE's are:

$$P(1,1) = (3/2)n_c(\rho,t) \quad \text{(B5)}$$

$$
\begin{aligned}
P(1,3) &= (3/2)T_i(\rho,t) && \text{(B6)}\\
Q(1) &= -\left[P_\alpha f_\alpha + P_{ion} + \frac{n_c}{\tau_{ie}}\left(T_e - T_i\right) - \frac{n_c}{\tau_{\|}}\left(\phi_i + T_i\right)\right] && \text{(B7)}\\
R(1) &= \frac{n_c}{a^2}\,\chi_i \frac{\partial T_i}{\partial \rho} && \text{(B8)}\\
P(2,2) &= (3/2)n_c(\rho,t) && \text{(B9)}\\
P(2,3) &= (3/2)T_e(\rho,t) && \text{(B10)}\\
Q(2) &= -\Bigg[P_\alpha\left(1-f_\alpha\right) + P_{ECH} + P_{PLUG} - \frac{n_c}{\tau_{ie}}\left(T_e - T_i\right) - P_{Brem}\\
&\qquad -\frac{n_c}{\tau_{\|}}\left(\phi_e + T_e\right)\Bigg] && \text{(B11)}\\
R(2) &= \frac{n_c}{a^2}\,\chi_e \frac{\partial T_e}{\partial \rho} && \text{(B12)}\\
P(3,3) &= 1 && \text{(B13)}\\
Q(3) &= -\left[S_N - \frac{n_c}{\tau_{\|}}\right] && \text{(B14)}\\
R(3) &= D\frac{\partial n_c}{\partial \rho} && \text{(B15)}\\
P(4,4) &= 1 && \text{(B16)}\\
Q(4) &= -\left[S_P - \frac{n_p}{\tau_p}\right] && \text{(B17)}\\
R(4) &= 0\,. && \text{(B18)}
\end{aligned}
$$

The substitutions $T_i(\rho,t) \rightarrow U(1)$, $T_e(\rho,t) \rightarrow U(2)$, $n_c(\rho,t) \rightarrow U(3)$ and $n_p(\rho,t) \rightarrow U(4)$ are also made to conform to specifications in **D03PCF**. All other relevant quantities are defined previously in Eqs. (5) - (23).

In **symtran2.f**, *Eq. (4) is not used, whereas in* **symtran.c.f**, *Eq. (4) is replaced by Eq. (53) (with appropriately defined* $P(4,4)$, $Q(4)$ *and* $R(4)$ *to match Eq. (53), and the substitution* $E_B \rightarrow U(4)$*). To keep* $n_c(\rho,t)$ *constant in time in both* **symtran2.f** *and* **symtran.c.f**, *let* $P(3,3) = 1$, $Q(3) = 0$ *and* $R(3) = 0$.

The boundary conditions in **symtran4.f** are: at $\rho = (r/a) = 0$, $(\partial T_i/\partial\rho) = (\partial T_e/\partial\rho) = (\partial n_c/\partial\rho) = (\partial n_p/\partial\rho) = 0$; and at $\rho = (r/a) = 1$, $T_i = T_i|_{\rho=1}$, $T_e = T_e|_{\rho=1}$, $n_c = n_c|_{\rho=1}$ and $n_p = n_p|_{\rho=1}$. The boundary conditions in **symtran2.f** and **symtran.c.f** are similarly defined, and hence not shown explicitly. The boundary conditions are expressed in the form:

$$\beta(1)R(1) = \gamma(1) \tag{B19}$$
$$\beta(2)R(2) = \gamma(2) \tag{B20}$$
$$\beta(3)R(3) = \gamma(3) \tag{B21}$$
$$\beta(4)R(4) = \gamma(4) \tag{B22}$$

where $R(1)$ - $R(4)$ are defined in Eqs. (B8), (B12), (B15) and (B18) respectively. $\gamma(i)$'s are defined in Table 1.

The initial conditions are written as

$$T_i(\rho, t=0) = \left[T_i(\rho=0, t=0) - T_i|_{\rho=1}\right]\left(1-\rho^2\right) + T_i|_{\rho=1} \tag{B23}$$
$$T_e(\rho, t=0) = \left[T_e(\rho=0, t=0) - T_e|_{\rho=1}\right]\left(1-\rho^2\right) + T_e|_{\rho=1} \tag{B24}$$
$$n_c(\rho, t=0) = \left[n_c(\rho=0, t=0) - n_c|_{\rho=1}\right]\left(1-\rho^2\right) + n_c|_{\rho=1} \tag{B25}$$
$$n_p(\rho, t=0) = \left[n_p(\rho=0, t=0) - n_p|_{\rho=1}\right]\left(1-\rho^2\right) + n_p|_{\rho=1} \tag{B26}$$
$$E_B(\rho, t=0) = \left[E_B(\rho=0, t=0) - E_B|_{\rho=1}\right]\left(1-\rho^2\right) + n_p|_{\rho=1} \tag{B27}$$

Hence, either flat or parabolic profiles could be imposed by choosing whether $F(\rho=0, t=0)$ equal $F|_{\rho=1}$.

Table 1: Coefficients for boundary conditions

	$\rho = 0$	$\rho = 1$
$\beta(1)$	1	0
$\gamma(1)$	0	$U(1) - T_i\|_{\rho=1}$
$\beta(2)$	1	0
$\gamma(1)$	0	$U(2) - T_e\|_{\rho=1}$
$\beta(3)$	1	0
$\gamma(3)$	0	$U(3) - n_c\|_{\rho=1}$
$\beta(4)^\dagger$	1	0
$\gamma(4)^\dagger$	0	$U(4) - n_p\|_{\rho=1}$

$^\dagger\beta(4)$ and $\gamma(4)$ in tables are for **symtran4.f**, but they are not use in **symtran2.f**. In **symtran.c.f**, $\gamma(4) = U(4) - C(t)E_B\|_{\rho=1}$ at $\rho = 1$.

2. Tables of parameters and output files

Table 2: Numerical Parameters for **symtran2.f symtran4.f** and **symtran.c.f**

Input	Symbol in symtran	Typical Values
starting time	TS	0
time step	DELTAT	10^{-5}
number of time steps	NSTEP	10^6
number of spatial profiles output†	NWRITE	20
frequency of output for core values‡	NCORE	1000
accuracy*	ACC	1.0×10^{-4}

†Spatial profiles are printed every (NSTEP/NWRITE) time-steps (e.g. in Tables 4, 6 and 8, t_j = N $\cdot$ [(NSTEP/NWRITE) $\cdot$ DELTAT]), for $N = 1, 2, \ldots,$ NWRITE).

‡Core values (at $(r/R) = 0$) are printed every NCORE time-steps.

*There is $\sim 5\%$ difference between solutions with $ACC = 1.0 \times 10^{-4}$ and solutions with $ACC = 1.0 \times 10^{-7}$ in **symtran2.f** and **symtran4.f**. However, it requires longer time to find solutions with a smaller ACC. For the case in which an analytic solution exist for **symtran.c.f**, an $ACC = 1.0 \times 10^{-7}$ is required so that the numerical differs from the analytic solution by $< 1\%$. Hence $ACC = 1.0 \times 10^{-7}$ is chosen for all runs using **symtran.c.f**. For more details on ACC, consult the specifications of **D03PCF**, available at **www.nag.com**'s fortran 77 numerical library.

Table 3: Physical Input Parameters for Symtran2.f

Input	Symbol in Symtran2.f	Typical Values
$T_i(\rho = 1, t)$	Ti_EDGE	0.001
$T_e(\rho = 1, t)$	Te_EDGE	0.001
$n_c(\rho = 1, t)$	den_EDGE	1.0
$T_i(\rho = 0, t = 0)$	Ti_O	0.002
$T_e(\rho = 0, t = 0)$	Te_O	0.002
$n_c(\rho = 0, t = 0)$	den_O	1.0
(m_i/m_e)	MR	4583
n_{p0}	N_P	10
-	N_P_C[†]	1, 2, or 3
E_α	E_ALPHA	3520
$\langle E_\alpha \rangle$	E_ALPHA_AV	2000
Z (in p_{Brem})	Z_EFF	0
mirror Ratio	R_M	9.2
minor radius	RA	1.5
reactor length	ZL	30
beam energy	EB	1000
magnetic field	BMAG	1.0
electron ECH power[‡]	P_ECH	1000 (i), 70 (f)
$T_e(\rho = 0, t)$ to reduce ECH	Te_crit	40
plug radius (normalized)	X_PLUG	0.98 or > 1
C_{ETG}	C_ETG	0

[†]In **symtran2.f**, choose N_P_C = 1 for $n_p = n_{p0}$, N_P_C = 2 for $n_p = n_{p0}(1 - 7T_e/EB)$ and N_P_C = 3 $n_p = n_{p0}(1 - 7T_e/EB)[1 - \rho^2]$. [‡]In **symgran2.f**, AS is the ratio of P_ECH input initially, and AS1 is the ratio of P_ECH input after core electron temperature has reached Te_crit.

Table 4: Example Output Files for symtran2.f

File Name	Data Columns				
U0.dat	t	$T_i(\rho=0,t)$	$T_e(\rho=0,t)$	$n_c(\rho=0,t)$	P_{ECH}
U.dat	ρ	$T_i(\rho,t_j)$	$T_e(\rho,t_j)$	$n_c(\rho,t_j)$	n_p
Phi.dat	ρ	$\phi_i(\rho,t_j)$	$\phi_e(\rho,t_j)$	error ratio†	-
Power.dat	t	$\int_0^1 d\rho\ \rho P_{input}(\rho,t_j)$	$\int_0^1 d\rho\ \rho[5P_\alpha(\rho,t_j)]$	Q	P_{ECH}
Endloss.x.dat	ρ	$[n_c(\phi_i+T_i)/\tau_\|\|]_{t=t_j}$	$[n_c(\phi_i+T_i)/\tau_\|\|]_{t=t_j}$		-

†Take $V1=(\phi_e/T_e)\exp[\phi_e/T_e]$, and $V2=(m_i/m_2)^{\frac{1}{2}}[T_i/T_e]^{\frac{3}{2}}\{(\phi_i/T_i)\exp[\phi_i/T_i]\}$, then error ratio $=\mid[\log V2-\log V1]/\log V1\mid$.

Table 5: Physical Input Parameters for Symtran4.f

Input	Symbol in Symtran4.f	Typical Values
$T_i(\rho = 1, t)$	Ti_EDGE	0.001
$T_e(\rho = 1, t)$	Te_EDGE	0.001
$n_c(\rho = 1, t)$	den_EDGE	0.010
$n_p(\rho = 1, t)$	N_P_EDGE	0.012
$T_i(\rho = 0, t = 0)$	Ti_O	0.002
$T_e(\rho = 0, t = 0)$	Te_O	0.002
$n_c(\rho = 0, t = 0)$	den_O	0.020
$n_p(\rho = 0, t = 0)$	N_P_O	0.024
(m_i/m_e)	MR	4583
E_α	E_ALPHA	3520
$\langle E_\alpha \rangle$	E_ALPHA_AV	2000
Z (in p_{Brem})	Z_EFF	0
mirror Ratio	R_M	20
minor radius	RA	1.5
reactor length	ZL	100
beam energy	EB	1000
magnetic field	BMAG	0.5
electron ECH power[†]	P_ECH	0 (i), 2.0×10^4 (f)
$T_e(\rho = 0, t)$ to turn on ECH	Te_crit	60
C_{ETG}	C_ETG	0
S_N	S_N	5.0
S_P	C_1_N_P	100

[†]In **symgran4.f**, AS is the ratio of P_ECH input initially, and AS1 is the ratio of P_ECH input after core electron temperature has reached Te_crit.

Table 6: Example Output Files for symtran4.f

File Name	Data Columns				
U0.dat	t	$T_i(\rho=0,t)$	$T_e(\rho=0,t)$	$n_c(\rho=0,t)$	$n_p(\rho=0,t)$
U.dat	ρ	$T_i(\rho,t_j)$	$T_e(\rho,t_j)$	$n_c(\rho,t_j)$	$n_p(\rho,t_j)$
Phi.dat	ρ	$\phi_i(\rho,t_j)$	$\phi_e(\rho,t_j)$	error ratio†	-
Power.dat	t	$\int_0^1 d\rho\ \rho P_{input}(\rho,t_j)$	$\int_0^1 d\rho\ \rho[5P_\alpha(\rho,t_j)]$	Q	P_{ECH}
Endloss.x.dat	ρ	$[n_c(\phi_i+T_i)/\tau_{\|\|}]_{t=t_j}$	$[n_c(\phi_i+T_i)/\tau_{\|\|}]_{t=t_j}$		-

†Take $V1=(\phi_e/T_e)\exp[\phi_e/T_e]$, and $V2=(m_i/m_2)^{\frac{1}{2}}[T_i/T_e]^{\frac{3}{2}}\{(\phi_i/T_i)\exp[\phi_i/T_i]\}$, then error ratio $=\mid[\log V2-\log V1]/\log V1\mid$.

Table 7: Physical Input Parameters for Symtran.c.f

Input	Symbol in Symtran.c.f	Typical Values
$T_i(\rho = 1, t)$	Ti_EDGE	0.007
$T_e(\rho = 1, t)$	Te_EDGE	0.007
$n_c(\rho = 1, t)$	den_EDGE	(i) 0.0 or (ii) 0.025
$E_B(\rho = 1, t)$	EB_EDGE	1.3
$T_i(\rho = 0, t = 0)$	Ti_O	0.007
$T_e(\rho = 0, t = 0)$	Te_O	0.007
$n_c(\rho = 0, t = 0)$	den_O	(i) 0.0 or (ii) 0.025
$E_B(\rho = 0, t = 0)$	EB_O	1.3
E_α	E_ALPHA	3520
$\langle E_\alpha \rangle$	E_ALPHA_AV	2000
Z (in p_{Brem})	Z_EFF	0
initial plug density	N_P	0.06
initial mirror Ratio	R_M0	1
minor radius	RA	0.2
reactor length	ZL	3
magnetic field	BMAG	1/R_M0
electron ECH power†	P_ECH	(i) 0 or (ii) 50
C_{ETG}	C_ETG	0
$C(t = 0)^\dagger$	COM_0	1
Maximum C†	COM_MAX	(i) 3.5 or (ii) 15
$\tau^\dagger_{comp}$	TAU_COM	(i) 5×10^{-5} or (ii) 1×10^{-3} s

$^\dagger C(t) =$ COM_0 + [(COM_MAX - COM_0)/TAU_COM] $\cdot\, t$ for $t \leq \tau_{comp}$, and $C(t) = 0$ for $t > \tau_{comp}$.

Table 8: Example Output Files for symtran.c.f

File Name	Data Columns					
U0.dat	t	$T_i(\rho = 0, t)$	$T_e(\rho = 0, t)$	$n_c(\rho = 0, t)$	$E_B(\rho = 0, t)$	$C(t)$
U.dat	ρ	$T_i(\rho, t_j)$	$T_e(\rho, t_j)$	$n_c(\rho, t_j)$	$E_B(\rho, t_j)$	-

3. FORTRAN files for symtran2.f, symtran4.f and symtran.c.f

As written, **symtran2.f** runs case 4: Figures 8 - 10, **symtran4.f** runs case 8: Figures 24 - 27, and **symtran.c.f** yields Figure 38.

