Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Physics Communications 161 (2004) 36–64
Computer Physics Communications
www.elsevier.com/locate/cpc

GTNEUT: A code for the calculation of neutral particle transport in plasmas based on the Transmission and Escape Probability method ☆

John Mandrekas *

Fusion Research Center, Georgia Institute of Technology, Atlanta, GA 30332-0425, USA

Received 23 October 2003; accepted 11 April 2004

Available online 19 June 2004

Abstract

GTNEUT is a two-dimensional code for the calculation of the transport of neutral particles in fusion plasmas. It is based on the Transmission and Escape Probabilities (TEP) method and can be considered a computationally efficient alternative to traditional Monte Carlo methods. The code has been benchmarked extensively against Monte Carlo and has been used to model the distribution of neutrals in fusion experiments.

Program summary

Title of program: GTNEUT
Catalogue identifier: ADTX
Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/ADTX
Computer for which the program is designed and others on which it has been tested: The program was developed on a SUN Ultra 10 workstation and has been tested on other Unix workstations and PCs.
Operating systems or monitors under which the program has been tested: Solaris 8, 9, HP-UX 11i, Linux Red Hat v8.0, Windows NT/2000/XP.
Programming language used: Fortran 77
Memory required to execute with typical data: 6 219 388 bytes
No. of bits in a word: 32
No. of processors used: 1
Has the code been vectorized or parallelized?: No
No. of bytes in distributed program, including test data, etc.: 300 709
No. of lines in distributed program, including test data, etc.: 17 365
Distribution format: compressed tar gzip file

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on SienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Tel.: +1-404-894-7730; Fax: +1-404-894-3733.
E-mail address: john.mandrekas@me.gatech.edu (J. Mandrekas).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2004.04.009

Keywords: Neutral transport in plasmas, Escape probability methods
Nature of physical problem: This code calculates the transport of neutral particles in thermonuclear plasmas in two-dimensional geometric configurations.
Method of solution: The code is based on the Transmission and Escape Probability (TEP) methodology [1], which is part of the family of integral transport methods for neutral particles and neutrons. The resulting linear system of equations is solved by standard direct linear system solvers (sparse and non-sparse versions are included).
Restrictions on the complexity of the problem: The current version of the code can handle only one species of atomic neutrals.
Typical running time: It depends on the size of the problem and the computing platform. For example, it takes 15.6 seconds of user time to run the second test problem on a SUN Ultra 10 workstation, using the sparse linear matrix solver.
Unusual features of the program: The program requires linking with the publicly available LAPACK linear algebra library which is usually included with the Fortran compilers of many UNIX vendors or can be obtained from NETLIB (www.netlib.org). To use the optional sparse matrix solver, the UMFPACK library is required which can be obtained from http://www.cise.ufl.edu/research/sparse/umfpack.
References: W.M. Stacey, J. Mandrekas, Nucl. Fusion 35 (1994) 1385.

PACS: 52.25.Ya; 52.65.-y; 52.55.Fa

Keywords: Neutral transport; *Neutrals in plasmas*; *Integral transport*; *Escape probability*; *Interface current method*

1. Introduction

Neutral particles (atoms and molecules) are always present in thermonuclear laboratory plasmas such as those encountered in magnetic fusion experiments, especially in regions near material surfaces. The majority of these neutrals are the direct result of particle recycling, i.e., plasma ions striking material walls and reflected back as neutrals, but they can also arise from external injection of neutral atoms into the system for fueling or heating purposes or be created by recombining plasma ions.

These neutrals can have an important effect on the local plasma particle and power balance. They affect the energy and particle fluxes to the first wall and divertor plates (therefore, playing a major role in processes such as wall erosion and intrinsic impurity production), as well as on the performance of important reactor engineering components such as the fueling and pumping systems. In addition, experimental observations and theoretical predictions suggest that neutrals play an important role in the overall performance of the core plasma, since they can affect the attainment of various improved confinement regimes, induce density limiting thermal instabilities in the plasma edge, etc. Therefore, the modeling of transport of neutral particles at the edge of magnetically confined plasmas is very important for the interpretation of present day fusion experiments and for the design of next generation fusion reactors, and computer codes that perform such simulations are indispensable tools for plasma modelers.