3A. FORTAN code for symtran2.f

```
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NPDE, NPTS, INTPTS, ITYPE, NEQN, NIW, NWK, NW
      PARAMETER        (NPDE=3,NPTS=201,INTPTS=201,ITYPE=1,
     +                 NEQN=NPDE*NPTS,
     +                 NIW=NEQN+24,NWK=(10+6*NPDE)*NEQN,
     +                 NW=NWK+(21+3*NPDE)*NPDE+7*NPTS+54)
*     .. Scalars in Common ..
      DOUBLE PRECISION N_P, E_ALPHA, Z_EFF, PA, BMAG, MR, R_M,
     +                 AS, RA, Ti_EDGE, Te_EDGE, den_EDGE,
     +                 Ti_0, Te_0, den_0, X_PLUG, PS_I,
     +                 ZL, PHI_E, P_ECH,
     +                 N_P_Ti, N_P_Te, EB, E_ALPHA_AV, C_ETG
      INTEGER          N_P_C
*     .. Local Scalars ..
      DOUBLE PRECISION ACC, HX, PI, PIBY2, TOUT, TS
      DOUBLE PRECISION DELTAT, MR, N_P
      DOUBLE PRECISION PHI_I, V1, V2
      DOUBLE PRECISION SIGV1, SIGV2, SIGV3, SIGV4, SIGV5, SIGV6,
     +                 SV, P_ALPHA, PE1, PE1F, PI1, QN, QD, P_INP,
     +                 QF, QNT, QDT, N_PX, N_PU, TAU_II, TAU1, TAU2,
     +                 TAU, EL_I, EL_E, ELX_I, ELX_E, Te_crit, AS1
      INTEGER          I, IFAIL, IND, IT, ITASK, ITRACE, M
*     .. Local Arrays ..
      DOUBLE PRECISION U(NPDE,NPTS), UOUT(NPDE,INTPTS,ITYPE), W(NW),
     +                 X(NPTS), XOUT(INTPTS)
      INTEGER          IW(NIW)
      INTEGER          J, J1, J2, NSTEP, NWRITE, NCORE, K1
*     .. External Functions ..
      DOUBLE PRECISION X01AAF
      EXTERNAL         X01AAF
*     .. External Subroutines ..
      EXTERNAL         BNDARY, D03PCF, D03PZF, PDEDEF, UINIT
*     .. Intrinsic Functions ..
      INTRINSIC        DSIN, DABS, DLOG, DEXP
*************************************************************
*************************************************************
```

```
*      Root finding variables and parameters
*      * .. Parameters ..
*      * .. Local Scalars ..
       DOUBLE PRECISION Z_A, Z_B, CEPS, ETA, H, Z1, Z2, Z_X
*
       DOUBLE PRECISION FR
       DOUBLE PRECISION FACT, R_E, R_I, V1, V2, R_D
       INTEGER JFAIL
       EXTERNAL FR
       COMMON  /root/Z1, Z2, PHI_I
*****************************************************************
*****************************************************************
*     .. Common blocks ..
      COMMON            /Diff/E_ALPHA, Z_EFF, PA, BMAG,
     +                   AS, RA, ZL, PS_I,
     +                   X_PLUG, P_ECH,
     +                   N_P, N_P_Ti, N_P_Te, EB, E_ALPHA_AV,
     +                   C_ETG, R_M, N_P_C
      COMMON            /P1/MR
      COMMON            /BC0/Ti_0, Te_0, den_0
      COMMON            /BC1/Ti_EDGE, Te_EDGE, den_EDGE
      COMMON            /diag/K1
*
*     .. Executable Statements ..
*     Opening a set of output files
*     Running statistics
      OPEN(UNIT=10, FILE='stat.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=20, FILE='U.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=30, FILE='U0.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=40, FILE='Phi.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=50, FILE='Power.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=60, FILE='Power.x.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=70, FILE='Endloss.x.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=100, FILE='diag.dat1',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=110, FILE='diag.dat2',FORM='FORMATTED',
     *     STATUS='OLD')
*
      WRITE (10,*) 'D03PCF Example Program Results'
*
*     Local Error parameter
```

```
      ACC = 1.0D-4
*
*     Selecting spatial geometry
*     M = 0 (Cartesian),
*     M = 1 (Cylindrical)
*     M = 2 (Spherical)
      M = 1
*
*     Program message control parameter
      ITRACE = 0
*
*     Parameter used in PDE
*
*     Parameter starting the integration in time
      IND = 0
*
*     Normal Computation of output values at t = TOUT
      ITASK = 1
*
*     Various input parameters
*     Ion temperature at edge (r/R = 1) [KeV]
      Ti_EDGE = 1.0D-3
*
*     Electron temperature at edge (r/R = 1) [KeV]
      Te_EDGE = 1.0D-3
*
*     Density at edge (r/R = 1) [1E20/m**3]
      den_EDGE = 1.00D0
*
*     Initial ion temperature at origin [KeV]
      Ti_0 = 2.0*Ti_EDGE
*
*     Initial ion temperature at origin [KeV]
      Te_0 = 2.0*Te_EDGE
*
*     Initial density at origin [1E20/m**3]
      den_0 = den_EDGE
*
*     Mass ratio between DT nucleus and electron
      MR = 4.583D3
*
*     Plug density magnitude [1E20/m^3]
      N_P = 10.0D0
*
*     Choosing functional dependence of the plug density
*       N_P_C = 1 -> NPX = const
*       N_P_C = 2 -> NPX = N_P*(1 - N_PU/EB)
*       N_P_C = 3 -> NPX = N_P*(1 - N_PU/EB)*(1 - X**2)
```

```
      N_P_C = 2
*
*     Coefficient for Ti used in the plug density
      N_P_Ti = 0.0D0
*
*     Coefficient for Ti used in the plug density
      N_P_Te = 7.0D0
*
*     Beam energy used in the plug density
      EB = 1.0D3
*
*     Alpha energy [KeV]
      E_ALPHA = 3.52D3
*
*     Average alpha energy used in the partition of alpha power
*       between ions and electrons
      E_ALPHA_AV = 2.0D3
*
*     Z-effective for Bremstrahlun power
      Z_EFF = 0.0D0
*
*     Mirror ratio
      R_M = 9.2D0
*
*     Pashtukhov Constant
      PA = (3.1415927D0)**(0.5D0)/4.0D0*(R_M+1.0D0)*
     +         DLOG(2.0D0*R_M+2.0D0)/R_M
*
*     Radius [m]
      RA = 1.5D0
*
*     Length of reactor [m]
      ZL = 3.0D1
*
*     Magnetic field [T]
      BMAG = 1.0D0
*
*     Electron ECH [KeV*1E20]/[m**3*sec]
      P_ECH = 1.0D3
*
*     Ratio of P_ECH power input
      AS = 1.0D0
*
*     Core electron temperature at which to change P_ECH
      Te_crit = 4.0D1
*
*     Ratio of P_ECH after core electron temperature has reached
*            Te_crit
```

```
      AS1 = 7.0D-2
*
*     Ion external power  [KeV*1E20]/[m**3*sec]
      PS_I = 0.0D0
*
*     Plug radius (normalized)
      X_PLUG = 0.98D0
*
*     Initial value of time-integrated numerator of Q
      QNT = 0.0D0
*
*     Initial value of time-integrated denominator of Q
      QDT = 0.0D0
*
*     Coefficient for the ETG diffusion coefficient
      C_ETG = 0.0D0
*
*     Set spatial mesh points spacing using sine
*
*     This is Pi/2
      PIBY2 = 0.5D0*X01AAF(PI)
*
*     This is step-size
      HX = PIBY2/(NPTS-1)
*
*     Starting and ending points
      X(1) = 0.0D0
      X(NPTS) = 1.0D0
*
*     Do-loop for mesh points used in PDE
      DO 20 I = 2, NPTS - 1
         X(I) = DSIN(HX*(I-1))
20    CONTINUE
*
*     Creating a set of output mesh points
*
*     Starting and ending points
      XOUT(1) = X(1)
      XOUT(INTPTS) = X(NPTS)
*
*     Do-loop for output points
      DO 1000 J1 = 2, INTPTS - 1
         XOUT(J1) = (J1 - 1)*(X(NPTS) - X(1))/(INTPTS-1)
1000  CONTINUE
*
*     Set initial conditions
*
*     Starting time
```

```
      TS = 0.0D0
*
*     Time step
      DELTAT = 1.0D-5
*
*     Number of time steps
      NSTEP = 1000000
*
*     Number of times the spatial profiles are outputed
      NWRITE = 20
*
*     The frequency of printing out core values (at (r/R) = 0)
      NCORE = 1000
*
*     Starting number for which a diagnostic file is needed
      K1 = 1
*
*     Running statistics
      WRITE (10,99999) ACC, E_ALPHA
*
*     Set the initial values
      CALL UINIT(U,X,NPTS)
*
*     Writing initial values to file
      WRITE (30,82000) TS, Ti_0, Te_0, den_0, AS*P_ECH
*
*     This is the main time do-loop
      DO 40 IT = 1, NSTEP
*
*     Priting out error messsage parameter
         IFAIL = -1
*     Definition of the next time-step
         TOUT = IT*DELTAT
*
*     Control of P_ECH input
         IF (UOUT(2,1,1).GT.Te_crit) THEN
            AS = AS1
         ENDIF
*
*     Calling the PDE solver subroutine
         CALL D03PCF(NPDE,M,TS,TOUT,PDEDEF,BNDARY,U,NPTS,X,ACC,W,NW,IW,
     +               NIW,ITASK,ITRACE,IND,IFAIL)
*
*        Interpolate at required spatial points
*
         CALL D03PZF(NPDE,M,U,NPTS,X,XOUT,INTPTS,ITYPE,UOUT,IFAIL)
*
*        Writing results to output files
```

```
*
         DO 1300 I = 1, NSTEP/NCORE
            IF (IT.EQ.I*NCORE) THEN
            WRITE (30,82000) TOUT, UOUT(1,1,1), UOUT(2,1,1),
     +                       UOUT(3,1,1), AS*P_ECH
            ENDIF
1300     CONTINUE
*
         DO 1200 J = 1, NWRITE
         IF (IT.EQ.J*NSTEP/NWRITE) THEN
*
*        Initializing the endloss spatial integration
         ELX_I = 0.0D0
         ELX_E = 0.0D0
*
*        Time
            WRITE (20,80000) TOUT
            WRITE (40,80000) TOUT
            WRITE (70,80000) TOUT
*
*        Spatial values
         DO 1100 J1 = 1, INTPTS
            IF (DABS(UOUT(1,J1,1)).LT.1.0D-10) THEN
                UOUT(1,J1,1) = 0.0D0
            ENDIF
            IF (DABS(UOUT(2,J1,1)).LT.1.0D-10) THEN
                UOUT(2,J1,1) = 0.0D0
            ENDIF
            IF (DABS(UOUT(3,J1,1)).LT.1.0D-10) THEN
                UOUT(3,J1,1) = 0.0D0
            ENDIF
*
*
*           Calculating n_p
              IF (XOUT(J1).LT.X_PLUG) THEN
               N_PU = (N_P_Ti*UOUT(1,J1,1) + N_P_Te*UOUT(2,J1,1))
               IF (N_P_C.EQ.1) THEN
                N_PX = N_P
               ELSE IF (N_P_C.EQ.2) THEN
                N_PX = N_P*(1.0D0 - N_PU/EB)
               ELSE IF (N_P_C.EQ.3) THEN
                N_PX = N_P*(1.0D0 - N_PU/EB)*(1.0D0-XOUT(J1)**2.0D0)
               ENDIF
              ELSE
                N_PX = 0.0D0
              ENDIF
*
*           Printing out Ti, Te, den, n_p
```

```
            WRITE (20,81000) XOUT(J1), UOUT(1,J1,1), UOUT(2,J1,1),
     +                       UOUT(3,J1,1), N_PX
*
*           Checking whether Phi_e and Phi_i satisfy condition
            IF (XOUT(J1).LT.X_PLUG) THEN
              Z1 = UOUT(1,J1,1)
              Z2 = UOUT(2,J1,1)
*
              N_PU = (N_P_Ti*Z1 + N_P_Te*Z2)
              IF (N_P_C.EQ.1) THEN
                N_PX = N_P
              ELSE IF (N_P_C.EQ.2) THEN
                N_PX = N_P*(1.0D0 - N_PU/EB)
              ELSE IF (N_P_C.EQ.3) THEN
                N_PX = N_P*(1.0D0 - N_PU/EB)*(1.0D0-XOUT(J1)**2.0D0)
              ENDIF
*
              IF (N_PX.GT.UOUT(3,J1,1)) THEN
                PHI_I = Z2*DLOG(N_PX/UOUT(3,J1,1))
              ELSE
                PHI_I = 0.0D0
              ENDIF
*
*           Step size for root search
              H  = 1.0D-1
*           Error tolerance
              CEPS = 1.0D-7
*           Closeness of FR to 0.0
              ETA = 0.0D0
*           Root finding program progress monitor parameter
              JFAIL = 1
*           Factor used in initial root estimate
              FACT = 1.0D0
*           Initial estimate for root
              Z_X = PHI_I*(Z2/Z1)
*
            CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
*
*           Checking if the root Z_X is correct
              R_E = Z_X/Z2
              R_I = PHI_I/Z1
              V1 = DLOG(R_E) + R_E
              V2 = 0.50D0*DLOG(MR)
     +            + 1.5D0*DLOG(Z1/Z2)
     +            + DLOG(R_I) + R_I
              R_D = DABS((V2-V1)/V1)
*
              IF (Z_X.LE.0.0D0.OR.R_D.GT.1.0D-2) THEN
```

```
               FACT = FACT + 1.0D0
               H  = H + 10.0D0
               Z_X = PHI_I*(Z2/Z1)
               CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
               R_E = Z_X/Z2
               R_I = PHI_I/Z1
               V1 = DLOG(R_E) + R_E
               V2 = 0.50D0*DLOG(MR)
     +             + 1.5D0*DLOG(Z1/Z2)
     +             + DLOG(R_I) + R_I
               R_D = DABS((V2-V1)/V1)
              ENDIF
*
              PHI_E = Z_X
            ELSE
              PHI_I = 0.0D0
              PHI_E = 4.0D0*UOUT(2,J1,1)
              V1 = 0.0D0
              V2 = 0.0D0
              FACT = 0.0D0
            ENDIF
*
            WRITE (40,83000) XOUT(J1),PHI_I,PHI_E,R_D
*
*        Writing End-loss terms to file
*        Characteristics times
         TAU_II =  1.6D16*UOUT(1,J1,1)**(1.5D0)/(UOUT(3,J1,1)*1.0D20)
         IF (PHI_I/UOUT(1,J1,1).LT.1.0D2) THEN
          TAU1 = TAU_II*PA*PHI_I/UOUT(1,J1,1)*DEXP(PHI_I/UOUT(1,J1,1))
         ELSE
          TAU1 = 1.0D43
         ENDIF
         TAU2 = 3.6D-6*R_M*ZL/UOUT(1,J1,1)**(0.5D0)
     +              *DEXP(PHI_I/UOUT(1,J1,1))
         TAU = TAU1 + TAU2
*
*        End-loss terms
         EL_I = UOUT(3,J1,1)*(PHI_I + UOUT(1,J1,1))/TAU
         EL_E = UOUT(3,J1,1)*(PHI_E + UOUT(2,J1,1))/TAU
         IF (J1.LT.INTPTS) THEN
           ELX_I = ELX_I + EL_I*XOUT(J1+1)*(XOUT(J1+1)-XOUT(J1))
           ELX_E = ELX_E + EL_E*XOUT(J1+1)*(XOUT(J1+1)-XOUT(J1))
         ENDIF
*
         WRITE (70,86000) XOUT(J1), EL_I, EL_E
*
1100     CONTINUE
*
```

```
        WRITE (20,*)
        WRITE (40,*)
        WRITE (70,*)
        ENDIF
1200    CONTINUE
*
*       Calculating Q
*
*       Starting value for numerator of Q
        QN = 0.0D0
*
*       Starting value for denominator of W
        QD = 0.0D0
*
*       Doing the spatial and time integration of various powers
        DO 1400 J2 = 1, INTPTS-1
*       <Sigma*Velocity>_fusion reaction
        IF (UOUT(1,J2,1).GE.1.0D0.AND.UOUT(1,J2,1).LE.8.0D1) THEN
           SIGV1 = (-2.138D1)*UOUT(1,J2,1)**(-2.935D-1)
           SIGV2 = -2.520D1
           SIGV3 =  (-7.101D-2)*UOUT(1,J2,1)
           SIGV4 =  (1.938D-4)*UOUT(1,J2,1)**2.0D0
           SIGV5 =  (4.925D-6)*UOUT(1,J2,1)**3.0D0
           SIGV6 =  (-3.984D-8)*UOUT(1,J2,1)**4.0D0
           SV = (1.0D-6)*DEXP(SIGV1+SIGV2+SIGV3+SIGV4+SIGV5+SIGV6)
        ELSE
           SV = 0.0D0
        ENDIF
*
*       Alpha Power
        P_ALPHA = 0.25D20*UOUT(3,J2,1)**(2.0D0)*SV*E_ALPHA
*
*       Input Powers
        IF (XOUT(J2).LT.X_PLUG) THEN
          PE1F = 1.0D0 - N_P_Te*UOUT(2,J2,1)/EB
*         PE1 = AS*P_ECH/PE1F
          PE1 = AS*P_ECH
          PI1 = AS*PS_I
        ELSE
          PE1 = 0.0D0
          PI1 = 0.0D0
        ENDIF
*
*       Integration of Alpha Power
        QN = QN + 5.0D0*(XOUT(J2+1)-XOUT(J2))*XOUT(J2+1)*P_ALPHA
*
*       Integration of input power
        P_INP = PE1 + PI1
```

```
         QD = QD + (XOUT(J2+1)-XOUT(J2))*XOUT(J2+1)*P_INP
*
*        Writing to output file
*        IF (J2.EQ.1) THEN
*         WRITE (60,85000) TOUT, XOUT(J2), P_ALPHA, P_INP, QN, QD
*         WRITE (60,85000) TOUT, XOUT(J2), P_ALPHA
*        ENDIF
*
*
1400     CONTINUE
*
         QNT = QNT + QN*DELTAT
         QDT = QDT + QD*DELTAT
         QF = QNT/QDT
*
         DO 1500 I = 1, NSTEP/NCORE
            IF (IT.EQ.I*NCORE) THEN
            WRITE (50,84000) TOUT, QN, QD, QN/(QD+1.0D-5), AS*P_ECH
            ENDIF
1500     CONTINUE
*
*
         IF (IFAIL.EQ.3) THEN
           WRITE (10,90000) 'IFAIL IS', IFAIL, TOUT
         ENDIF
*
40    CONTINUE
*
*     Print integration statistics
*
      WRITE (10,99997) IW(1), IW(2), IW(3), IW(5)
      STOP
*
99999 FORMAT (//' Accuracy requirement  = ',D12.5,/' Parameter E_AL =',
     +       '       ',D12.3,/)
99997 FORMAT (' Number of integration steps in time                  ',
     +       I4,/' Number of residual evaluations of resulting ODE sys',
     +       'tem',I4,/' Number of Jacobian evaluations               ',
     +       '         ',I4,/' Number of iterations of nonlinear solve',
     +       'r              ',I4,/)
90000 FORMAT (1X,A,I1,D12.4)
80000 FORMAT (1X,D12.4)
81000 FORMAT (1X,5D12.4)
82000 FORMAT (1X,6D12.4)
*83000 FORMAT (1X,5D12.4,F7.2)
83000 FORMAT (1X,6D12.4)
84000 FORMAT (1X,6D12.4)
85000 FORMAT (1X,6D12.4)
```

```
86000 FORMAT (1X,5D12.4)
      END
*
      SUBROUTINE UINIT(U,X,NPTS)
*     Routine for PDE initial conditon
*     .. Scalar Arguments ..
      INTEGER          NPTS
*     .. Array Arguments ..
      DOUBLE PRECISION U(3,NPTS), X(NPTS)
*     .. Scalars in Common ..
      DOUBLE PRECISION Ti_EDGE, Te_EDGE, den_EDGE,
     +                 Ti_0, Te_0, den_0
*     .. Local Scalars ..
      INTEGER          I
*     .. Common blocks ..
      COMMON           /BC0/Ti_0, Te_0, den_0
      COMMON           /BC1/Ti_EDGE, Te_EDGE, den_EDGE
*     .. Intrinsic functions ..
      INTRINSIC        DLOG, DEXP
*     .. Executable Statements ..
      WRITE (10,*)
      WRITE (10,*) 'Checking root finder'
      WRITE (10,*)
*
      DO 20 I = 1, NPTS
*
         U(1,I) = (Ti_0-Ti_EDGE)*(1.0D0-X(I)**2.0D0)+Ti_EDGE
         U(2,I) = (Te_0-Te_EDGE)*(1.0D0-X(I)**2.0D0)+Te_EDGE
*        U(3,I) = (den_0)*(1.0D0-X(I)**2.0D0)+den_EDGE
         U(3,I) = (den_0-den_EDGE)*(1.0D0-X(I)**2.0D0)+den_EDGE
*
*     Printing out the results
         WRITE (10,70000) X(I), U(1,I), U(2,I), U(3,I)
*
20    CONTINUE
         WRITE (10,*)
70000    FORMAT (1X,4D12.4)
      RETURN
      END
*
*
       DOUBLE PRECISION FUNCTION FR(Z_X)
*      * .. Scalar Arguments ..
       DOUBLE PRECISION Z_X, Z1, Z2, PHI_I, MR
*      * .. Intrinsic Functions ..
       INTRINSIC DEXP, DLOG, DABS
*
       COMMON /P1/MR
```

```
       COMMON /root/Z1, Z2, PHI_I
*
*      * .. Executable Statements ..
*
       FR = Z_X/Z2*DEXP(Z_X/Z2) - MR**(0.5D0)*(Z1/Z2)**(1.5D0)
     #            *(PHI_I/Z1)*DEXP(PHI_I/Z1)
       RETURN
       END
*
*
      SUBROUTINE PDEDEF(NPDE,T,X,U,DUDX,P,Q,R,IRES)
*     .. Scalar Arguments ..
      DOUBLE PRECISION  T, X
      INTEGER           IRES, NPDE
      INTEGER           K2
*     .. Array Arguments ..
      DOUBLE PRECISION  DUDX(NPDE), P(NPDE,NPDE), Q(NPDE), R(NPDE),
     +                  U(NPDE)
      DOUBLE PRECISION  PHI_I, SIGV1, SIGV2, SIGV3, SIGV4, SIGV5,
     +                  SIGV6, SV, P_ALPHA, P_BREM, TAU_II, TAU_IE,
     +                  TAU, R_LI, CHI_I, CHI_E, D_N, Q11, Q12, Q21,
     +                  Q22, Q23, Q24, Q25, PHI_E, Q13,
     +                  N_PX, U30, N_PU, XOLD, TAU_A, F_A_I,
     +                  Q14, CHI_E_ETG, INV_L_CRIT
*     .. Scalars in Common ..
      DOUBLE PRECISION  N_P, E_ALPHA, Z_EFF, PA, BMAG, MR,
     +                  AS, RA, X_PLUG, ZL, PS_I,
     +                  P_ECH, N_P_Ti, N_P_Te,
     +                  EB, E_ALPHA_AV, C_ETG, R_M
      INTEGER           K1, N_P_C
*     .. Intrinsic Functions ..
      INTRINSIC        DABS, DLOG, DEXP
*****************************************************************
*****************************************************************
*      Root finding variables and parameters
*      * .. Parameters ..
*      * .. Local Scalars ..
       DOUBLE PRECISION Z_A, Z_B, CEPS, ETA, H, Z1, Z2, Z_X
*
       DOUBLE PRECISION FR
       DOUBLE PRECISION FACT, R_E, R_I, V1, V2, R_D
       INTEGER JFAIL
       EXTERNAL FR
       COMMON  /root/Z1, Z2, PHI_I
*****************************************************************
*****************************************************************
*     .. Common blocks ..
      COMMON           /Diff/E_ALPHA, Z_EFF, PA, BMAG,
```

```
     +                  AS, RA, ZL, PS_I,
     +                  X_PLUG, P_ECH,
     +                  N_P, N_P_Ti, N_P_Te, EB, E_ALPHA_AV,
     +                  C_ETG, R_M, N_P_C
      COMMON           /P1/MR
      COMMON           /diag/K1
*     .. Executable Statements ..
*     Variables and Parameters
*     U(1) = Ti(x,t), U(2) = Te(x,t), U(3) = n(x,t)
*
*     Preventing trouble when quantities become negative
      IF (U(1).LE.0.0D0) THEN
         U(1) = 1.0D-8
      ENDIF
*
      IF (U(2).LE.0.0D0) THEN
         U(2) = 1.0D-8
      ENDIF
*
      IF (U(3).LE.0.0D0) THEN
         U(3) = 1.0D-8
      ENDIF
*
*     Ion potential
*
*     Finding density at (r/R) = 0
      IF (X.LT.XOLD) THEN
        U30 = U(3)
      ENDIF
*
*     Definition of the plug density
      N_PU = N_P_Ti*U(1) + N_P_Te*U(2)
*
      IF (N_P_C.EQ.1) THEN
         N_PX = N_P
      ELSE IF (N_P_C.EQ.2) THEN
         N_PX = N_P*(1.0D0 - N_PU/EB)
      ELSE IF (N_P_C.EQ.3) THEN
         N_PX = N_P*(1.0D0 - N_PU/EB)*(1.0D0-X**2.0D0)
      ENDIF
*
      IF (N_PX.GT.U(3)) THEN
        PHI_I = U(2)*DLOG(N_PX/U(3))
      ELSE
        PHI_I = 0.0D0
      ENDIF
*
*     Electron potential
```

```
       IF (X.LT.X_PLUG) THEN
         Z1 = U(1)
         Z2 = U(2)
*     Step size for root search
         H  = 1.0D-1
*     Error tolerance
         CEPS = 1.0D-7
*     Closeness of FR to 0.0
         ETA = 0.0D0
*     Root finding program progress monitor parameter
         JFAIL = 1
*     Factor used in initial root estimate
         FACT = 1.0D0
*     Initial estimate for root
         Z_X = PHI_I*(Z2/Z1)
*
      CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
*
*     Checking if the root Z_X is correct
         R_E = Z_X/U(2)
         R_I = PHI_I/U(1)
         V1 = DLOG(R_E) + R_E
         V2 = 0.50D0*DLOG(MR)
     +       + 1.5D0*DLOG(Z1/Z2)
     +       + DLOG(R_I) + R_I
         R_D = DABS((V2-V1)/V1)
*
         IF (Z_X.LE.0.0D0.OR.R_D.GT.1.0D-2) THEN
          FACT = FACT + 1.0D0
          H  = H + 10.0D0
          Z_X = PHI_I*(Z2/Z1)
          CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
          R_E = Z_X/U(2)
          R_I = PHI_I/U(1)
          V1 = DLOG(R_E) + R_E
          V2 = 0.50D0*DLOG(MR)
     +        + 1.5D0*DLOG(Z1/Z2)
     +        + DLOG(R_I) + R_I
          R_D = DABS((V2-V1)/V1)
         ENDIF
*
         PHI_E = Z_X
       ELSE
         PHI_I = 0.0D0
         PHI_E = 4.0D0*U(2)
         V1 = 0.0D0
         V2 = 0.0D0
         FACT = 0.0D0
```

```
       ENDIF
*
*     <Sigma*Velocity>_fusion reaction
      IF (U(1).GE.1.0D0.AND.U(1).LE.8.0D1) THEN
        SIGV1 = (-2.138D1)*U(1)**(-2.935D-1)
        SIGV2 = -2.520D1
        SIGV3 =  (-7.101D-2)*U(1)
        SIGV4 =  (1.938D-4)*U(1)**2.0D0
        SIGV5 =  (4.925D-6)*U(1)**3.0D0
        SIGV6 =  (-3.984D-8)*U(1)**4.0D0
        SV = (1.0D-6)*DEXP(SIGV1+SIGV2+SIGV3+SIGV4+SIGV5+SIGV6)
      ELSE
        SV = 0.0D0
      ENDIF
*
*     Alpha Power
      P_ALPHA = 0.25D20*U(3)**(2.0D0)*SV*E_ALPHA
*
*     Bremstrahlung
      P_BREM =  0.3125D0*Z_EFF*U(3)**(2.0D0)*U(2)**(0.5D0)
*
*     Characteristic Times
      TAU_II =  1.6D16*U(1)**(1.5D0)/(U(3)*1.0D20)
      TAU_IE =  1.0D18*U(2)**(1.5D0)/(U(3)*1.0D20)
      IF (PHI_I/U(1).LT.1.0D2) THEN
        TAU = TAU_II*PA*PHI_I/U(1)*DEXP(PHI_I/U(1))
     +      + 3.6D-6*R_M*ZL/U(1)**(0.5D0)*DEXP(PHI_I/U(1))
      ELSE
        TAU = 1.0D43
      ENDIF
*
*     Fraction of alpha power going to the ions
      TAU_A = 1.6D16*E_ALPHA_AV**(1.5D0)/(U(3)*1.0D20)
*
      F_A_I = TAU_IE*(E_ALPHA_AV - U(1))
     +       /(TAU_IE*(E_ALPHA_AV - U(1))
     +           + TAU_A*(E_ALPHA_AV - U(2)))
*     F_A_I = 0.0D0
*
*     Ion gyro-radius
      R_LI =  6.4D-3*U(1)**(0.5D0)/BMAG
*
*     Diffusion coefficients
      CHI_I = R_LI**(2.0D0)/TAU_II
      CHI_E = MR**(-0.5D0)*(U(1)/U(2))**(0.5D0)*CHI_I
*
      INV_L_CRIT = DABS(0.66667D0*DUDX(3)/U(3))
      CHI_E_ETG = C_ETG*U(2)**(1.5D0)/BMAG**(2.0D0)
```

```
     +           *(DABS(DUDX(2))/U(2) - INV_L_CRIT)
*
      IF (CHI_E_ETG.LE.0.0D0) THEN
        CHI_E_ETG = 0.0D0
      ENDIF
*
      D_N = CHI_E
*
*     Coefficients needed in the PDE solver
*
*     U(1) - T_i
*     Coefficients for time-derivatives
      P(1,1) = 1.5D0*U(3)
      P(1,2) = 0.0D0
      P(1,3) = 1.5D0*U(1)
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q11 = U(3)*(U(2)-U(1))/TAU_IE
      Q12 = -U(3)*(PHI_I+U(1))/TAU
*
      IF (X.LT.X_PLUG) THEN
        Q13 = AS*PS_I
      ELSE
        Q13 = 0.0D0
      ENDIF
*
*     Alpha heating of the ions
      Q14 = P_ALPHA*F_A_I
*
      Q(1) = -(Q11+Q12+Q13+Q14)
*
*     Coefficients for space-derivatives
      R(1) = U(3)*CHI_I/RA**(2.0D0)*DUDX(1)
*
*
*     U(2) - T_e
*     Coefficients for time-derivatives
      P(2,1) = 0.0D0
      P(2,2) = 1.5D0*U(3)
      P(2,3) = 1.5D0*U(2)
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q21 = -U(3)*(U(2)-U(1))/TAU_IE
*
      Q22 = -U(3)*(PHI_E+U(2))/TAU
*
      Q23 = P_ALPHA*(1.0D0 - F_A_I)
*
```

```
      IF (X.LT.X_PLUG) THEN
        Q24 = AS*P_ECH
      ELSE
        Q24 = 0.0D0
      ENDIF
*
      Q25 = -P_BREM
*
      Q(2) = -(Q21+Q22+Q23+Q24+Q25)
*
*     Coefficients for space-derivatives
      R(2) = U(3)*(CHI_E+CHI_E_ETG)/RA**(2.0D0)*DUDX(2)
*
*
*     U(3) - n
*     Coefficients for time-derivatives
      P(3,1) = 0.0D0
      P(3,2) = 0.0D0
      P(3,3) = 1.0D0
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q(3) = 0.0D0
*
*     Coefficients for space-derivatives
*     IF (X.LT.X_PLUG) THEN
*       R(3) = CHI_E/RA**(2.0D0)*DUDX(3)
*     ELSE
*       R(3) = 0.0D0
*     ENDIF
*
      R(3) = 0.0D0
*
*     Printing out diagnostic files if needed
*     IF (T.GE.6.000D-1) THEN
*     K1 = K1 + 1
*     DO 2000 K2 = 1, 10
*      IF (K2.EQ.K1/1000) THEN
*      WRITE (100,60000) T, X, U30, N_PU, N_PX, PHI_I
*      WRITE (110,61000) T, X, Q22, Q23, Q24, Q(2)
*      60000  FORMAT (1X,6D12.4)
*      61000  FORMAT (1X,7D12.4)
*      ENDIF
*      2000  CONTINUE
*     ENDIF
*
*
      XOLD = X
*
```

```
      RETURN
      END
*
      SUBROUTINE BNDARY(NPDE,T,U,UX,IBND,BETA,GAMMA,IRES)
*     .. Scalar Arguments ..
      DOUBLE PRECISION  T
      INTEGER           IBND, IRES, NPDE
      DOUBLE PRECISION Ti_EDGE, Te_EDGE, den_EDGE
*     .. Array Arguments ..
      DOUBLE PRECISION  BETA(NPDE), GAMMA(NPDE), U(NPDE), UX(NPDE)
*     .. Common Blocks ..
      COMMON            /BC1/Ti_EDGE, Te_EDGE, den_EDGE
*     .. Executable Statements ..
*     Boundary condition is defined in the form
*              BETA(I)*R(I) = GAMMA(I)
*     Usually R(I) is proportional to dU(I)/dr so that BC involving
*     df/dr has BETA(I) = 1.0
*     Dirichlet BC is usually set with BETA(I) = 0.0 and
*     GAMMA(I) = U(I) - U_BC
*     IBND = 0 denotes the left boundary point
      IF (IBND.EQ.0) THEN
         BETA(1) = 1.0D0
         BETA(2) = 1.0D0
         BETA(3) = 1.0D0
         GAMMA(1) = 0.0D0
         GAMMA(2) = 0.0D0
         GAMMA(3) = 0.0D0
      ELSE
         BETA(1) = 0.0D0
         BETA(2) = 0.0D0
         BETA(3) = 0.0D0
         GAMMA(1) = U(1) - Ti_EDGE
         GAMMA(2) = U(2) - Te_EDGE
         GAMMA(3) = U(3) - den_EDGE
      END IF
      RETURN
      END
```