The modeling of neutral transport in fusion plasmas is challenging, since the highest neutral concentrations occur in regions characterized by considerable geometrical complexity (divertors, baffles, pumps, plenums, etc.), widely varying neutral mean-free-paths and background plasmas with densities and temperatures characterized by strong gradients.

Most state-of-the art codes for neutral particle transport are based on the Monte Carlo method [1–3], although methods based on alternative concepts (diffusion [4,5], discrete ordinates, various forms of integral transport [6]) have also been considered. Monte Carlo methods are capable of representing the complex geometries encountered at the plasma edge exactly, can treat the complex atomic and molecular physical processes characterizing the plasma edge region efficiently and can achieve very good accuracy if a sufficient number of particle histories are run. The most serious disadvantage of Monte Carlo-based neutral transport codes is their computational speed. They are computationally expensive, requiring a large number of particle histories in order to yield acceptable statistics. While this may not be a serious problem for stand-alone simulations with fixed background plasma, it becomes much more limiting in coupled plasma-neutrals simulations where a large number of iterations may be

required until the two-dimensional (2-D) plasma fluid calculation (which is usually computationally demanding itself) and the neutrals calculation converge. In addition, the numerical noise inherently present in Monte Carlo simulations makes convergence even more difficult. Such coupled edge plasma-neutrals simulations are becoming increasingly more common and the need for a faster alternative to traditional Monte Carlo codes has been recognized by the international fusion community.

The Georgia Tech Neutral Transport code GTNEUT described in this paper is such an alternative. GTNEUT is a computationally efficient and accurate tool for the calculation of neutral transport at the edge of thermonuclear plasmas based on the transmission and escape probability method [7]. The code has been benchmarked extensively against Monte Carlo and experiment [8–10].

The present version of the code has reached a level of maturity and stability and several research groups have expressed an interest in using it for their neutral simulation needs. We therefore feel that an extensive description of the code and the methodology is warranted in order to facilitate its use by the wider fusion community.

This paper is organized as follows: The basic assumptions and equations of the code are summarized in Section 2; the details of the code implementation, including a description of the input preparation, a discussion of the solution methodology and the overall structure of the code is presented in Section 3; two test problems, included in the code distribution, are presented and discussed in Section 4; conclusions and a brief discussion, including plans for future development, follow in Section 5; and, finally, a complete list of the input variables is included in Appendix A.

2. Equations and methods

GTNEUT is based on the Transmission and Escape Probabilities (TEP) method. While the details of the TEP method and its application to specific problems have been published elsewhere [7–10], the basic methodology and governing equations are described in this section for completeness.

In the TEP method [7], the region of interest is subdivided into an arbitrary number of straight-sided convex cells.[1] The shape of each cell can be as complicated as required in order to match the local geometry, and can have an arbitrary number of sides. The computational domain of interest is bounded by the first wall and divertor plates (including any other relevant first wall structures, such as pumps, baffles, plenums, etc.), and, optionally, by the *core plasma*. Core plasma, in this context, is the part of the plasma where knowledge of the exact neutral density population is not necessary (usually because it is anticipated to be very small) and it is treated with an albedo boundary condition. It does not necessarily correspond to the entire plasma inside the separatrix in diverted tokamaks, since the boundary can be arbitrarily selected to include as much of the core plasma in the neutral computational domain as it is desired.

The present version of the GTNEUT code is 2-dimensional (2-D), i.e., it is assumed that the plasma and neutral distributions, as well as any surface or volume neutral sources, are unchanged along the third (ignorable) coordinate. This is a choice that we made for simplicity and computational economy and is not an inherent limitation of the TEP method, which can be easily extended to three dimensions.

It is important to emphasize that, while references to diverted tokamak geometry will be frequently encountered in this paper, the methodology and the code are not restricted to this specific topology. GTNEUT can calculate the neutral density and related quantities of any 2-D configuration that can be subdivided into straight convex cells (e.g., linear plasma devices, selected regions near plasma facing components, etc.).

[1] Earlier versions of the code supported circular segments [7], but this option was rather restrictive and has not been carried over to later versions of the code.