3B. FORTAN code for symtran4.f

```
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NPDE, NPTS, INTPTS, ITYPE, NEQN, NIW, NWK, NW
      PARAMETER        (NPDE=4,NPTS=201,INTPTS=201,ITYPE=1,
     +                 NEQN=NPDE*NPTS,
     +                 NIW=NEQN+24,NWK=(10+6*NPDE)*NEQN,
     +                 NW=NWK+(21+3*NPDE)*NPDE+7*NPTS+54)
*     .. Scalars in Common ..
      DOUBLE PRECISION N_P, E_ALPHA, Z_EFF, PA, BMAG, MR, R_M,
     +                 AS, S_N, RA, Ti_EDGE, Te_EDGE, den_EDGE,
     +                 Ti_0, Te_0, den_0, PS_I,
     +                 ZL, PHI_E, P_ECH,
     +                 E_ALPHA_AV, C_ETG, C1_N_P,
     +                 C2_N_P, N_P_EDGE, N_P_0, EB,
     +                 PSX
*     .. Local Scalars ..
      DOUBLE PRECISION ACC, HX, PI, PIBY2, TOUT, TS
      DOUBLE PRECISION DELTAT, MR
      DOUBLE PRECISION PHI_I, V1, V2
      DOUBLE PRECISION SIGV1, SIGV2, SIGV3, SIGV4, SIGV5, SIGV6,
     +                 SV, P_ALPHA, PE1, PI1, PE2, QN, QD, P_INP,
     +                 QF, QNT, QDT, TAU_II, TAU1, TAU2,
     +                 TAU, EL_I, EL_E, ELX_I, ELX_E, TAU_IE,
     +                 PE1F, Te_crit, AS1
      INTEGER          I, IFAIL, IND, IT, ITASK, ITRACE, M
*     .. Local Arrays ..
      DOUBLE PRECISION U(NPDE,NPTS), UOUT(NPDE,INTPTS,ITYPE), W(NW),
     +                 X(NPTS), XOUT(INTPTS)
      INTEGER          IW(NIW)
      INTEGER          J, J1, J2, NSTEP, NWRITE, NCORE, K1
*     .. External Functions ..
      DOUBLE PRECISION X01AAF
      EXTERNAL         X01AAF
*     .. External Subroutines ..
      EXTERNAL         BNDARY, D03PCF, D03PZF, PDEDEF, UINIT
*     .. Intrinsic Functions ..
      INTRINSIC        DSIN, DABS, DLOG, DEXP
************************************************************
************************************************************
*      Root finding variables and parameters
*      * .. Parameters ..
*      * .. Local Scalars ..
       DOUBLE PRECISION Z_A, Z_B, CEPS, ETA, H, Z1, Z2, Z_X
*
```

```
       DOUBLE PRECISION FR
       DOUBLE PRECISION FACT, R_E, R_I, V1, V2, R_D
       INTEGER JFAIL
       EXTERNAL FR
       COMMON  /root/Z1, Z2, PHI_I
************************************************************
************************************************************
*     .. Common blocks ..
      COMMON           /Diff/E_ALPHA, Z_EFF, PA, BMAG,
     +                  AS, S_N, RA, ZL, PS_I,
     +                  P_ECH,
     +                  N_P, E_ALPHA_AV,
     +                  C_ETG, R_M, C1_N_P, C2_N_P,
     +                  EB, PSX
      COMMON           /P1/MR
      COMMON           /BC0/Ti_0, Te_0, den_0, N_P_0
      COMMON           /BC1/Ti_EDGE, Te_EDGE, den_EDGE, N_P_EDGE
      COMMON           /diag/K1
*
*     .. Executable Statements ..
*     Opening a set of output files
*     Running statistics
      OPEN(UNIT=10, FILE='stat.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=20, FILE='U.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=30, FILE='U0.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=40, FILE='Phi.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=50, FILE='Power.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=60, FILE='Power.x.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=70, FILE='Endloss.x.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=100, FILE='diag.dat1',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=110, FILE='diag.dat2',FORM='FORMATTED',
     *     STATUS='OLD')
*
      WRITE (10,*) 'D03PCF Example Program Results'
*
*     Local Error parameter
      ACC = 1.0D-4
*
*     Selecting spatial geometry
*     M = 0 (Cartesian),
```

```
*     M = 1 (Cylindrical)
*     M = 2 (Spherical)
      M = 1
*
*     Program message control parameter
      ITRACE = 0
*
*     Parameter used in PDE
*
*     Parameter starting the integration in time
      IND = 0
*
*     Normal Computation of output values at t = TOUT
      ITASK = 1
*
*     Various input parameters
*     Ion temperature at edge (r/R = 1) [KeV]
      Ti_EDGE = 1.0D-3
*
*     Electron temperature at edge (r/R = 1) [KeV]
      Te_EDGE = 1.0D-3
*
*     Density at edge (r/R = 1) [1E20/m**3]
      den_EDGE = 1.0D-2
*
*     Plug density at edge (r/R = 1) [1E20/m**3]
      N_P_EDGE = 1.2D0*den_EDGE
*
*     Initial ion temperature at origin [KeV]
      Ti_0 = 2.0*Ti_EDGE
*
*     Initial ion temperature at origin [KeV]
      Te_0 = 2.0*Te_EDGE
*
*     Initial density at origin [1E20/m**3]
      den_0 = 2.0D0*den_EDGE
*
*     Initial plug density at origin [1E20/m**3]
      N_P_0 = 1.2D0*den_0
*
*     Mass ratio between DT nucleus and electron
      MR = 4.583D3
*
*     Alpha energy [KeV]
      E_ALPHA = 3.52D3
*
*     Average alpha energy used in the partition of alpha power
*       between ions and electrons
```

```
      E_ALPHA_AV = 2.0D3
*
*     Z-effective for Bremstrahlun power
      Z_EFF = 0.0D0
*
*     Mirror ratio
      R_M = 2.0D1
*
*     Pashtukhov Constant
      PA = (3.1415927D0)**(0.5D0)/4.0D0*(R_M+1.0D0)*
     +         DLOG(2.0D0*R_M+2.0D0)/R_M
*
*     Radius [m]
      RA = 1.5D0
*
*     Length of reactor [m]
      ZL = 1.0D2
*
*     Beam energy used in PS_E
      EB = 1.0D3
*
*     Magnetic field [T]
      BMAG = 0.5D0
*
*     Factor in electron input plug power
*     PSX = 2.0D0*(RA/ZL)/R_M**(1.5D0)
      PSX = 1.7E-3
*
*     Ion external power  [KeV*1E20]/[m**3*sec]
      PS_I = 0.0D0
*
*     ECH [KeV*1E20]/[m**3*sec]
      P_ECH = 2.0D4
*
*     Initial factor of ECH power
      AS = 0.0D0
*
*     Core electron temperature at which to change ECH power
      Te_crit = 6.0D1
*
*     Factor of ECH power after core electron temperature has
*            reached Te_crit
      AS1 = 1.0D0
*
*     Factor in external density source [1E20 particles/m^3/s]
      S_N = 5.00D0
*
*     Initial value of time-integrated numerator of Q
```

```
      QNT = 0.0D0
*
*     Initial value of time-integrated denominator of Q
      QDT = 0.0D0
*
*     Coefficient for the ETG diffusion coefficient
      C_ETG = 0.0D0
*
*     Factor used in the plug density source term
      C1_N_P = 1.00D2
*
*     Factor used in the plug density loss term
      C2_N_P = 1.0D0
*
*     Set spatial mesh points spacing using sine
*
*     This is Pi/2
      PIBY2 = 0.5D0*X01AAF(PI)
*
*     This is step-size
      HX = PIBY2/(NPTS-1)
*
*     Starting and ending points
      X(1) = 0.0D0
      X(NPTS) = 1.0D0
*
*     Do-loop for mesh points used in PDE
      DO 20 I = 2, NPTS - 1
         X(I) = DSIN(HX*(I-1))
20    CONTINUE
*
*     Creating a set of output mesh points
*
*     Starting and ending points
      XOUT(1) = X(1)
      XOUT(INTPTS) = X(NPTS)
*
*     Do-loop for output points
      DO 1000 J1 = 2, INTPTS - 1
         XOUT(J1) = (J1 - 1)*(X(NPTS) - X(1))/(INTPTS-1)
1000  CONTINUE
*
*     Set initial conditions
*
*     Starting time
      TS = 0.0D0
*
*     Time step
```

```
      DELTAT = 1.0D-5
*
*     Number of time steps
      NSTEP = 2000000
*
*     Number of times the spatial profiles are outputed
      NWRITE = 10
*
*     The frequency of printing out core values (at (r/R) = 0)
      NCORE = 1000
*
*     Starting number for which a diagnostic file is needed
      K1 = 1
*
*     Running statistics
      WRITE (10,99999) ACC, E_ALPHA, PSX
*
*     Set the initial values
      CALL UINIT(U,X,NPTS)
*
*     Writing initial values to file
      WRITE (30,82000) TS, Ti_0, Te_0, den_0, N_P_0
*
*     This is the main time do-loop
      DO 40 IT = 1, NSTEP
*
*     Priting out error messsage parameter
         IFAIL = -1
*     Definition of the next time-step
*        IF (TOUT.LE.1.0D0) THEN
            TOUT = IT*DELTAT
*        ELSE
*           TOUT = IT*(DELTAT*1.0D1)
*        ENDIF
*
*     Control of external power input
         IF (UOUT(2,1,1).GT.Te_crit) THEN
            AS = AS1
         ENDIF
*
*     Calling the PDE solver subroutine
         CALL D03PCF(NPDE,M,TS,TOUT,PDEDEF,BNDARY,U,NPTS,X,ACC,W,NW,IW,
     +               NIW,ITASK,ITRACE,IND,IFAIL)
*
*        Interpolate at required spatial points
*
         CALL D03PZF(NPDE,M,U,NPTS,X,XOUT,INTPTS,ITYPE,UOUT,IFAIL)
*
```

```
*        Writing results to output files
*
         DO 1300 I = 1, NSTEP/NCORE
            IF (IT.EQ.I*NCORE) THEN
            WRITE (30,82000) TOUT, UOUT(1,1,1), UOUT(2,1,1),
     +                       UOUT(3,1,1), UOUT(4,1,1)
            ENDIF
1300     CONTINUE
*
         DO 1200 J = 1, NWRITE
         IF (IT.EQ.J*NSTEP/NWRITE) THEN
*
*        Initializing the endloss spatial integration
         ELX_I = 0.0D0
         ELX_E = 0.0D0
*
*        Time
            WRITE (20,80000) TOUT
            WRITE (40,80000) TOUT
            WRITE (70,80000) TOUT
*
*        Spatial values
         DO 1100 J1 = 1, INTPTS
            IF (DABS(UOUT(1,J1,1)).LT.1.0D-10) THEN
                UOUT(1,J1,1) = 0.0D0
            ENDIF
            IF (DABS(UOUT(2,J1,1)).LT.1.0D-10) THEN
                UOUT(2,J1,1) = 0.0D0
            ENDIF
            IF (DABS(UOUT(3,J1,1)).LT.1.0D-10) THEN
                UOUT(3,J1,1) = 0.0D0
            ENDIF
            IF (DABS(UOUT(4,J1,1)).LT.1.0D-10) THEN
                UOUT(4,J1,1) = 0.0D0
            ENDIF
*
*        Stop running if numbers become negative
            IF (UOUT(1,J1,1).LT.0.0D0.OR.UOUT(2,J1,1).LT.0.0D0.OR.
     +          UOUT(3,J1,1).LT.0.0D0.OR.UOUT(4,J1,1).LT.0.0D0)  THEN
               IT = NSTEP
            ENDIF
*
*           Printing out Ti, Te, den
            WRITE (20,81000) XOUT(J1), UOUT(1,J1,1), UOUT(2,J1,1),
     +                       UOUT(3,J1,1), UOUT(4,J1,1)
*
*           Checking whether Phi_e and Phi_i satisfy condition
              Z1 = UOUT(1,J1,1)
```

```
              Z2 = UOUT(2,J1,1)
*
              IF (UOUT(4,J1,1).GT.UOUT(3,J1,1)) THEN
                PHI_I = Z2*DLOG(UOUT(4,J1,1)/UOUT(3,J1,1))
              ELSE
                PHI_I = 0.0D0
              ENDIF
*
*           Step size for root search
              H  = 1.0D-1
*           Error tolerance
              CEPS = 1.0D-7
*           Closeness of FR to 0.0
              ETA = 0.0D0
*           Root finding program progress monitor parameter
              JFAIL = 1
*           Factor used in initial root estimate
              FACT = 1.0D0
*           Initial estimate for root
              Z_X = PHI_I*(Z2/Z1)
*
           CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
*
*           Checking if the root Z_X is correct
              R_E = Z_X/Z2
              R_I = PHI_I/Z1
              V1 = DLOG(R_E) + R_E
              V2 = 0.50D0*DLOG(MR)
     +            + 1.5D0*DLOG(Z1/Z2)
     +            + DLOG(R_I) + R_I
              R_D = DABS((V2-V1)/V1)
*
              IF (Z_X.LE.0.0D0.OR.R_D.GT.1.0D-2) THEN
               FACT = FACT + 1.0D0
               H  = H + 10.0D0
               Z_X = PHI_I*(Z2/Z1)
               CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
               R_E = Z_X/Z2
               R_I = PHI_I/Z1
               V1 = DLOG(R_E) + R_E
               V2 = 0.50D0*DLOG(MR)
     +            + 1.5D0*DLOG(Z1/Z2)
     +            + DLOG(R_I) + R_I
               R_D = DABS((V2-V1)/V1)
              ENDIF
*
              PHI_E = Z_X
*
```

```
           WRITE (40,83000) XOUT(J1),PHI_I,PHI_E,R_D
*
*        Writing End-loss terms to file
*        Characteristics times
         TAU_II =  1.6D16*UOUT(1,J1,1)**(1.5D0)/(UOUT(3,J1,1)*1.0D20)
         IF (PHI_I/UOUT(1,J1,1).LT.1.0D2) THEN
          TAU1 = TAU_II*PA*PHI_I/UOUT(1,J1,1)*DEXP(PHI_I/UOUT(1,J1,1))
         ELSE
          TAU1 = 1.0D43
         ENDIF
         TAU2 = 3.6D-6*R_M*ZL/UOUT(1,J1,1)**(0.5D0)
     +              *DEXP(PHI_I/UOUT(1,J1,1))
         TAU = TAU1 + TAU2
*
*        End-loss terms
         EL_I = UOUT(3,J1,1)*(PHI_I + UOUT(1,J1,1))/TAU
         EL_E = UOUT(3,J1,1)*(PHI_E + UOUT(2,J1,1))/TAU
         IF (J1.LT.INTPTS) THEN
           ELX_I = ELX_I + EL_I*XOUT(J1+1)*(XOUT(J1+1)-XOUT(J1))
           ELX_E = ELX_E + EL_E*XOUT(J1+1)*(XOUT(J1+1)-XOUT(J1))
         ENDIF
*
         WRITE (70,86000) XOUT(J1), EL_I, EL_E
*
1100     CONTINUE
*
         WRITE (20,*)
         WRITE (40,*)
         WRITE (70,*)
         ENDIF
1200     CONTINUE
*
*        Calculating Q at fixed time intervals
*
         DO 1500 I = 1, NSTEP/NCORE
          IF (IT.EQ.I*NCORE) THEN
*        Starting value for numerator of Q
         QN = 0.0D0
*
*        Starting value for denominator of W
         QD = 0.0D0
*
*        Doing the spatial and time integration of various powers
         DO 1400 J2 = 1, INTPTS-1
*
*        Checking whether Phi_e and Phi_i satisfy condition
           Z1 = UOUT(1,J2,1)
           Z2 = UOUT(2,J2,1)
```

```
*
           IF (UOUT(4,J2,1).GT.UOUT(3,J2,1)) THEN
              PHI_I = Z2*DLOG(UOUT(4,J2,1)/UOUT(3,J2,1))
           ELSE
              PHI_I = 0.0D0
           ENDIF
*
*          Step size for root search
             H = 1.0D-1
*          Error tolerance
             CEPS = 1.0D-7
*          Closeness of FR to 0.0
             ETA = 0.0D0
*          Root finding program progress monitor parameter
             JFAIL = 1
*          Factor used in initial root estimate
             FACT = 1.0D0
*          Initial estimate for root
             Z_X = PHI_I*(Z2/Z1)
*
           CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
*
*          Checking if the root Z_X is correct
             R_E = Z_X/Z2
             R_I = PHI_I/Z1
             V1 = DLOG(R_E) + R_E
             V2 = 0.50D0*DLOG(MR)
     +            + 1.5D0*DLOG(Z1/Z2)
     +            + DLOG(R_I) + R_I
             R_D = DABS((V2-V1)/V1)
*
             IF (Z_X.LE.0.0D0.OR.R_D.GT.1.0D-2) THEN
              FACT = FACT + 1.0D0
              H  = H + 10.0D0