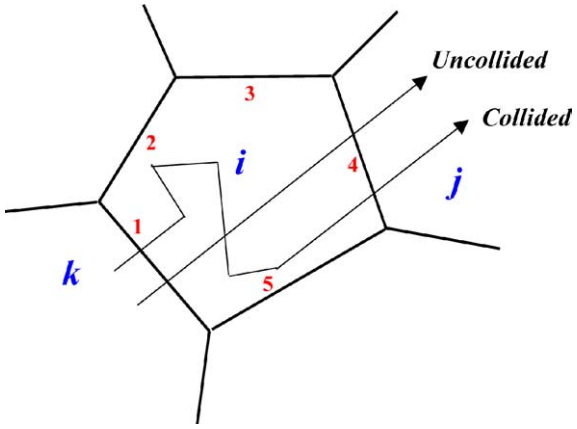

Fig. 1. Schematic diagram showing region i and its adjacent regions and the partial currents at the interfaces.

2.1. Basic TEP equations

The equations are formulated in terms of the interface partial currents or "fluxes"[2] through the sides of each cell, $\Gamma_{i,j}$. The flux $\Gamma_{i,j}$ represents the total partial current, in units of *# of particles/s*, from cell "i" into adjacent cell "j" (Fig. 1). In terms of the neutral angular flux $\psi(\mathbf{r},\boldsymbol{\Omega})$, $\Gamma_{i,j}$ is defined as:

$$\Gamma_{i,j} \equiv \int_{S_{ij}} \mathrm{d}S \int_{\boldsymbol{\Omega}\cdot\hat{\mathbf{n}}>0} \mathrm{d}\boldsymbol{\Omega}(\boldsymbol{\Omega}\cdot\hat{\mathbf{n}})\psi(\mathbf{r}_{ij},\boldsymbol{\Omega}) \tag{1}$$

where S_{ij} is the interface between cells i and j, $\hat{\mathbf{n}}$ is the normal unit vector at the S_{ij} interface and $\psi(\mathbf{r}_{ij},\boldsymbol{\Omega})$ is the angular flux of the neutrals at $\mathbf{r}_{ij}$. The integration over all solid angles satisfying $\boldsymbol{\Omega}\cdot\hat{\mathbf{n}} \geqslant 0$ ensures that we count the particles entering cell j from cell i through the interface S_{ij}.

For each interface between two cells we can write particle balance equations that relate the various fluxes incident on the sides of the cell. To do this, we note that the total particle flux $\Gamma_{i,j}$ from cell i into cell j consists of three distinct contributions:

- *Uncollided neutrals*: These are neutrals that entered cell i from one of its adjacent cells and were transmitted across cell i into the adjacent cell j without undergoing any collisions with background plasma ions and electrons. The uncollided contribution can be expressed as:

$$\Gamma^u_{i,j} = \sum_k \Gamma_{k,i} T^i_{k,j} \tag{2}$$

  where the summation extends over all the cells k contiguous to cell i, and $T^i_{k,j}$ is the first-flight transmission coefficient. If we assume that the neutral angular flux is isotropic on both sides of the interfaces between the cells (also known as the *double* P_0 or DP_0 approximation in transport theory) and that is also uniform at the interface, the first-flight transmission coefficient $T^i_{k,j}$ is equal to [11]:

$$T^i_{k,j} = \frac{2}{\pi L_{ki}} \int_0^{L_{ki}} \mathrm{d}\xi_{ki} \int_{\phi_{\min}(\xi_{ki})}^{\phi_{\max}(\xi_{ki})} \mathrm{d}\phi \, \sin\phi \, K_{i3}\left(\frac{l_i(\phi(\xi_{ki}))}{\lambda_i}\right) \tag{3}$$

[2] Notice that we refer to the quantity Γ as "*flux*" or "*current*" although its units are #/s rather than #/m^2-s, i.e., it represents the *total* number of particles crossing the surface under consideration.

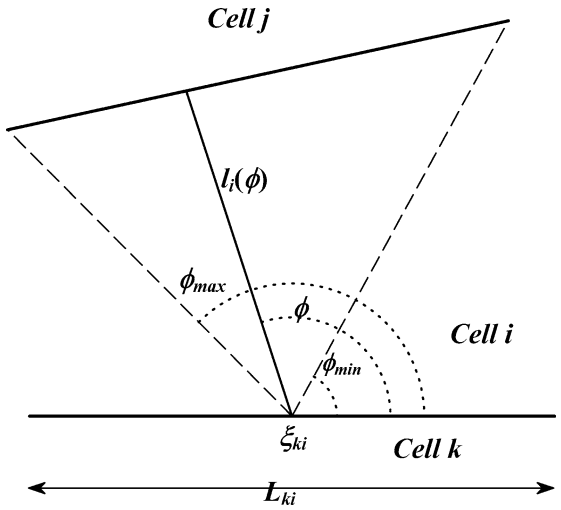

Fig. 2. Geometry for the calculation of the first-flight transmission coefficients, Eq. (3).