              Z_X = PHI_I*(Z2/Z1)
              CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
              R_E = Z_X/Z2
              R_I = PHI_I/Z1
              V1 = DLOG(R_E) + R_E
              V2 = 0.50D0*DLOG(MR)
     +             + 1.5D0*DLOG(Z1/Z2)
     +             + DLOG(R_I) + R_I
              R_D = DABS((V2-V1)/V1)
             ENDIF
*
             PHI_E = Z_X
*
*        <Sigma*Velocity>_fusion reaction
```

```
         IF (UOUT(1,J2,1).GE.1.0D0.AND.UOUT(1,J2,1).LE.8.0D1) THEN
            SIGV1 = (-2.138D1)*UOUT(1,J2,1)**(-2.935D-1)
            SIGV2 = -2.520D1
            SIGV3 =  (-7.101D-2)*UOUT(1,J2,1)
            SIGV4 =  (1.938D-4)*UOUT(1,J2,1)**2.0D0
            SIGV5 =  (4.925D-6)*UOUT(1,J2,1)**3.0D0
            SIGV6 =  (-3.984D-8)*UOUT(1,J2,1)**4.0D0
            SV = (1.0D-6)*DEXP(SIGV1+SIGV2+SIGV3+SIGV4+SIGV5+SIGV6)
         ELSE
            SV = 0.0D0
         ENDIF
*
*        Characteristic time
         TAU_IE = 1.0D-2*UOUT(2,J2,1)**(1.5D0)/UOUT(3,J2,1)
*
*        Alpha Power
         P_ALPHA = 0.25D20*UOUT(3,J2,1)**(2.0D0)*SV*E_ALPHA
*
*        Input Powers
           PE1F = 1.0D0 - (PHI_I + PHI_E)/EB
           PE1 = PSX*(UOUT(4,J2,1)**2.0D0)/(UOUT(3,J2,1)*TAU_IE)*EB/PE1F
     +             + AS*P_ECH
           PI1 = AS*PS_I
*
*        Integration of Alpha Power
         QN = QN + 5.0D0*(XOUT(J2+1)-XOUT(J2))*XOUT(J2+1)*P_ALPHA
*
*        Integration of input power
         P_INP = PE1 + PE2 + PI1
         QD = QD + (XOUT(J2+1)-XOUT(J2))*XOUT(J2+1)*P_INP
*
*        Writing to output file
*        IF (J2.EQ.1) THEN
*         WRITE (60,85000) TOUT, XOUT(J2), PHI_I, PHI_E, R_D
*         WRITE (60,85000) TOUT, XOUT(J2), P_ALPHA
*        ENDIF
*
*
1400     CONTINUE
*
*        Calculating the time integral of Q
         QNT = QNT + QN*DELTAT
         QDT = QDT + QD*DELTAT
         QF = QNT/QDT
*
*        Writing the output of various Q
*        WRITE (50,84000) TOUT, QN, QNT, QD, QDT, QN/(QD+1.0D-5)
         WRITE (50,84000) TOUT, QN, QD, QN/(QD+1.0D-5), AS*P_ECH
```

```
*
        ENDIF
1500    CONTINUE
*
*
         IF (IFAIL.EQ.3) THEN
           WRITE (10,90000) 'IFAIL IS', IFAIL, TOUT
         ENDIF
*
40    CONTINUE
*
*     Print integration statistics
*
      WRITE (10,99997) IW(1), IW(2), IW(3), IW(5)
      STOP
*
99999 FORMAT (//' Accuracy requirement  = ',D12.5,/' Parameter E_AL =',
     +       '       ',D12.3,/)
99997 FORMAT (' Number of integration steps in time                  ',
     +       I4,/' Number of residual evaluations of resulting ODE sys',
     +       'tem',I4,/' Number of Jacobian evaluations               ',
     +       '         ',I4,/' Number of iterations of nonlinear solve',
     +       'r              ',I4,/)
90000 FORMAT (1X,A,I1,D12.4)
80000 FORMAT (1X,D12.4)
81000 FORMAT (1X,5D12.4)
82000 FORMAT (1X,6D12.4)
*83000 FORMAT (1X,5D12.4,F7.2)
83000 FORMAT (1X,6D12.4)
84000 FORMAT (1X,6D12.4)
85000 FORMAT (1X,6D12.4)
86000 FORMAT (1X,6D12.4)
      END
*
      SUBROUTINE UINIT(U,X,NPTS)
*     Routine for PDE initial conditon
*     .. Scalar Arguments ..
      INTEGER          NPTS
*     .. Array Arguments ..
      DOUBLE PRECISION U(4,NPTS), X(NPTS)
*     .. Scalars in Common ..
      DOUBLE PRECISION Ti_EDGE, Te_EDGE, den_EDGE, N_P_EDGE,
     +                 Ti_0, Te_0, den_0, N_P_0
*     .. Local Scalars ..
      INTEGER          I
*     .. Common blocks ..
      COMMON           /BC0/Ti_0, Te_0, den_0, N_P_0
      COMMON           /BC1/Ti_EDGE, Te_EDGE, den_EDGE, N_P_EDGE
```

```
*     .. Intrinsic functions ..
      INTRINSIC        DLOG, DEXP
*     .. Executable Statements ..
      WRITE (10,*)
      WRITE (10,*) 'Checking root finder'
      WRITE (10,*)
*
      DO 20 I = 1, NPTS
*
         U(1,I) = (Ti_0-Ti_EDGE)*(1.0D0-X(I)**2.0D0)+Ti_EDGE
         U(2,I) = (Te_0-Te_EDGE)*(1.0D0-X(I)**2.0D0)+Te_EDGE
         U(3,I) = (den_0-den_EDGE)*(1.0D0-X(I)**2.0D0)+den_EDGE
         U(4,I) = (N_P_0-N_P_EDGE)*(1.0D0-X(I)**2.0D0)+N_P_EDGE
*
*     Printing out the results
         WRITE (10,70000) X(I), U(1,I), U(2,I), U(3,I), U(4,I)
*
20    CONTINUE
         WRITE (10,*)
70000    FORMAT (1X,5D12.4)
      RETURN
      END
*
*
       DOUBLE PRECISION FUNCTION FR(Z_X)
*      * .. Scalar Arguments ..
       DOUBLE PRECISION Z_X, Z1, Z2, PHI_I, MR
*      * .. Intrinsic Functions ..
       INTRINSIC DEXP, DLOG, DABS
*
       COMMON /P1/MR
       COMMON /root/Z1, Z2, PHI_I
*
*      * .. Executable Statements ..
*
       FR = Z_X/Z2*DEXP(Z_X/Z2) - MR**(0.5D0)*(Z1/Z2)**(1.5D0)
     #            *(PHI_I/Z1)*DEXP(PHI_I/Z1)
       RETURN
       END
*
*
      SUBROUTINE PDEDEF(NPDE,T,X,U,DUDX,P,Q,R,IRES)
*     .. Scalar Arguments ..
      DOUBLE PRECISION  T, X
      INTEGER           IRES, NPDE
      INTEGER           K2
*     .. Array Arguments ..
      DOUBLE PRECISION  DUDX(NPDE), P(NPDE,NPDE), Q(NPDE), R(NPDE),
```

```
     +                 U(NPDE)
      DOUBLE PRECISION  PHI_I, SIGV1, SIGV2, SIGV3, SIGV4, SIGV5,
     +                  SIGV6, SV, P_ALPHA, P_BREM, TAU_II, TAU_IE,
     +                  TAU, R_LI, CHI_I, CHI_E, D_N, Q11, Q12, Q21,
     +                  Q22, Q23, Q24, Q25, Q31, Q32, PHI_E, Q13,
     +                  U30, XOLD, TAU_A, F_A_I,
     +                  Q14, CHI_E_ETG, Q41, Q42, Q42_1, INV_L_CRIT
*     .. Scalars in Common ..
      DOUBLE PRECISION  N_P, E_ALPHA, Z_EFF, PA, BMAG, MR,
     +                  AS, S_N, RA, ZL, PS_I,
     +                  P_ECH,
     +                  E_ALPHA_AV, C_ETG, R_M, C1_N_P,
     +                  C2_N_P, EB, PSX
      INTEGER           K1
*     .. Intrinsic Functions ..
      INTRINSIC        DABS, DLOG, DEXP
*****************************************************************
*****************************************************************
*      Root finding variables and parameters
*      * .. Parameters ..
*      * .. Local Scalars ..
       DOUBLE PRECISION Z_A, Z_B, CEPS, ETA, H, Z1, Z2, Z_X
*
       DOUBLE PRECISION FR
       DOUBLE PRECISION FACT, R_E, R_I, V1, V2, R_D
       INTEGER JFAIL
       EXTERNAL FR
       COMMON  /root/Z1, Z2, PHI_I
*****************************************************************
*****************************************************************
*     .. Common blocks ..
      COMMON           /Diff/E_ALPHA, Z_EFF, PA, BMAG,
     +                  AS, S_N, RA, ZL, PS_I,
     +                  P_ECH,
     +                  N_P, E_ALPHA_AV,
     +                  C_ETG, R_M, C1_N_P, C2_N_P,
     +                  EB, PSX
      COMMON           /P1/MR
      COMMON           /diag/K1
*     .. Executable Statements ..
*     Variables and Parameters
*     U(1) = Ti(x,t), U(2) = Te(x,t), U(3) = n(x,t), U(4) = n_p(x,t)
*
*     Preventing trouble when quantities become negative
      IF (U(1).LE.0.0D0) THEN
         U(1) = 1.0D-8
      ENDIF
*
```

```
      IF (U(2).LE.0.0D0) THEN
         U(2) = 1.0D-8
      ENDIF
*
      IF (U(3).LE.0.0D0) THEN
         U(3) = 1.0D-8
      ENDIF
*
      IF (U(4).LT.0.0D0) THEN
         U(4) = 1.0D-8
      ENDIF
*
*     Ion potential
*
*     Finding density at (r/R) = 0
      IF (X.LT.XOLD) THEN
        U30 = U(3)
      ENDIF
*
      IF (U(4).GT.U(3)) THEN
        PHI_I = U(2)*DLOG(U(4)/U(3))
      ELSE
        PHI_I = 0.0D0
      ENDIF
*
*     Electron potential
         Z1 = U(1)
         Z2 = U(2)
*     Step size for root search
         H  = 1.0D-1
*     Error tolerance
         CEPS = 1.0D-7
*     Closeness of FR to 0.0
         ETA = 0.0D0
*     Root finding program progress monitor parameter
         JFAIL = 1
*     Factor used in initial root estimate
         FACT = 1.0D0
*     Initial estimate for root
         Z_X = PHI_I*(Z2/Z1)
*
      CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
*
*     Checking if the root Z_X is correct
         R_E = Z_X/U(2)
         R_I = PHI_I/U(1)
         V1 = DLOG(R_E) + R_E
         V2 = 0.50D0*DLOG(MR)
```

```
     +       + 1.5D0*DLOG(Z1/Z2)
     +       + DLOG(R_I) + R_I
             R_D = DABS((V2-V1)/V1)
*
             IF (Z_X.LE.0.0D0.OR.R_D.GT.1.0D-2) THEN
              FACT = FACT + 1.0D0
              H  = H + 10.0D0
              Z_X = PHI_I*(Z2/Z1)
              CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
              R_E = Z_X/U(2)
              R_I = PHI_I/U(1)
              V1 = DLOG(R_E) + R_E
              V2 = 0.50D0*DLOG(MR)
     +        + 1.5D0*DLOG(Z1/Z2)
     +        + DLOG(R_I) + R_I
              R_D = DABS((V2-V1)/V1)
             ENDIF
*
             PHI_E = Z_X
*
*     <Sigma*Velocity>_fusion reaction
      IF (U(1).GE.1.0D0.AND.U(1).LE.8.0D1) THEN
        SIGV1 = (-2.138D1)*U(1)**(-2.935D-1)
        SIGV2 = -2.520D1
        SIGV3 =  (-7.101D-2)*U(1)
        SIGV4 =  (1.938D-4)*U(1)**2.0D0
        SIGV5 =  (4.925D-6)*U(1)**3.0D0
        SIGV6 =  (-3.984D-8)*U(1)**4.0D0
        SV = (1.0D-6)*DEXP(SIGV1+SIGV2+SIGV3+SIGV4+SIGV5+SIGV6)
      ELSE
        SV = 0.0D0
      ENDIF
*
*     Alpha Power
      P_ALPHA = 0.25D20*U(3)**(2.0D0)*SV*E_ALPHA
*
*     Bremstrahlung
      P_BREM =  0.3125D0*Z_EFF*U(3)**(2.0D0)*U(2)**(0.5D0)
*
*     Characteristic Times
      TAU_II =  1.6D16*U(1)**(1.5D0)/(U(3)*1.0D20)
      TAU_IE =  1.0D18*U(2)**(1.5D0)/(U(3)*1.0D20)
      IF (PHI_I/U(1).LT.1.0D2) THEN
        TAU = TAU_II*PA*PHI_I/U(1)*DEXP(PHI_I/U(1))
     +      + 3.6D-6*R_M*ZL/U(1)**(0.5D0)*DEXP(PHI_I/U(1))
      ELSE
        TAU = 1.0D43
      ENDIF
```

```
*
*     Fraction of alpha power going to the ions
      TAU_A = 1.6D16*E_ALPHA_AV**(1.5D0)/(U(3)*1.0D20)
      F_A_I = TAU_IE*(E_ALPHA_AV - U(1))
     +       /(TAU_IE*(E_ALPHA_AV - U(1))
     +           + TAU_A*(E_ALPHA_AV - U(2)))
*
*     Ion gyro-radius
      R_LI =  6.4D-3*U(1)**(0.5D0)/BMAG
*
*     Diffusion coefficients
      CHI_I = R_LI**(2.0D0)/TAU_II
      CHI_E = MR**(-0.5D0)*(U(1)/U(2))**(0.5D0)*CHI_I
*
*
      INV_L_CRIT = DABS(0.66667D0*DUDX(3)/U(3))
      CHI_E_ETG = C_ETG*U(2)**(1.5D0)/BMAG**(2.0D0)
     +           *(DABS(DUDX(2))/U(2) - INV_L_CRIT)
*
      IF (CHI_E_ETG.LE.0.0D0) THEN
        CHI_E_ETG = 0.0D0
      ENDIF
*
      D_N = CHI_E
*
*     Coefficients needed in the PDE solver
*
*     U(1) - T_i
*     Coefficients for time-derivatives
      P(1,1) = 1.5D0*U(3)
      P(1,2) = 0.0D0
      P(1,3) = 1.5D0*U(1)
      P(1,4) = 0.0D0
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q11 = U(3)*(U(2)-U(1))/TAU_IE
      Q12 = -U(3)*(PHI_I+U(1))/TAU
*
      Q13 = AS*PS_I
*
*     Alpha heating of the ions
      Q14 = P_ALPHA*F_A_I
*
      Q(1) = -(Q11+Q12+Q13+Q14)
*
*     Coefficients for space-derivatives
      R(1) = U(3)*CHI_I/RA**(2.0D0)*DUDX(1)
*
```

```
*
*     U(2) - T_e
*     Coefficients for time-derivatives
      P(2,1) = 0.0D0
      P(2,2) = 1.5D0*U(3)
      P(2,3) = 1.5D0*U(2)
      P(2,4) = 0.0D0
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q21 = -U(3)*(U(2)-U(1))/TAU_IE
*
      Q22 = -U(3)*(PHI_E+U(2))/TAU
*
      Q23 = P_ALPHA*(1.0D0 - F_A_I)
*
      Q24 = PSX*U(4)**(2.0D0)/(U(3)*TAU_IE)*EB + AS*P_ECH
*
      Q25 = -P_BREM
*
      Q(2) = -(Q21+Q22+Q23+Q24+Q25)
*
*     Coefficients for space-derivatives
      R(2) = U(3)*(CHI_E+CHI_E_ETG)/RA**(2.0D0)*DUDX(2)
*
*
*     U(3) - n
*     Coefficients for time-derivatives
      P(3,1) = 0.0D0
      P(3,2) = 0.0D0
      P(3,3) = 1.0D0
      P(3,4) = 0.0D0
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q31 = S_N*U(3)
*
      Q32 = -U(3)/TAU
*
      Q(3) = -(Q31+Q32)
*
*     Coefficients for space-derivatives
      R(3) = CHI_E/RA**(2.0D0)*DUDX(3)
*
*
*     U(4) - N_P
*     Coefficients for time-derivatives
      P(4,1) = 0.0D0
      P(4,2) = 0.0D0
      P(4,3) = 0.0D0
```

```
      P(4,4) = 1.0D0
*
*     Coefficients for sources (Q>0) and sinks(Q<0)
      Q41 = U(4)*C1_N_P
      IF ((PHI_I+ PHI_E).LT.EB) THEN
        Q42_1 = 1.0D0 - C2_N_P*(PHI_I + PHI_E)/EB
      ELSE
        Q42_1 = 1.0D-6
      ENDIF
      Q42 = -U(4)**(2.0D0)/(U(3)*TAU_IE)/Q42_1
*
      Q(4) = -(Q41+Q42)
*
*     Coefficients for space-derivatives
      R(4) = 0.0D0
*
*
*     Printing out diagnostic files if needed
*     IF (T.GE.1.1D0) THEN
*     K1 = K1 + 1
*     DO 2000 K2 = 1, 10
*      IF (K2.EQ.K1/1000) THEN
*      WRITE (100,60000) T, X, U(1), U(2), U(3), U(4), AS*P_ECH
*      WRITE (110,61000) T, X, Q21, Q22, Q34, Q24, Q(2)
*      60000  FORMAT (1X,7D12.4)
*      61000  FORMAT (1X,7D12.4)
*      ENDIF
*      2000  CONTINUE
*     ENDIF
*
*
      XOLD = X
*
      RETURN
      END
*
      SUBROUTINE BNDARY(NPDE,T,U,UX,IBND,BETA,GAMMA,IRES)
*     .. Scalar Arguments ..
      DOUBLE PRECISION  T
      INTEGER           IBND, IRES, NPDE
      DOUBLE PRECISION Ti_EDGE, Te_EDGE, den_EDGE, N_P_EDGE
*     .. Array Arguments ..
      DOUBLE PRECISION  BETA(NPDE), GAMMA(NPDE), U(NPDE), UX(NPDE)
*     .. Common Blocks ..
      COMMON            /BC1/Ti_EDGE, Te_EDGE, den_EDGE, N_P_EDGE
*     .. Executable Statements ..
*     Boundary condition is defined in the form
*              BETA(I)*R(I) = GAMMA(I)
```

```
*     Usually R(I) is proportional to dU(I)/dr so that BC involving
*     df/dr has BETA(I) = 1.0
*     Dirichlet BC is usually set with BETA(I) = 0.0 and
*     GAMMA(I) = U(I) - U_BC
*     IBND = 0 denotes the left boundary point
      IF (IBND.EQ.0) THEN
         BETA(1) = 1.0D0
         BETA(2) = 1.0D0
         BETA(3) = 1.0D0
         BETA(4) = 1.0D0
         GAMMA(1) = 0.0D0
         GAMMA(2) = 0.0D0
         GAMMA(3) = 0.0D0
         GAMMA(4) = 0.0D0
      ELSE
         BETA(1) = 0.0D0
         BETA(2) = 0.0D0
         BETA(3) = 0.0D0
         BETA(4) = 0.0D0
         GAMMA(1) = U(1) - Ti_EDGE
         GAMMA(2) = U(2) - Te_EDGE
         GAMMA(3) = U(3) - den_EDGE
         GAMMA(4) = U(4) - N_P_EDGE
      END IF
      RETURN
      END
```

3B. FORTAN code for symtran4.f

```
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NPDE, NPTS, INTPTS, ITYPE, NEQN, NIW, NWK, NW
      PARAMETER        (NPDE=4,NPTS=201,INTPTS=201,ITYPE=1,
     +                 NEQN=NPDE*NPTS,
     +                 NIW=NEQN+24,NWK=(10+6*NPDE)*NEQN,
     +                 NW=NWK+(21+3*NPDE)*NPDE+7*NPTS+54)
*     .. Scalars in Common ..
      DOUBLE PRECISION N_P, E_ALPHA, Z_EFF, PA, BMAG, MR, R_M,
     +                 AS, S_N, RA, Ti_EDGE, Te_EDGE, den_EDGE,
     +                 Ti_0, Te_0, den_0, PS_I,
     +                 ZL, PHI_E, P_ECH,
     +                 EB, E_ALPHA_AV, C_ETG, L_CRIT,
     +                 EB_0, EB_EDGE, COM_0, COM_MAX,
     +                 TAU_COM, COM_K, COM_BC
*     .. Local Scalars ..
      DOUBLE PRECISION ACC, HX, PI, PIBY2, TOUT, TS
      DOUBLE PRECISION DELTAT, MR, N_P
      DOUBLE PRECISION PHI_I, V1, V2
      DOUBLE PRECISION SIGV1, SIGV2, SIGV3, SIGV4, SIGV5, SIGV6,
     +                 SV, P_ALPHA, PE1, PI1, PE2, QN, QD, P_INP,
     +                 QF, QNT, QDT, N_PX, TAU_II, TAU1, TAU2,
     +                 TAU, EL_I, EL_E, ELX_I, ELX_E,
     +                 R_M0, L_P, COM, COMDOT
      INTEGER          I, IFAIL, IND, IT, ITASK, ITRACE, M
*     .. Local Arrays ..
      DOUBLE PRECISION U(NPDE,NPTS), UOUT(NPDE,INTPTS,ITYPE), W(NW),
     +                 X(NPTS), XOUT(INTPTS)
      INTEGER          IW(NIW)
      INTEGER          J, J1, J2, NSTEP, NWRITE, NCORE, K1
*     .. External Functions ..
      DOUBLE PRECISION X01AAF
      EXTERNAL         X01AAF
*     .. External Subroutines ..
      EXTERNAL         BNDARY, D03PCF, D03PZF, PDEDEF, UINIT
*     .. Intrinsic Functions ..
      INTRINSIC        DSIN, DABS, DLOG, DEXP
************************************************************
************************************************************
*      Root finding variables and parameters
*      * .. Parameters ..
*      * .. Local Scalars ..
       DOUBLE PRECISION Z_A, Z_B, CEPS, ETA, H, Z1, Z2, Z_X
*
```

```
       DOUBLE PRECISION FR
       DOUBLE PRECISION FACT, R_E, R_I, V1, V2, R_D
       INTEGER JFAIL
       EXTERNAL FR
       COMMON  /root/Z1, Z2, PHI_I
*************************************************************
*************************************************************
*     .. Common blocks ..
      COMMON           /Diff/E_ALPHA, Z_EFF, PA, BMAG,
     +                  AS, S_N, RA, ZL, PS_I,
     +                  P_ECH,
     +                  N_P, EB, E_ALPHA_AV,
     +                  C_ETG, L_CRIT, R_M, COM_0, COM_MAX, TAU_COM,
     +                  COM_K
      COMMON           /P1/MR
      COMMON           /BC0/Ti_0, Te_0, den_0, EB_0
      COMMON           /BC1/Ti_EDGE, Te_EDGE, den_EDGE, EB_EDGE
      COMMON           /BC2/COM_BC
      COMMON           /diag/K1
*
*     .. Executable Statements ..
*     Opening a set of output files
*     Running statistics
      OPEN(UNIT=10, FILE='stat.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=20, FILE='U.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=30, FILE='U0.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=40, FILE='Phi.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=50, FILE='Power.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=60, FILE='Power.x.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=70, FILE='Endloss.x.dat',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=100, FILE='diag.dat1',FORM='FORMATTED',
     *     STATUS='OLD')
      OPEN(UNIT=110, FILE='diag.dat2',FORM='FORMATTED',
     *     STATUS='OLD')
*
      WRITE (10,*) 'D03PCF Example Program Results'
*
*     Local Error parameter
      ACC = 1.0D-7
*
*     Selecting spatial geometry
```

```
*     M = 0 (Cartesian),
*     M = 1 (Cylindrical)
*     M = 2 (Spherical)
      M = 1
*
*     Program message control parameter
      ITRACE = 0
*
*     Parameter used in PDE
*
*     Parameter starting the integration in time
      IND = 0
*
*     Normal Computation of output values at t = TOUT
      ITASK = 1
*
*     Various input parameters
*     Ion temperature at edge (r/R = 1) [KeV]
      Ti_EDGE = 7.0D-3
*
*     Electron temperature at edge (r/R = 1) [KeV]
      Te_EDGE = 7.0D-3
*
*     Density at edge (r/R = 1) [1E20/m**3]
      den_EDGE = 2.5D-2
*
*     EB at edge (r/R = 1)
      EB_EDGE = 1.30D0
*
*     Initial ion temperature at origin [KeV]
      Ti_0 = Ti_EDGE
*
*     Initial ion temperature at origin [KeV]
      Te_0 = Te_EDGE
*
*     Initial density at origin [1E20/m**3]
      den_0 = den_EDGE
*
*     Initial EB at origin [KeV]
      EB_0 = EB_EDGE
*
*     Mass ratio between DT nucleus and electron
      MR = 4.583D3
*
*     Initial Plug density [1E20/m^3]
      N_P = 6.0D-2
*
*     Alpha energy [KeV]
```

```
      E_ALPHA = 3.52D3
*
*     Average alpha energy used in the partition of alpha power
*       between ions and electrons
      E_ALPHA_AV = 2.0D3
*
*     Z-effective for Bremstrahlun power
      Z_EFF = 0.0D0
*
*     Initial Mirror ratio
      R_M0 = 1.0D0
*
*     Pashtukhov Constant
      PA = (3.1415927D0)**(0.5D0)/4.0D0*(R_M0+1.0D0)*
     +         DLOG(2.0D0*R_M0+2.0D0)/R_M0
*
*     Radius [m]
      RA = 0.2D0
*
*     Length of reactor [m]
      ZL = 3.0D0
*
*     Length used in compression factor
      L_P = 0.3D0
*
*     Magnetic field [T]
      BMAG = 1.0D0/R_M0
*
*     Ion external power  [KeV*1E20]/[m**3*sec]
      PS_I = 0.0D0
*
*     ECH [KeV*1E20]/[m**3*sec]
      P_ECH = 5.0D1
*
*     Factor to control external power
      AS = 1.0D0
*
*     External density source [1E20 particles/m^3/s]
      S_N = 0.00D0
*
*     Initial value of time-integrated numerator of Q
      QNT = 0.0D0
*
*     Initial value of time-integrated denominator of Q
      QDT = 0.0D0
*
*     Coefficient for the ETG diffusion coefficient
      C_ETG = 0.0D0
```

```
*
*     Critical length used in the ETG diffusion coefficient
      L_CRIT = 1.0D0
*
*     Compression Parameters
*     Initial compression
      COM_0 = 1.0D0
      COM_BC = COM_0
*
*     Maximum compression
      COM_MAX = 1.5D1
*
*     Compression time
      TAU_COM = 1.0D-3
*
*     Compression factor in electron heat equation
      COM_K = 2.0D0*(L_P/ZL)*(N_P/R_M0)
*
*     Set spatial mesh points spacing using sine
*
*     This is Pi/2
      PIBY2 = 0.5D0*X01AAF(PI)
*
*     This is step-size
      HX = PIBY2/(NPTS-1)
*
*     Starting and ending points
      X(1) = 0.0D0
      X(NPTS) = 1.0D0
*
*     Do-loop for mesh points used in PDE
      DO 20 I = 2, NPTS - 1
         X(I) = DSIN(HX*(I-1))
20    CONTINUE
*
*     Creating a set of output mesh points
*
*     Starting and ending points
      XOUT(1) = X(1)
      XOUT(INTPTS) = X(NPTS)
*
*     Do-loop for output points
      DO 1000 J1 = 2, INTPTS - 1
         XOUT(J1) = (J1 - 1)*(X(NPTS) - X(1))/(INTPTS-1)
1000  CONTINUE
*
*     Set initial conditions
*
```

```
*     Starting time
      TS = 0.0D0
*
*     Time step
      DELTAT = 1.0D-6
*
*     Number of time steps
      NSTEP = 8000
*
*     Number of times the spatial profiles are outputed
      NWRITE = 20
*
*     The frequency of printing out core values (at (r/R) = 0)
      NCORE = 10
*
*     Starting number for which a diagnostic file is needed
      K1 = 1
*
*     Running statistics
      WRITE (10,99999) ACC, E_ALPHA
*
*     Set the initial values
      CALL UINIT(U,X,NPTS)
*
*     Writing initial values to file
      WRITE (30,82000) TS, Ti_0, Te_0, den_0, EB_0, COM_0
*
*     This is the main time do-loop
      DO 40 IT = 1, NSTEP
*
*     Priting out error messsage parameter
         IFAIL = -1
*     Definition of the next time-step
         TOUT = IT*DELTAT
*
*     Control of external power input
         IF (TOUT.GT.TAU_COM) THEN
            AS = 0.00D0
         ENDIF
*
*     Defining compression parameter
         IF (TOUT.LE.TAU_COM) THEN
            COM = COM_0 + (COM_MAX - COM_0)/TAU_COM*TOUT
            COMDOT = (COM_MAX - COM_0)/TAU_COM
            COM_BC = COM
         ELSE
            COM = COM_MAX
            COMDOT = 0.0D0
```

```
            COM_BC = COM
         ENDIF
*
*     Defining mirror ratio
         R_M = R_M0*COM
*
*     Pashtukhov Constant
         PA = (3.1415927D0)**(0.5D0)/4.0D0*(R_M+1.0D0)*
     +         DLOG(2.0D0*R_M+2.0D0)/R_M
*
*     Calling the PDE solver subroutine
         CALL D03PCF(NPDE,M,TS,TOUT,PDEDEF,BNDARY,U,NPTS,X,ACC,W,NW,IW,
     +               NIW,ITASK,ITRACE,IND,IFAIL)
*
*        Interpolate at required spatial points
*
         CALL D03PZF(NPDE,M,U,NPTS,X,XOUT,INTPTS,ITYPE,UOUT,IFAIL)
*
*        Writing results to output files
*
         DO 1300 I = 1, NSTEP/NCORE
            IF (IT.EQ.I*NCORE) THEN
            WRITE (30,82000) TOUT, UOUT(1,1,1), UOUT(2,1,1),
     +                       UOUT(3,1,1), UOUT(4,1,1), COM
            ENDIF
1300     CONTINUE
*
         DO 1200 J = 1, NWRITE
         IF (IT.EQ.J*NSTEP/NWRITE) THEN
*
*        Initializing the endloss spatial integration
         ELX_I = 0.0D0
         ELX_E = 0.0D0
*
*        Time
            WRITE (20,80000) TOUT
*           WRITE (40,80000) TOUT
*           WRITE (70,80000) TOUT
*
*        Spatial values
         DO 1100 J1 = 1, INTPTS
            IF (DABS(UOUT(1,J1,1)).LT.1.0D-10) THEN
                UOUT(1,J1,1) = 0.0D0
            ENDIF
            IF (DABS(UOUT(2,J1,1)).LT.1.0D-10) THEN
                UOUT(2,J1,1) = 0.0D0
            ENDIF
            IF (DABS(UOUT(3,J1,1)).LT.1.0D-10) THEN
```

```
              UOUT(3,J1,1) = 0.0D0
            ENDIF
            IF (DABS(UOUT(4,J1,1)).LT.1.0D-10) THEN
              UOUT(4,J1,1) = 0.0D0
            ENDIF
*
*        End program early when physical quantities become negative
            IF (UOUT(1,J1,1).LT.0.0D0.OR.UOUT(2,J1,1).LT.0.0D0.OR.
     +          UOUT(3,J1,1).LT.0.0D0.OR.UOUT(4,J1,1).LT.0.0D0) THEN
              IT = NSTEP
            ENDIF
*
*           Printing out Ti, Te, den
            WRITE (20,81000) XOUT(J1), UOUT(1,J1,1), UOUT(2,J1,1),
     +                       UOUT(3,J1,1), UOUT(4,J1,1)
*
*           Checking whether Phi_e and Phi_i satisfy condition
              Z1 = UOUT(1,J1,1)
              Z2 = UOUT(2,J1,1)
*
*
              N_PX = N_P*COM
*
            IF (N_PX.GT.UOUT(3,J1,1)) THEN
              PHI_I = Z2*DLOG(N_PX/UOUT(3,J1,1))
            ELSE
              PHI_I = 0.0D0
            ENDIF
*
*           Step size for root search
              H  = 1.0D-1
*           Error tolerance
              CEPS = 1.0D-7
*           Closeness of FR to 0.0
              ETA = 0.0D0
*           Root finding program progress monitor parameter
              JFAIL = 1
*           Factor used in initial root estimate
              FACT = 1.0D0
*           Initial estimate for root
              Z_X = PHI_I*(Z2/Z1)
*
            CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
*
*           Checking if the root Z_X is correct
              R_E = Z_X/Z2
              R_I = PHI_I/Z1
              V1 = DLOG(R_E) + R_E
```

```
              V2 = 0.50D0*DLOG(MR)
     +            + 1.5D0*DLOG(Z1/Z2)
     +            + DLOG(R_I) + R_I
              R_D = DABS((V2-V1)/V1)
*
              IF (Z_X.LE.0.0D0.OR.R_D.GT.1.0D-2) THEN
               FACT = FACT + 1.0D0
               H  = H + 10.0D0
               Z_X = PHI_I*(Z2/Z1)
               CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
               R_E = Z_X/Z2
               R_I = PHI_I/Z1
               V1 = DLOG(R_E) + R_E
               V2 = 0.50D0*DLOG(MR)
     +             + 1.5D0*DLOG(Z1/Z2)
     +             + DLOG(R_I) + R_I
               R_D = DABS((V2-V1)/V1)
              ENDIF
*
              PHI_E = Z_X
*
*           WRITE (40,83000) XOUT(J1), N_PX,PHI_I,PHI_E,R_D
*
*        Writing End-loss terms to file
*        Characteristics times
         TAU_II =  1.6D16*UOUT(1,J1,1)**(1.5D0)/(UOUT(3,J1,1)*1.0D20)
         IF (PHI_I/UOUT(1,J1,1).LT.1.0D2) THEN
          TAU1 = TAU_II*PA*PHI_I/UOUT(1,J1,1)*DEXP(PHI_I/UOUT(1,J1,1))
         ELSE
          TAU1 = 1.0D43
         ENDIF
         TAU2 = 3.6D-6*R_M*ZL/UOUT(1,J1,1)**(0.5D0)
     +              *DEXP(PHI_I/UOUT(1,J1,1))
         TAU = TAU1 + TAU2
*
*        End-loss terms
         EL_I = UOUT(3,J1,1)*(PHI_I + UOUT(1,J1,1))/TAU
         EL_E = UOUT(3,J1,1)*(PHI_E + UOUT(2,J1,1))/TAU
         IF (J1.LT.INTPTS) THEN
           ELX_I = ELX_I + EL_I*XOUT(J1+1)*(XOUT(J1+1)-XOUT(J1))
           ELX_E = ELX_E + EL_E*XOUT(J1+1)*(XOUT(J1+1)-XOUT(J1))
         ENDIF
*
*        WRITE (70,86000) XOUT(J1), EL_I, EL_E, ELX_I, ELX_E
*
1100     CONTINUE
*
         WRITE (20,*)
```

```
*        WRITE (40,*)
*        WRITE (70,*)
         ENDIF
1200     CONTINUE
*
*        Calculating Q
*
*        Starting value for numerator of Q
         QN = 0.0D0
*
*        Starting value for denominator of W
         QD = 0.0D0
*
*        Doing the spatial and time integration of various powers
         DO 1400 J2 = 1, INTPTS-1
*        <Sigma*Velocity>_fusion reaction
         IF (UOUT(1,J2,1).GE.1.0D0.AND.UOUT(1,J2,1).LE.8.0D1) THEN
            SIGV1 = (-2.138D1)*UOUT(1,J2,1)**(-2.935D-1)
            SIGV2 = -2.520D1
            SIGV3 =  (-7.101D-2)*UOUT(1,J2,1)
            SIGV4 =  (1.938D-4)*UOUT(1,J2,1)**2.0D0
            SIGV5 =  (4.925D-6)*UOUT(1,J2,1)**3.0D0
            SIGV6 =  (-3.984D-8)*UOUT(1,J2,1)**4.0D0
            SV = (1.0D-6)*DEXP(SIGV1+SIGV2+SIGV3+SIGV4+SIGV5+SIGV6)
         ELSE
            SV = 0.0D0
         ENDIF
*
*        Alpha Power
         P_ALPHA = 0.25D20*UOUT(3,J2,1)**(2.0D0)*SV*E_ALPHA
*
*        Input Powers
           PE1 = AS*P_ECH
           PI1 = AS*PS_I
*
*        Integration of Alpha Power
         QN = QN + 5.0D0*(XOUT(J2+1)-XOUT(J2))*XOUT(J2+1)*P_ALPHA
*
*        Integration of input power
         P_INP = PE1 + PE2 + PI1
         QD = QD + (XOUT(J2+1)-XOUT(J2))*XOUT(J2+1)*P_INP
*
*        Writing to output file
*        IF (J2.EQ.1) THEN
*         WRITE (60,85000) TOUT, XOUT(J2), P_ALPHA, P_INP, QN, QD
*         WRITE (60,85000) TOUT, XOUT(J2), P_ALPHA
*        ENDIF
*
```

```
*
1400     CONTINUE
*
         QNT = QNT + QN*DELTAT
         QDT = QDT + QD*DELTAT
         QF = QNT/QDT
*
*        DO 1500 I = 1, NSTEP/NCORE
*           IF (IT.EQ.I*NCORE) THEN
*           WRITE (50,84000) TOUT, QN, QNT, QD, QDT, QN/(QD+1.0D-5)
*           ENDIF
*        1500     CONTINUE
*
*
         IF (IFAIL.EQ.3) THEN
           WRITE (10,90000) 'IFAIL IS', IFAIL, TOUT
         ENDIF
*
40    CONTINUE
*
*     Print integration statistics
*
      WRITE (10,99997) IW(1), IW(2), IW(3), IW(5)
      STOP
*
99999 FORMAT (//' Accuracy requirement  = ',D12.5,/' Parameter E_AL =',
     +       '       ',D12.3,/)
99997 FORMAT (' Number of integration steps in time                  ',
     +       I4,/' Number of residual evaluations of resulting ODE sys',
     +       'tem',I4,/' Number of Jacobian evaluations               ',
     +       '         ',I4,/' Number of iterations of nonlinear solve',
     +       'r              ',I4,/)
90000 FORMAT (1X,A,I1,D12.4)
80000 FORMAT (1X,D12.4)
81000 FORMAT (1X,5D12.4)
82000 FORMAT (1X,7D12.4)
*83000 FORMAT (1X,5D12.4,F7.2)
83000 FORMAT (1X,6D12.4)
84000 FORMAT (1X,6D12.4)
85000 FORMAT (1X,6D12.4)
86000 FORMAT (1X,5D12.4)
      END
*
      SUBROUTINE UINIT(U,X,NPTS)
*     Routine for PDE initial conditon
*     .. Scalar Arguments ..
      INTEGER          NPTS
*     .. Array Arguments ..
```

```
      DOUBLE PRECISION U(4,NPTS), X(NPTS)
*     .. Scalars in Common ..
      DOUBLE PRECISION Ti_EDGE, Te_EDGE, den_EDGE, EB_EDGE,
     +                 Ti_0, Te_0, den_0, EB_0
*     .. Local Scalars ..
      INTEGER          I
*     .. Common blocks ..
      COMMON           /BC0/Ti_0, Te_0, den_0, EB_0
      COMMON           /BC1/Ti_EDGE, Te_EDGE, den_EDGE, EB_EDGE
*     .. Intrinsic functions ..
      INTRINSIC        DLOG, DEXP
*     .. Executable Statements ..
      WRITE (10,*)
      WRITE (10,*) 'Checking root finder'
      WRITE (10,*)
*
      DO 20 I = 1, NPTS
*
         U(1,I) = (Ti_0-Ti_EDGE)*(1.0D0-X(I)**2.0D0)+Ti_EDGE
         U(2,I) = (Te_0-Te_EDGE)*(1.0D0-X(I)**2.0D0)+Te_EDGE
*        U(3,I) = (den_0)*(1.0D0-X(I)**2.0D0)+den_EDGE
         U(3,I) = (den_0-den_EDGE)*(1.0D0-X(I)**2.0D0)+den_EDGE
         U(4,I) = (EB_0-EB_EDGE)*(1.0D0-X(I)**2.0D0)+EB_EDGE
*
*     Printing out the results
         WRITE (10,70000) X(I), U(1,I), U(2,I), U(3,I), U(4,I)
*
20    CONTINUE
         WRITE (10,*)
70000    FORMAT (1X,5D12.4)
      RETURN
      END
*
*
       DOUBLE PRECISION FUNCTION FR(Z_X)
*      * .. Scalar Arguments ..
       DOUBLE PRECISION Z_X, Z1, Z2, PHI_I, MR
*      * .. Intrinsic Functions ..
       INTRINSIC DEXP, DLOG, DABS
*
       COMMON /P1/MR
       COMMON /root/Z1, Z2, PHI_I
*
*      * .. Executable Statements ..
*
       FR = Z_X/Z2*DEXP(Z_X/Z2) - MR**(0.5D0)*(Z1/Z2)**(1.5D0)
     #            *(PHI_I/Z1)*DEXP(PHI_I/Z1)
       RETURN
```

```
       END
*
*
      SUBROUTINE PDEDEF(NPDE,T,X,U,DUDX,P,Q,R,IRES)
*     .. Scalar Arguments ..
      DOUBLE PRECISION  T, X
      INTEGER           IRES, NPDE
      INTEGER           K2
*     .. Array Arguments ..
      DOUBLE PRECISION  DUDX(NPDE), P(NPDE,NPDE), Q(NPDE), R(NPDE),
     +                  U(NPDE)
      DOUBLE PRECISION  PHI_I, SIGV1, SIGV2, SIGV3, SIGV4, SIGV5,
     +                  SIGV6, SV, P_ALPHA, P_BREM, TAU_II, TAU_IE,
     +                  TAU, R_LI, CHI_I, CHI_E, D_N, Q11, Q12, Q21,
     +                  Q22, Q23, Q24, Q25, Q31, PHI_E, Q13,
     +                  Q26, Q27, N_PX, U30, XOLD, TAU_A, F_A_I,
     +                  Q14, CHI_E_ETG, Q41, TAU_P, COM, COMDOT,
     +                  Q28
*     .. Scalars in Common ..
      DOUBLE PRECISION  N_P, E_ALPHA, Z_EFF, PA, BMAG, MR,
     +                  AS, S_N, RA, ZL, PS_I,
     +                  P_ECH,
     +                  EB, E_ALPHA_AV, C_ETG, L_CRIT, R_M,
     +                  COM_0, COM_MAX, TAU_COM, COM_K
      INTEGER           K1
*     .. Intrinsic Functions ..
      INTRINSIC        DABS, DLOG, DEXP
****************************************************************
****************************************************************
*      Root finding variables and parameters
*      * .. Parameters ..
*      * .. Local Scalars ..
       DOUBLE PRECISION Z_A, Z_B, CEPS, ETA, H, Z1, Z2, Z_X
*
       DOUBLE PRECISION FR
       DOUBLE PRECISION FACT, R_E, R_I, V1, V2, R_D
       INTEGER JFAIL
       EXTERNAL FR
       COMMON  /root/Z1, Z2, PHI_I
****************************************************************
****************************************************************
*     .. Common blocks ..
      COMMON           /Diff/E_ALPHA, Z_EFF, PA, BMAG,
     +                  AS, S_N, RA, ZL, PS_I,
     +                  P_ECH,
     +                  N_P, EB, E_ALPHA_AV,
     +                  C_ETG, L_CRIT, R_M, COM_0, COM_MAX, TAU_COM,
     +                  COM_K
```

```
      COMMON           /P1/MR
      COMMON           /diag/K1
*     .. Executable Statements ..
*     Variables and Parameters
*     U(1) = Ti(x,t), U(2) = Te(x,t), U(3) = n(x,t), U(4) = EB
*
*     Preventing trouble when quantities become negative
      IF (U(1).LE.0.0D0) THEN
         U(1) = 1.0D-8
      ENDIF
*
      IF (U(2).LE.0.0D0) THEN
         U(2) = 1.0D-8
      ENDIF
*
      IF (U(3).LE.0.0D0) THEN
         U(3) = 1.0D-8
      ENDIF
*
      IF (U(4).LE.0.0D0) THEN
         U(4) = 1.0D-8
      ENDIF
*
*     Ion potential
*
*     Finding density at (r/R) = 0
      IF (X.LT.XOLD) THEN
        U30 = U(3)
      ENDIF
*
*     Defining compression ratio
      IF (T.LE.TAU_COM) THEN
        COM = COM_0 + (COM_MAX - COM_0)/TAU_COM*T
        COMDOT = (COM_MAX - COM_0)/TAU_COM
      ELSE
        COM = COM_MAX
        COMDOT = 0.0D0
      ENDIF
*
*     Definition of the plug density
      N_PX = N_P*COM
*
      IF (N_PX.GT.U(3)) THEN
        PHI_I = U(2)*DLOG(N_PX/U(3))
      ELSE
        PHI_I = 0.0D0
      ENDIF
*
```

```
*     Electron potential
         Z1 = U(1)
         Z2 = U(2)
*     Step size for root search
         H  = 1.0D-1
*     Error tolerance
         CEPS = 1.0D-7
*     Closeness of FR to 0.0
         ETA = 0.0D0
*     Root finding program progress monitor parameter
         JFAIL = 1
*     Factor used in initial root estimate
         FACT = 1.0D0
*     Initial estimate for root
         Z_X = PHI_I*(Z2/Z1)
*
      CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
*
*     Checking if the root Z_X is correct
         R_E = Z_X/U(2)
         R_I = PHI_I/U(1)
         V1 = DLOG(R_E) + R_E
         V2 = 0.50D0*DLOG(MR)
     +       + 1.5D0*DLOG(Z1/Z2)
     +       + DLOG(R_I) + R_I
         R_D = DABS((V2-V1)/V1)
*
         IF (Z_X.LE.0.0D0.OR.R_D.GT.1.0D-2) THEN
          FACT = FACT + 1.0D0
          H  = H + 10.0D0
          Z_X = PHI_I*(Z2/Z1)
          CALL C05AGF(Z_X,H,CEPS,ETA,FR,Z_A,Z_B,JFAIL)
          R_E = Z_X/U(2)
          R_I = PHI_I/U(1)
          V1 = DLOG(R_E) + R_E
          V2 = 0.50D0*DLOG(MR)
     +        + 1.5D0*DLOG(Z1/Z2)
     +        + DLOG(R_I) + R_I
          R_D = DABS((V2-V1)/V1)
         ENDIF
*
         PHI_E = Z_X
*
*     <Sigma*Velocity>_fusion reaction
      IF (U(1).GE.1.0D0.AND.U(1).LE.8.0D1) THEN
        SIGV1 = (-2.138D1)*U(1)**(-2.935D-1)
        SIGV2 = -2.520D1
        SIGV3 =  (-7.101D-2)*U(1)
```

```
        SIGV4 =  (1.938D-4)*U(1)**2.0D0
        SIGV5 =  (4.925D-6)*U(1)**3.0D0
        SIGV6 =  (-3.984D-8)*U(1)**4.0D0
        SV = (1.0D-6)*DEXP(SIGV1+SIGV2+SIGV3+SIGV4+SIGV5+SIGV6)
      ELSE
        SV = 0.0D0
      ENDIF
*
*     Alpha Power
      P_ALPHA = 0.25D20*U(3)**(2.0D0)*SV*E_ALPHA
*
*     Bremstrahlung
      P_BREM =  0.3125D0*Z_EFF*U(3)**(2.0D0)*U(2)**(0.5D0)
*
*     Characteristic Times
      TAU_II =  1.6D16*U(1)**(1.5D0)/(U(3)*1.0D20)
      TAU_IE =  1.0D18*U(2)**(1.5D0)/(U(3)*1.0D20)
      TAU_P = U(3)*TAU_IE*(1.0D0 - 5.0D0*U(2)/U(4))/N_PX
*     TAU_P = 1.0D9
*
      IF (PHI_I/U(1).LT.1.0D2) THEN
        TAU = TAU_II*PA*PHI_I/U(1)*DEXP(PHI_I/U(1))
     +      + 3.6D-6*R_M*ZL/U(1)**(0.5D0)*DEXP(PHI_I/U(1))
      ELSE
        TAU = 1.0D43
      ENDIF
*
*     Fraction of alpha power going to the ions
      TAU_A = 1.6D16*E_ALPHA_AV**(1.5D0)/(U(3)*1.0D20)
*
      F_A_I = TAU_IE*(E_ALPHA_AV - U(1))
     +       /(TAU_IE*(E_ALPHA_AV - U(1))
     +           + TAU_A*(E_ALPHA_AV - U(2)))
*     F_A_I = 0.0D0
*
*     Ion gyro-radius
      R_LI =  6.4D-3*U(1)**(0.5D0)/BMAG
*
*     Diffusion coefficients
      CHI_I = R_LI**(2.0D0)/TAU_II
      CHI_E = MR**(-0.5D0)*(U(1)/U(2))**(0.5D0)*CHI_I
*
      CHI_E_ETG = C_ETG*U(2)**(1.5D0)/BMAG**(2.0D0)
     +          *(DABS(DUDX(2))/U(2) - 1.0D0/L_CRIT)
      IF (CHI_E_ETG.LE.0.0D0) THEN
        CHI_E_ETG = 0.0D0
      ENDIF
*
```

```
      D_N = CHI_E
*
*     Coefficients needed in the PDE solver
*
*     U(1) - T_i
*     Coefficients for time-derivatives
      P(1,1) = 1.5D0*U(3)
      P(1,2) = 0.0D0
      P(1,3) = 1.5D0*U(1)
      P(1,4) = 0.0D0
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q11 = U(3)*(U(2)-U(1))/TAU_IE
      Q12 = -U(3)*(PHI_I+U(1))/TAU
*
      Q13 = AS*PS_I
*
*     Alpha heating of the ions
      Q14 = P_ALPHA*F_A_I
*
      Q(1) = -(Q11+Q12+Q13+Q14)
*
*     Coefficients for space-derivatives
      R(1) = U(3)*CHI_I/RA**(2.0D0)*DUDX(1)
*
*
*     U(2) - T_e
*     Coefficients for time-derivatives
      P(2,1) = 0.0D0
      P(2,2) = 1.5D0*U(3) + 1.5D0*COM_K
      P(2,3) = 1.5D0*U(2)
      P(2,4) = 0.0D0
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q21 = -U(3)*(U(2)-U(1))/TAU_IE
*
      Q22 = -U(3)*(5.0D0*U(2)+U(2))/TAU
*
      Q23 = P_ALPHA*(1.0D0 - F_A_I)
*
      Q24 = COM_K*N_PX/(U(3)*TAU_IE)*U(4)
*
      Q25 = -P_BREM
*
      Q26 = AS*P_ECH
*
      Q27 = 1.5D0*COM_K*U(2)*COMDOT/COM
*
```

```
      Q28 = -COM_K*(5.0D0*U(2)+U(2))/TAU_P
*
      Q(2) = -(Q21+Q22+Q23+Q24+Q25+Q26+Q27+Q28)
*
*     Coefficients for space-derivatives
      R(2) = U(3)*(CHI_E+CHI_E_ETG)/RA**(2.0D0)*DUDX(2)
*
*
*     U(3) - n
*     Coefficients for time-derivatives
      P(3,1) = 0.0D0
      P(3,2) = 0.0D0
      P(3,3) = 1.0D0
      P(3,4) = 0.0D0
*
*     Coefficients for sources (Q>0) and sinks (Q<0)
      Q31 = S_N
*
      Q(3) = -(Q31)
*
      R(3) = 0.0D0
*
*
*     U(4) - EB
*     Coefficients for time-derivatives
      P(4,1) = 0.0D0
      P(4,2) = 0.0D0
      P(4,3) = 0.0D0
      P(4,4) = 1.0D0
*
*     Coefficients for sources (Q>0) and sinks(Q<0)
      Q41 = U(4)*COMDOT/COM
*
      Q42 = -U(4)/TAU_P

      Q(4) = -(Q41+Q42)
*
*     Coefficients for space-derivatives
      R(4) = 0.0D0
*
*     Printing out diagnostic files if needed
*     IF (T.GE.0.0D0) THEN
*     K1 = K1 + 1
*     DO 2000 K2 = 1, 10
*      IF (K2.EQ.K1/1000) THEN
*      WRITE (100,60000) T,X,P(2,2),N_PX,COM,Q24,Q27
*      WRITE (110,61000) T,X,COM,U(4),Q41,Q(4),R(4)
*      WRITE (110,61000) T,X,U(2),U(4),TAU_P,Q28,Q42
```

```
*      60000  FORMAT (1X,7D12.4)
*      61000  FORMAT (1X,7D12.4)
*      ENDIF
*      2000  CONTINUE
*     ENDIF
*
*
      XOLD = X
*
      RETURN
      END
*
      SUBROUTINE BNDARY(NPDE,T,U,UX,IBND,BETA,GAMMA,IRES)
*     .. Scalar Arguments ..
      DOUBLE PRECISION  T
      INTEGER           IBND, IRES, NPDE
      DOUBLE PRECISION Ti_EDGE, Te_EDGE, den_EDGE, EB_EDGE,
     +                 COM_BC
*     .. Array Arguments ..
      DOUBLE PRECISION  BETA(NPDE), GAMMA(NPDE), U(NPDE), UX(NPDE)
*     .. Common Blocks ..
      COMMON            /BC1/Ti_EDGE, Te_EDGE, den_EDGE, EB_EDGE
      COMMON            /BC2/COM_BC
*     .. Executable Statements ..
*     Boundary condition is defined in the form
*              BETA(I)*R(I) = GAMMA(I)
*     Usually R(I) is proportional to dU(I)/dr so that BC involving
*     df/dr has BETA(I) = 1.0
*     Dirichlet BC is usually set with BETA(I) = 0.0 and
*     GAMMA(I) = U(I) - U_BC
*     IBND = 0 denotes the left boundary point
      IF (IBND.EQ.0) THEN
         BETA(1) = 1.0D0
         BETA(2) = 1.0D0
         BETA(3) = 1.0D0
         BETA(4) = 1.0D0
         GAMMA(1) = 0.0D0
         GAMMA(2) = 0.0D0
         GAMMA(3) = 0.0D0
         GAMMA(4) = 0.0D0
      ELSE
         BETA(1) = 0.0D0
         BETA(2) = 0.0D0
         BETA(3) = 0.0D0
         BETA(4) = 0.0D0
         GAMMA(1) = U(1) - Ti_EDGE
         GAMMA(2) = U(2) - Te_EDGE
         GAMMA(3) = U(3) - den_EDGE
```

```
   GAMMA(4) = U(4) - COM_BC*EB_EDGE
END IF
RETURN
END
```