where, as shown in Fig. 2, L_{ki} is the length of the interface between regions k and i, ξ_{ki} is the coordinate along the L_{ki} interface, l_i is the length of the distance traveled by the neutral in the 2-D plane from a point ξ_{ki} on the entering surface to a point in the exiting surface, ϕ is the angle between the entering surface and the chord l_i, λ_i is the total neutral mean free path in cell i and Ki_3 is the third-order Bickley–Naylor function [11]:

$$Ki_n(\tau) = \int_0^{\pi/2} \mathrm{d}\theta \, \sin^{n-1}(\theta) \exp\left(-\frac{\tau}{\sin\theta}\right). \tag{4}$$

- *Collided neutrals*: These are the neutrals that entered cell i from all contiguous cells, had one or more charge exchange or elastic scattering collision in cell i with background plasma ions (*neutral–neutral scattering collisions are not considered in the present version of* GTNEUT) and eventually escaped into cell j through the common interface between cells i and j. The flux contribution of these *collided* neutrals can be written in the following form:

$$\Gamma^c_{i,j} = \sum_k \Gamma_{k,i} \left(1 - \sum_l T^i_{k,l}\right) c_i P_i \Lambda_{ij} \tag{5}$$

where c_i is the probability that a collision in cell i is a scattering or charge exchange event, P_i is the total escape probability that the particle or its progeny escapes region i after one or more charge exchange or elastic scattering collisions, and Λ_{ij} is the probability that particles escaping cell i exit through the side of i that is adjacent to cell j. The summations extend over all cells contiguous to cell i.

- *Source neutrals*: These can arise from either external or internal sources (e.g., volume recombination). If the volumetric neutral source in cell i is S^i_{ext}, then its contribution to the total particle flux can be written as:

$$\Gamma^s_{i,j} = S^i_{\mathrm{ext}} P_i \Lambda_{ij} \tag{6}$$

where the total escape probability P_i and geometric escape factor Λ_{ij} are similar to those appearing in Eq. (5).

Combining the three contributions to the total particle flux Eqs. (2), (5) and (6) the partial current balance equation for the interface between cells i and j can be written in the following form:

$$\Gamma_{i,j} = \sum_k \Gamma_{k,i} T^i_{k,j} + \sum_k \Gamma_{k,i} \left(1 - \sum_l T^i_{k,l}\right) c_i P_i \Lambda_{ij} + S^i_{\text{ext}} P_i \Lambda_{ij}. \tag{7}$$

We now discuss some of the coefficients and terms appearing in the equations above in more detail:

The charge exchange and elastic scattering fraction c_i, also known as the number of secondary neutrals per collision, is defined as follows:

$$c_i = \frac{\langle \sigma \upsilon \rangle_{\text{cx}} + \langle \sigma \upsilon \rangle_{\text{el}}}{\frac{n_e}{n_i} \langle \sigma \upsilon \rangle_e + \langle \sigma \upsilon \rangle_i + \langle \sigma \upsilon \rangle_{\text{cx}} + \langle \sigma \upsilon \rangle_{\text{el}}} \tag{8}$$

where $\langle \sigma \upsilon \rangle$stands for the Maxwellian averaged reaction rates, the subscripts cx, el, e, and i denote charge-exchange, elastic scattering (included here for those atomic databases that support it), electron impact ionization and ion impact ionization respectively, and n_e and n_i are the electron and ion densities of the background plasma. Although the present version of GTNEUT assumes a single-species hydrogenic plasma background, Eq. (8) can be generalized to handle an arbitrary number of background ion species, including charge states of impurities [7].