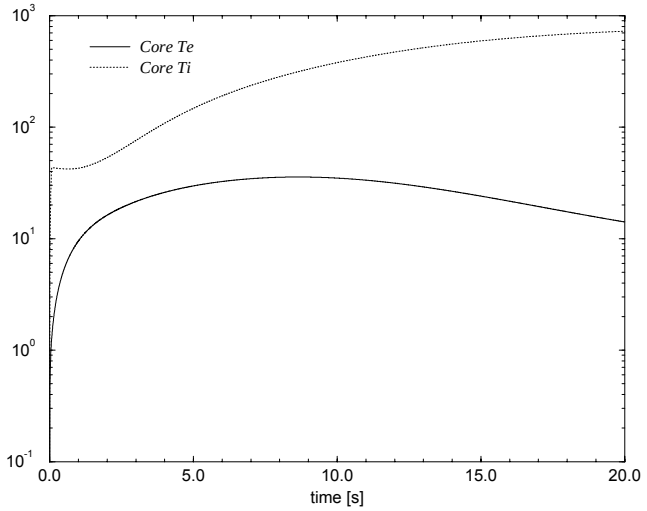

Figure 1: Run 1 - T_i and T_e at $(r/R) = 0$ versus time.

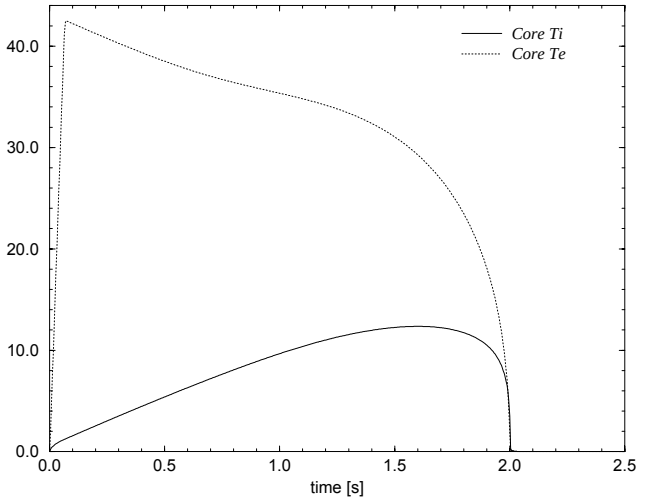

Figure 2: Run 2 - T_i and T_e at $(r/R) = 0$ versus time.

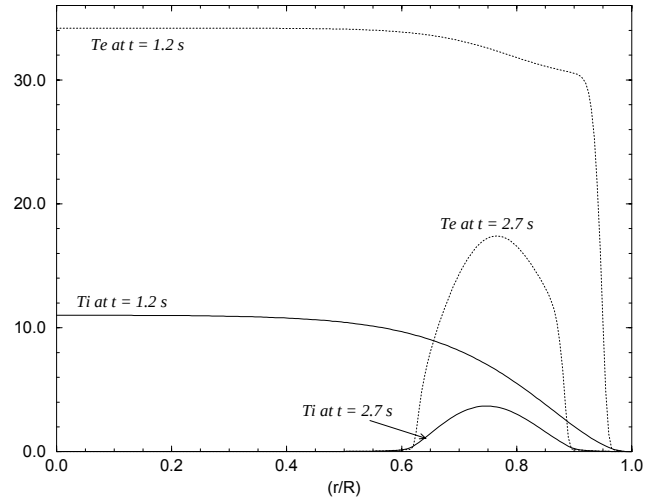

Figure 3: Run 2 - T_i and T_e versus (r/R) at $t = 1.2$ and 2.7 s.

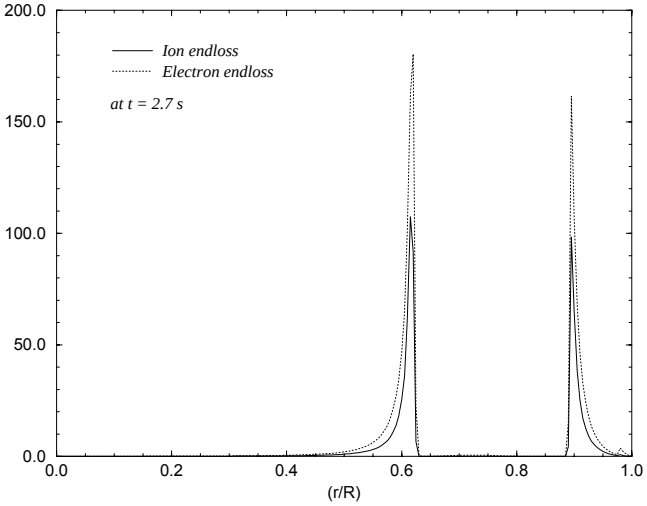

Figure 4: Run 2 - ion endloss, $n_c(\phi_i + T_i)/\tau$ and electron endloss, $n_c(\phi_e + T_e)/\tau$ versus (r/R) at $t = 2.7$ s.

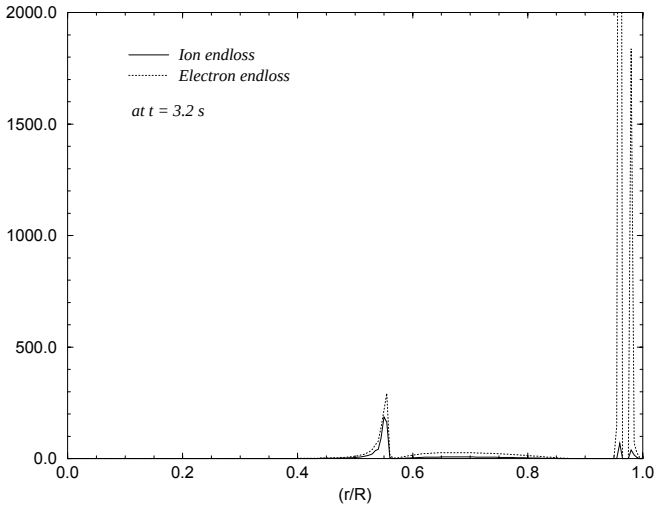

Figure 5: Run 2 - (extra $P_{ECH} = 200$ applied from $0.95 < (r/R) < 0.98$) ion endloss, $n_c(\phi_i + T_i)/\tau$ and electron endloss, $n_c(\phi_e + T_e)/\tau$ versus (r/R) at $t = 3.2$ s.

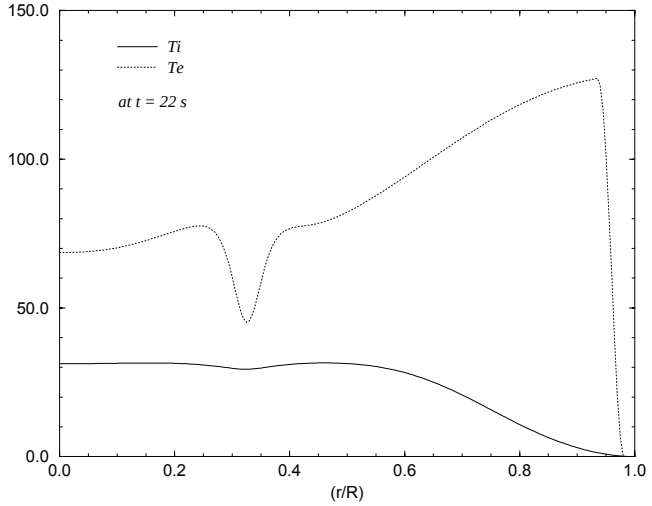

Figure 6: Run 3 - T_i and T_e versus (r/R) at $t = 22$ s.

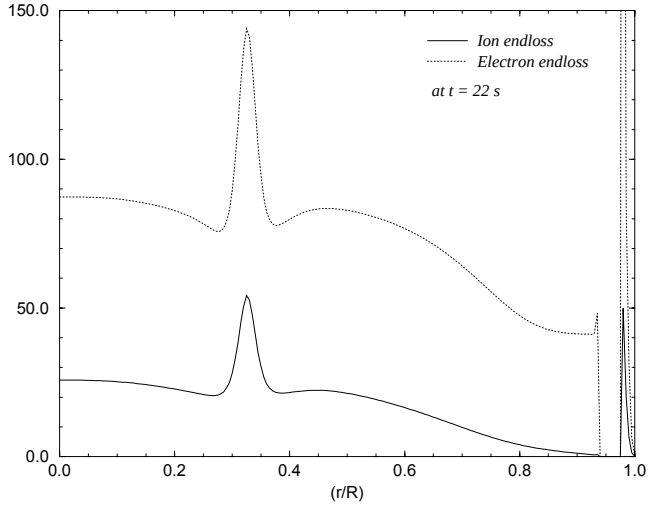

Figure 7: Run 3 - ion endloss, $n_c(\phi_i + T_i)/\tau$ and electron endloss, $n_c(\phi_e + T_e)/\tau$ versus (r/R) at $t = 22$ s.

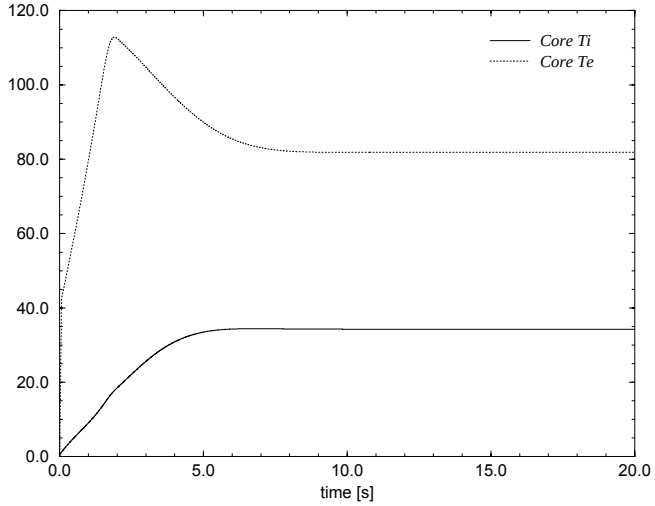

Figure 8: Run 4 - T_i and T_e at $(r/R) = 0$ versus time.

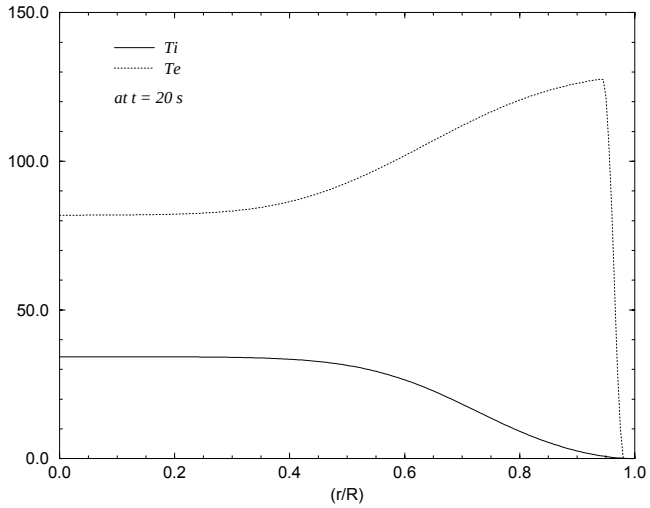

Figure 9: Run 4 - T_i and T_e versus (r/R) at and $t = 20$ s.

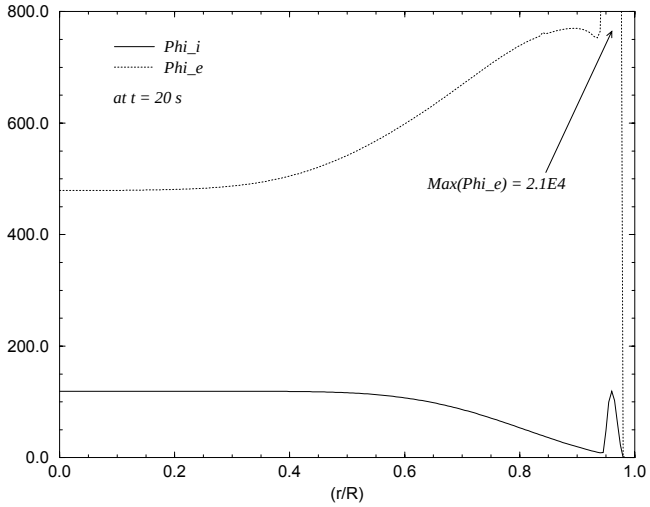

Figure 10: Run 4 - ϕ_i and ϕ_e versus (r/R) at $t = 20$ s.

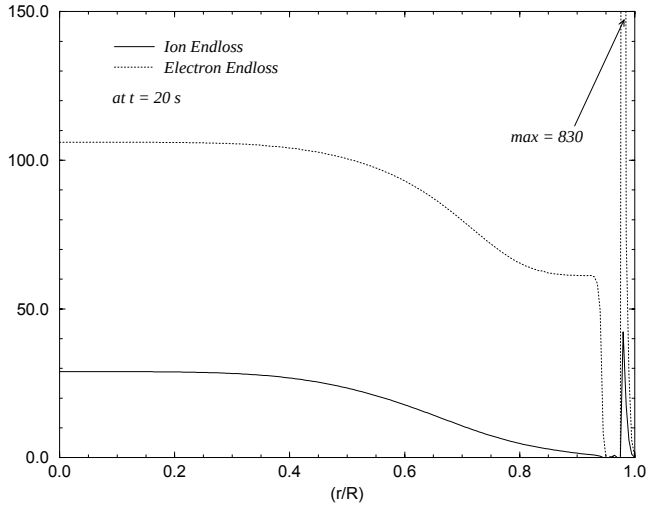

Figure 11: Run 4 - ion endloss, $n_c(\phi_i + T_i)/\tau$ and electron endloss, $n_c(\phi_e + T_e)/\tau$ versus (r/R) at $t = 20$ s.

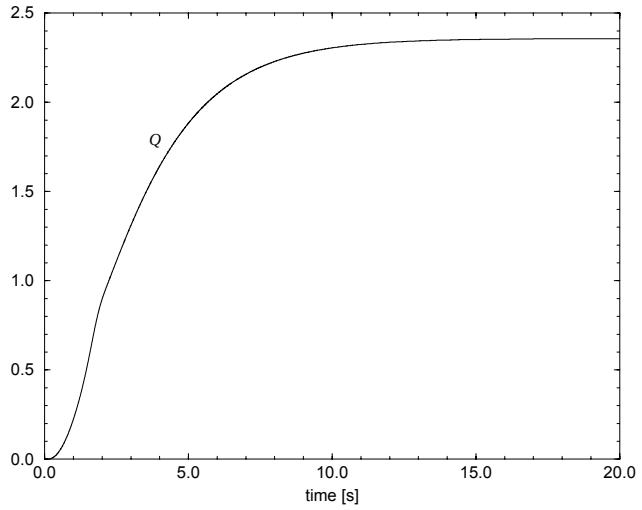

Figure 12: Run 4 - Q versus time.

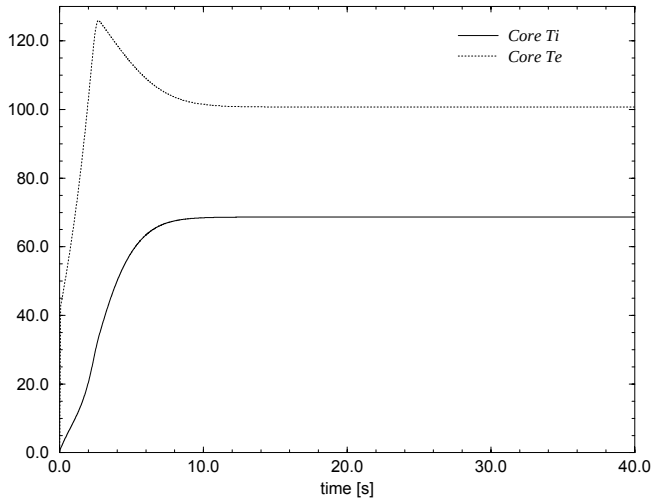

Figure 13: Run 5 - T_i and T_e at $(r/R) = 0$ versus time.

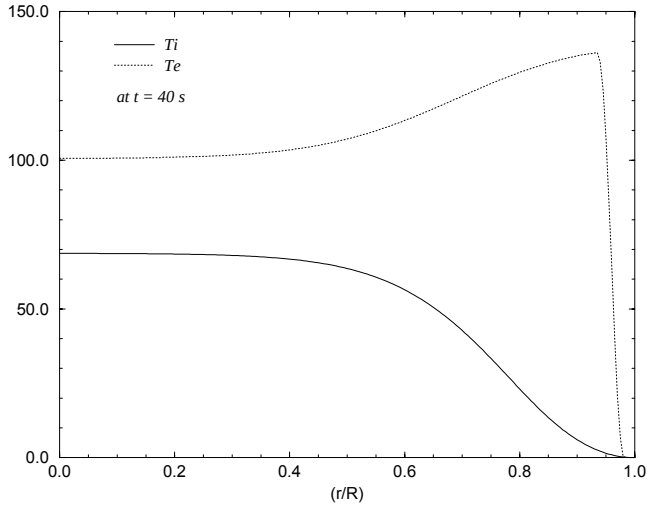

Figure 14: Run 5 - T_i and T_e versus (r/R) at $t = 40$ s.

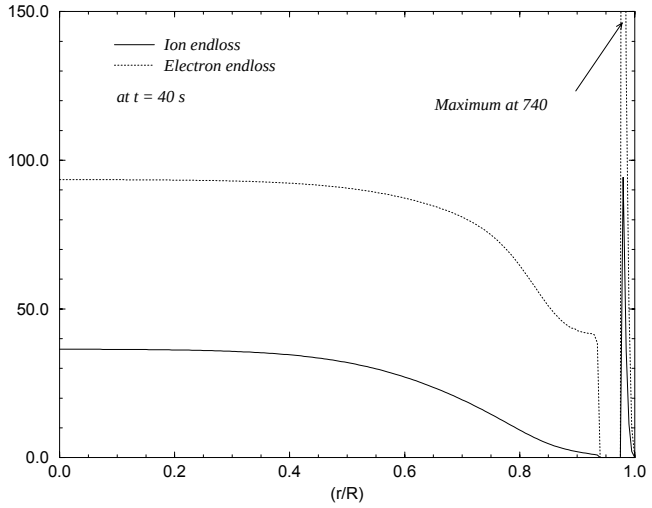

Figure 15: Run 5 - ion endloss, $n_c(\phi_i + T_i)/\tau$ and electron endloss, $n_c(\phi_e + T_e)/\tau$ versus (r/R) at $t = 40$ s.

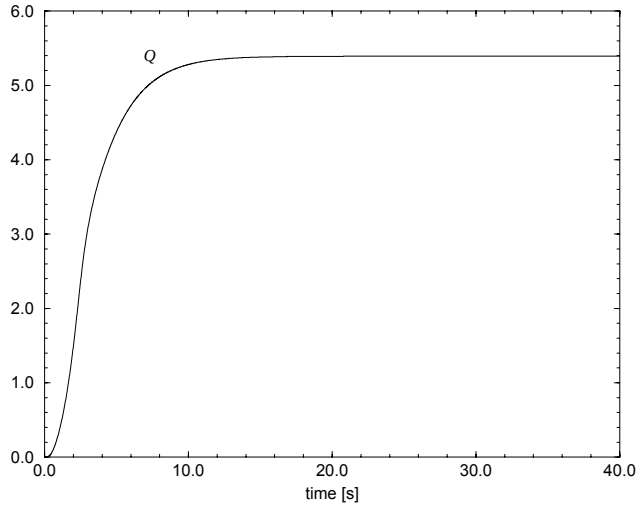

Figure 16: Run 5 - Q versus time.

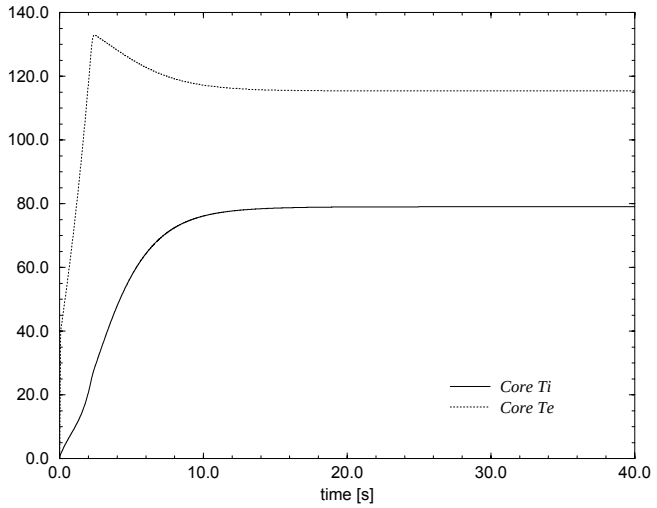

Figure 17: Run 6 - T_i and T_e at $(r/R) = 0$ versus time.

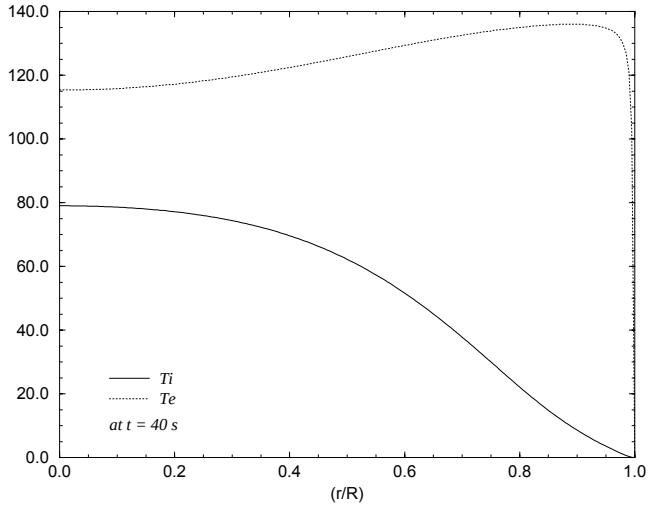

Figure 18: Run 6 - T_i and T_e versus (r/R) at $t = 40$ s.

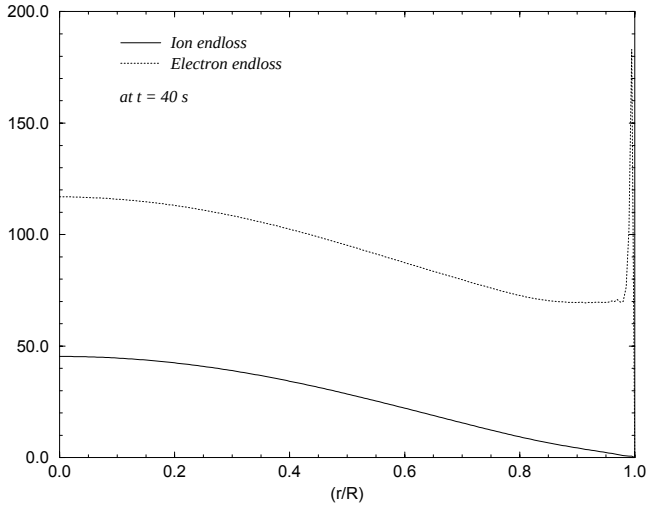

Figure 19: Run 6 - ion endloss, $n_c(\phi_i + T_i)/\tau$ and electron endloss, $n_c(\phi_e + T_e)/\tau$ versus (r/R) at $t = 40$ s.

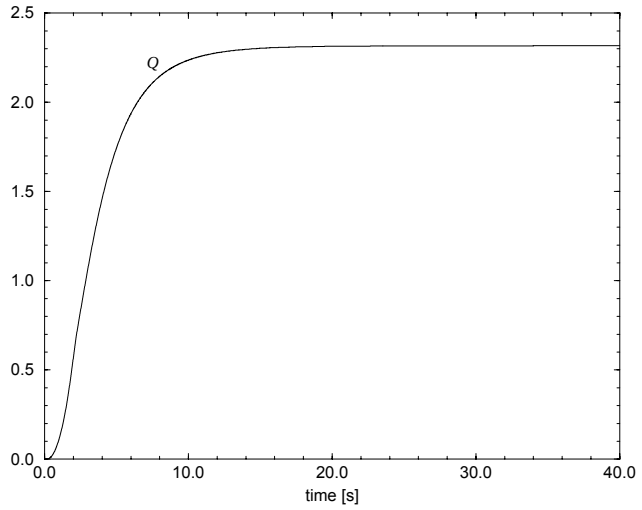

Figure 20: Run 6 - Q versus time.

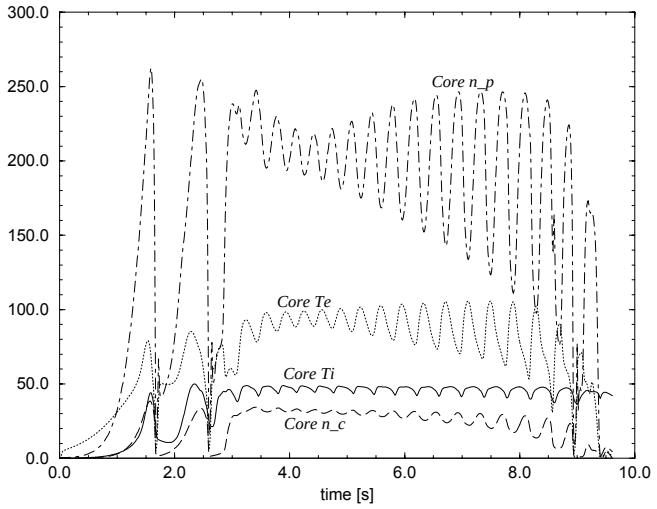

Figure 21: Run 7 - T_i, T_e, n_c and n_p at $(r/R) = 0$ versus time.