The various reaction rates are computed using the polynomial fits by Janev et al. [12], and depend on the neutral particle species and energy as well as on the background plasma electron or ion temperatures. In addition to the Janev database, two other options are available in GTNEUT for the calculation of the various atomic reaction rates: The first includes hydrogenic ionization and charge exchange rates from the DEGAS code [1], and the second is based on a recent compilation of data by E.W. Thomas [13]. It should be noted that these two alternative options include multi-step ionization effects, having been derived using a collisional radiative model (the electron impact ionization rates in GTNEUT from the Janev database correspond to the ground state). In addition, the Thomas data include ion-neutral elastic scattering rates (the term $\langle \sigma \upsilon \rangle_{\text{el}}$ in Eq. (8)) based on the work by D.R. Schultz et al. [14], as well as Lyman α suppression effects which can enhance ionization at high densities ($n > 10^{19}/\text{m}^3$) [13]. Appendix A explains how these options can be selected by the user. Potential users can easily implement their own databases, by modifying the routine *calcmfp* appropriately.

The *total* escape probability P_i for a neutral particle or its neutral progeny can be calculated by observing that of the c_i secondary neutrals produced per scattering or charge exchange event in cell i, a fraction P_{0i} escapes cell i without undergoing further collisions while the rest, $1 - P_{0i}$, remain in cell i to interact with the background plasma again. During this new generation of collisions, a fraction $c_i(1 - P_{0i})P_{0i}$ will escape cell i without suffering additional collisions, while the rest will constitute the next generation collision source which, in turn, will contribute $[c_i(1 - P_{0i})]^2 P_{0i}$ escaping neutrals, and so on. A schematic representation of this sequence of collision events for the first three generations is shown in Fig. 3. Summing over all generations, the total escape probability P_i can be written as follows:

$$P_i = P_{0i} \sum_{n=0}^{\infty} \left[c_i(1 - P_{0i})\right]^n = \frac{P_{0i}}{1 - c_i(1 - P_{0i})} \tag{9}$$

where P_{0i} is the first-flight escape probability. The expression for the *collided* flux given by Eq. (5) follows by summing all elements of the last column in Fig. 3, using Eq. (9) for the total escape probability, multiplying by the directional probability factor Λ_{ij} and noting that the first collision source $S^{(1)}_i$ is equal to $\sum_k \Gamma_{k,i}(1 - \sum_l T^i_{k,l})$.

Just like the first-flight transmission factor (Eq. (3)), the first-flight escape probability P_{0i} can be calculated exactly under the assumption of an isotropic collision rate distribution [8,11]. However, in the interest of computational efficiency, a rational approximation for the calculation of P_{0i} is used in GTNEUT:

$$P_{0i} = \frac{1}{X_i}\left[1 - \left(1 + \frac{X_i}{n}\right)^{-n}\right]. \tag{10}$$

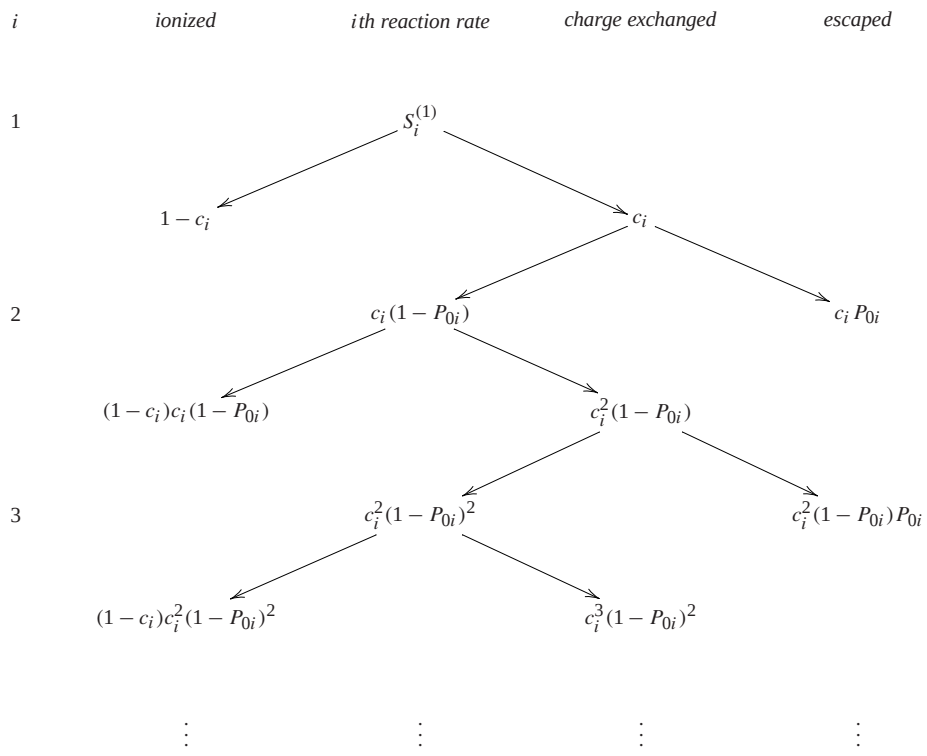

Fig. 3. Schematic representation of collision events in cell i for the first three generations, starting with the first collision source.