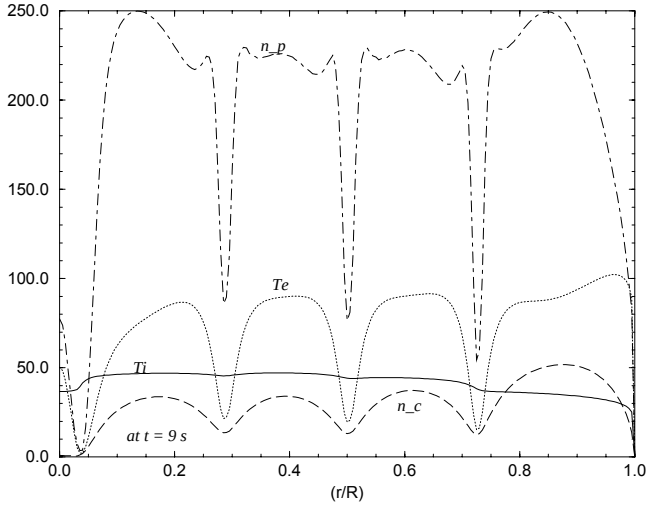

Figure 22: Run 7 - T_i, T_e, n_c and n_p versus (r/R) at $t = 9$ s.

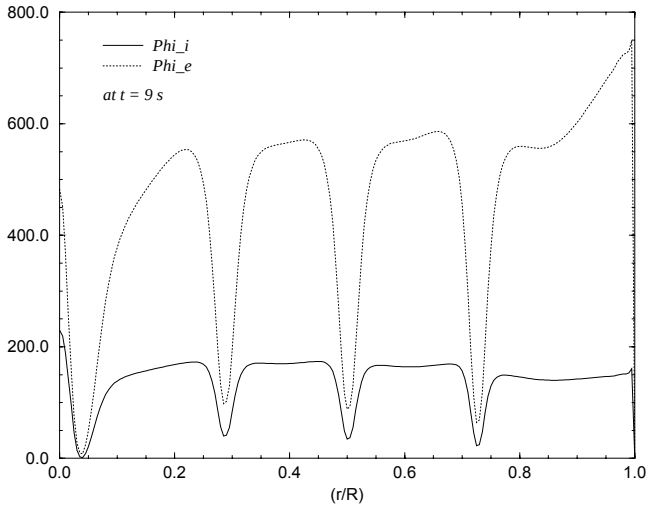

Figure 23: Run 7 - ϕ_i and ϕ_e versus (r/R) at $t = 9$ s.

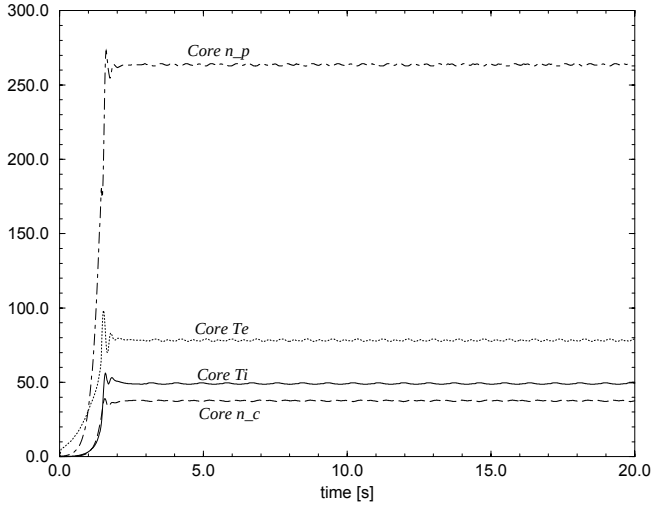

Figure 24: Run 8 - T_i, T_e, n_c and n_p at $(r/R) = 0$ versus time.

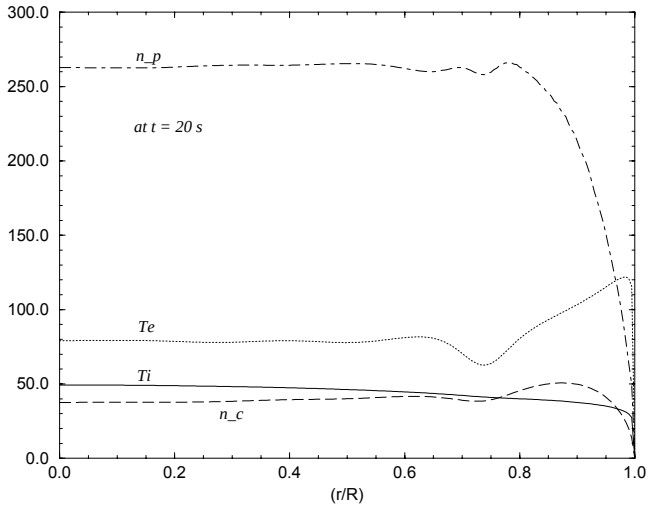

Figure 25: Run 8 - T_i, T_e, n_c and n_p versus (r/R) at $t = 20$ s.

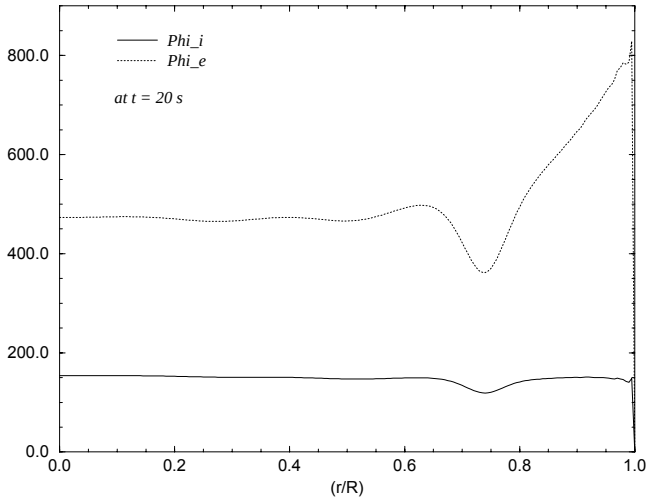

Figure 26: Run 8 - ϕ_i and ϕ_e versus (r/R) at $t = 20$ s.

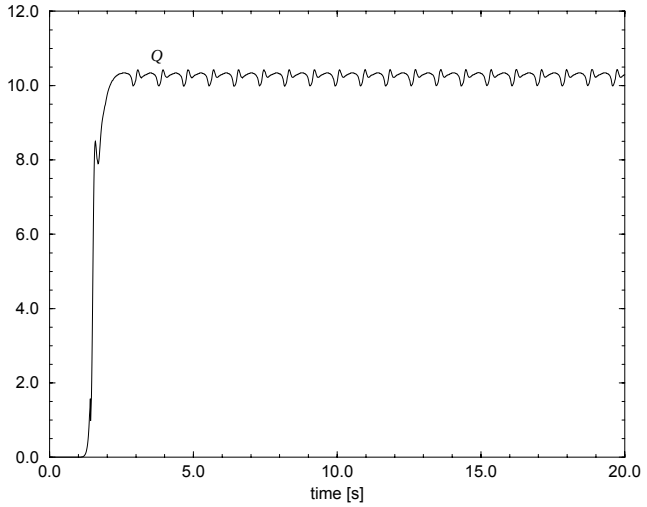

Figure 27: Run 8 - Q versus time.

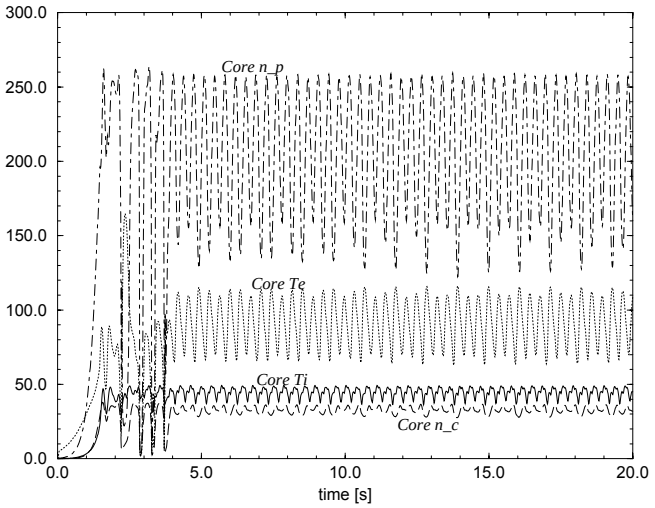

Figure 28: Run 9 - T_i, T_e, n_c and n_p at $(r/R) = 0$ versus time.

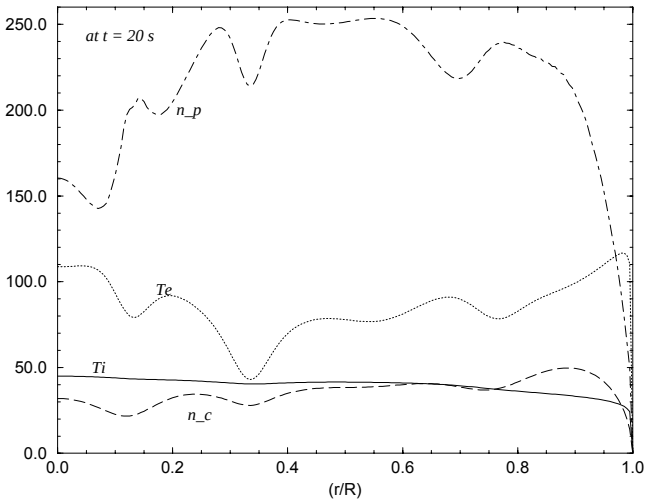

Figure 29: Run 9 - T_i, T_e, n_c and n_p versus (r/R) at $t = 20$ s.

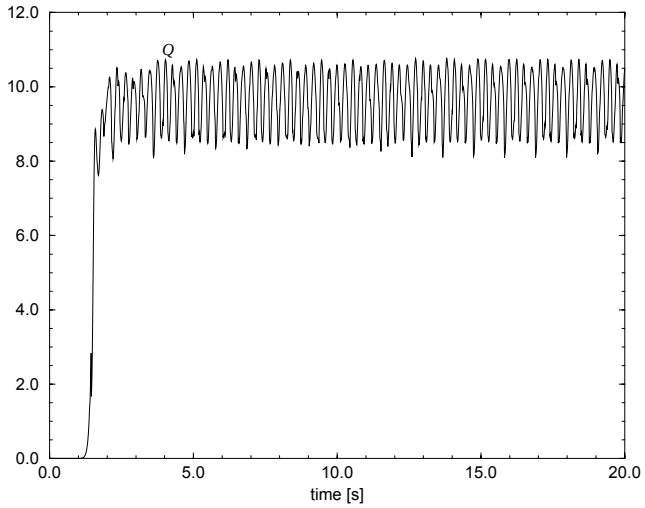

Figure 30: Run 9 - Q versus time.

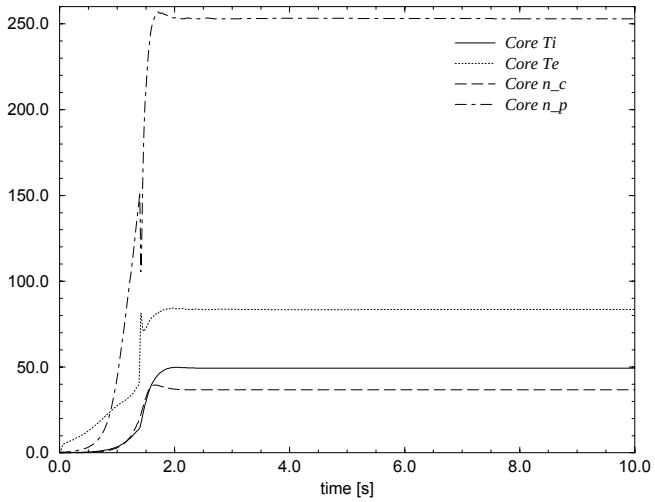

Figure 31: Run 10 - T_i, T_e, n_c and n_p at $(r/R) = 0$ versus time. Run 9 shares the same parameters with Run 10 except for (i) ETG diffusion activated for T_e (with numerical coefficient 0.1), and (ii) $P_{ECH} = 5.0 \times 10^4$ and turned on for core $T_e \geq 40$.

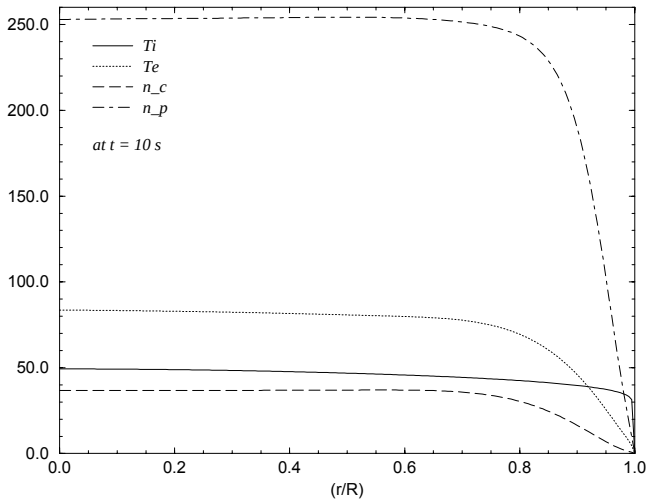

Figure 32: Run 10 - T_i, T_e, n_c and n_p versus (r/R) at $t = 10$ s. Run 9 shares the same parameters with Run 10 except for (i) ETG diffusion activated for T_e (with numerical coefficient 0.1), and (ii) $P_{ECH} = 5.0 \times 10^4$ and turned on for core $T_e \geq 40$.

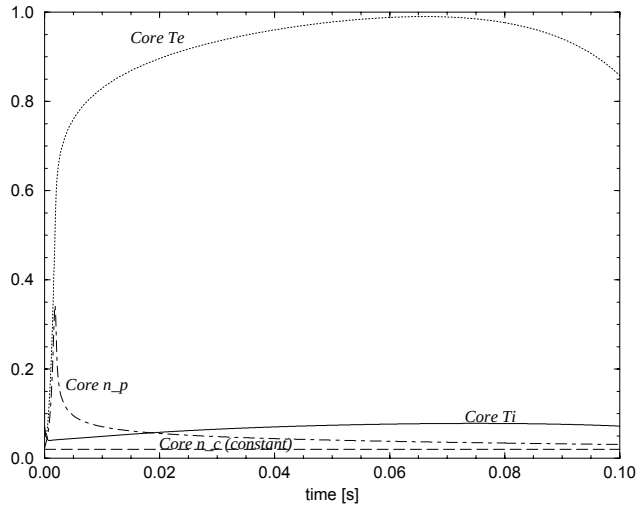

Figure 33: Pulse experiment - T_i, T_e, n_c and n_p at $(r/R) = 0$ versus time.

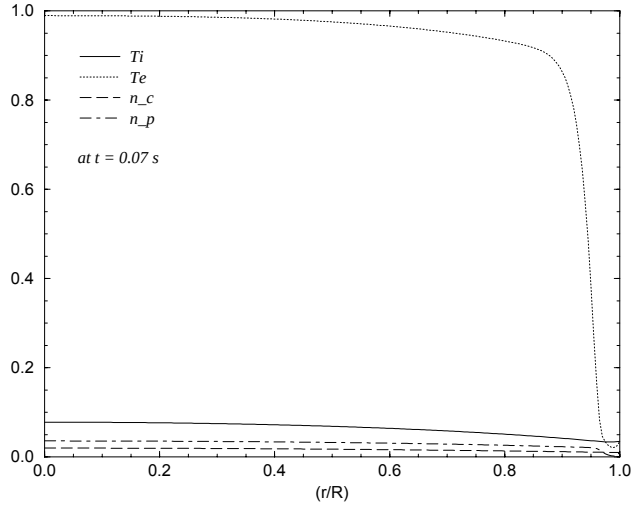

Figure 34: Pulse Experiment - T_i, T_e, n_c and n_p versus (r/R) at $t = 0.07$ s.

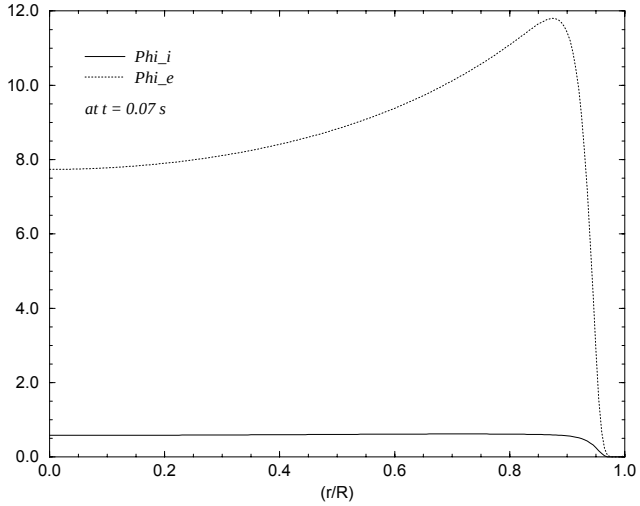

Figure 35: Pulse Experiment - ϕ_i and ϕ_e versus (r/R) at $t = 0.07$ s.

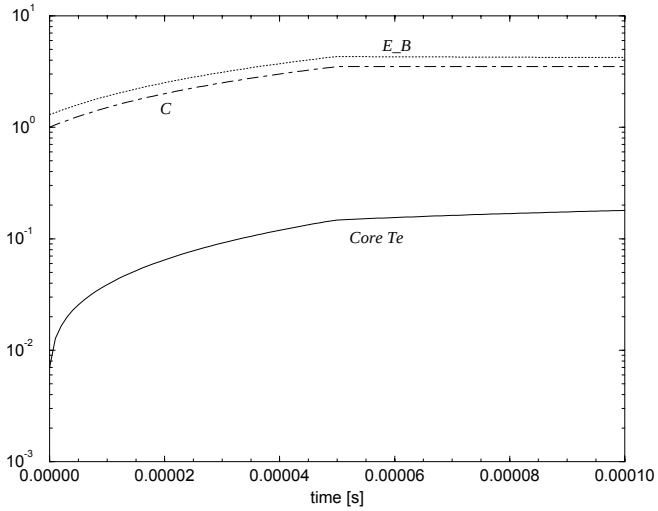

Figure 36: Simulation of single mirror compression experiment: $P_{ECH} = 0$, $C_{max} = 3.5$, $\tau_{comp} = 50\mu s$, $R_{M0} = 1$, $n_{p0} = 0.06$, $E_{B0} = 1.3$ and $T_{e0} = 0.007$.

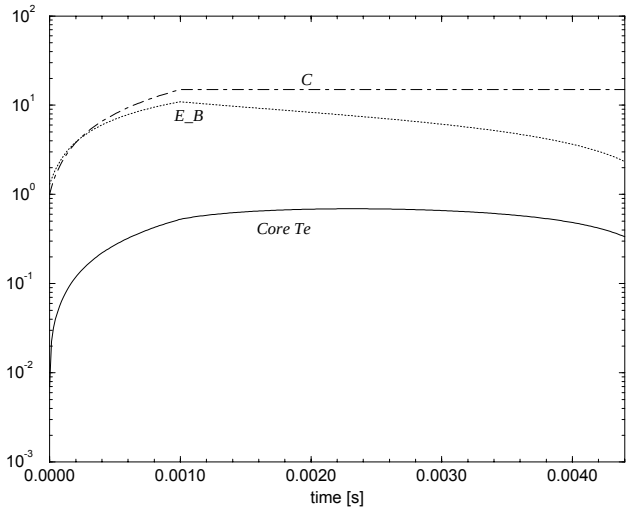

Figure 37: Simulation of tandem mirror compression experiment: $P_{ECH} = 0$, $C_{max} = 15$, $\tau_{comp} = 0.001\ s$, $R_{M0} = 1$, $n_{p0} = 0.06$, $n_{c0} = 0.025$, $E_{B0} = 1.3$ and $T_{e0} = 0.007$.

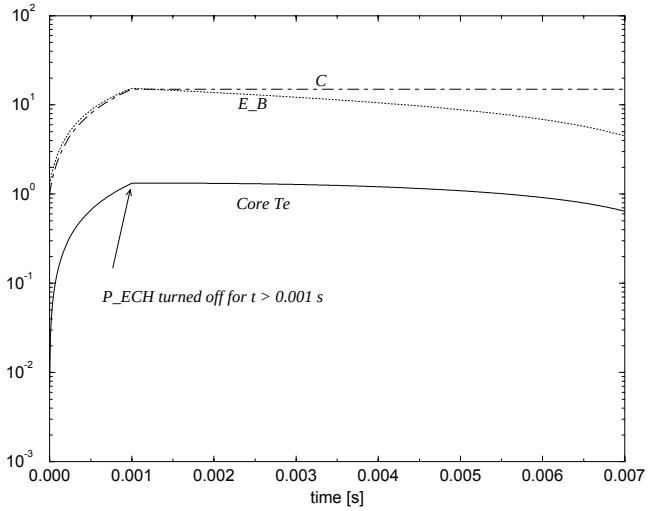

Figure 38: Simulation of tandem mirror compression experiment: $P_{ECH} = 50$, $C_{max} = 15$, $\tau_{comp} = 0.001\ s$, $R_{M0} = 1$, $n_{p0} = 0.06$, $n_{c0} = 0.025$, $E_{B0} = 1.3$ and $T_{e0} = 0.007$.