In Eq. (10), $X_i = 4V_i/\lambda_i S_i$, where V_i is the volume (surface in 2-D) of cell i, λ_i is the neutral mean free path in cell i, S_i is the surface area (perimeter in 2-D) of cell i, and n is an arbitrary exponent. The approximate form of Eq. (10) derives from similar rational approximations by Wigner [15] and Sauer [16]. Detailed calculations and comparisons with Monte Carlo have resulted in a value of $n = 2.09$ for our simulations [8,17].

Finally, the factor Λ_{ij} appearing in Eqs. (5)–(7) represents the probability that a neutral escaping cell i, will escape through the side if i that faces the adjacent cell j. While the presence of ion flows and magnetic fields can influence the value of Λ_{ij} [7], in the present version of the code Λ_{ij} is treated merely as a geometric directionality factor and is equal to the fraction of the boundary surface (or perimeter under our 2-D assumption) of cell i that is in contact with the adjacent cell j.

2.2. Neutral energy considerations

The current implementation of the TEP methodology in GTNEUT, assumes that the neutral energy in each cell is equal to the local ion temperature. An exception is made for wall originated (reflected) or source neutrals, which maintain their original energies (input-specified or determined by the wall reflection model) until they interact with the background plasma ions, in which case they, too, assume the local ion temperature. Extensive tests against Monte Carlo [8,17] have shown that this "local ion temperature" assumption is a good approximation when the neutral mean-free-path λ is comparable to or less than the characteristic dimension Δ of the cell under consideration, and when the background plasma properties are not characterized by strong gradients. A two-energy group formulation has been recently implemented [10], which has improved the accuracy of the code even in cases where the mean free path and background plasma gradient restrictions are not met. The two-energy group formulation is discussed in Section 2.6 below.

2.3. First collision source effects

The first collision source consists of those neutrals that entered cell i from one of its adjacent cells and had their first charge exchange or elastic scattering collision in cell i. In deriving Eq. (7) above, the first collision source was treated as the precursor of an infinite number of generations of collision sources, each described by the same secondary neutrals fraction c_i, escape probability P_{0i} and geometric factor Λ_{ij} resulting in Eq. (5) for the collided flux.

However, the first collision source is likely to be influenced by the originating cell of the neutrals that comprise it, in contrast to subsequent generations which are assumed to lose their ancestral "memory" after a few scattering collisions with the background ions of cell i. A neutral entering cell i from one of its contiguous cells still carries an energy consistent with the properties of that cell. This should affect the calculation of the neutral fraction c_i, which depends on the neutral energy through its dependence on the various neutral-ion reaction rates. In addition, if the mean free path λ_i of this neutral in cell i is much smaller than the characteristic dimension Δ_i of cell i, i.e., $\lambda_i/\Delta_i \ll 1$, then the first collision source will be highly non-uniform since most reactions with background plasma ions will occur near the interface of these two cells. As a result, the probability of escaping back across that incident surface would be greater than the probability of escaping across another surface. Since the calculation of the escape probability P_{0i} and the geometric transmission factor Λ_{ij} described in the previous section assume a uniform collision source, it is expected that these coefficients or their product $P_{0i}\Lambda_{ij}$ may be different for the first collision source. It is therefore desirable to rewrite the collided flux contribution, Eq. (5), in a slightly different form in which the first collision source is separated out:

$$\Gamma^c_{i,j} = \sum_k \Gamma_{k,i}\left(1 - \sum_l T^i_{k,l}\right) c_{i,k}\left[P_{0i,k}\Lambda^{(k)}_{ij} + (1 - P_{0i,k})c_i P_i \Lambda_{ij}\right]. \tag{11}$$

The first term in the brackets represents the first collision source and the second term the contribution from all the other generations. The "k" subscript or superscript appearing in $c_{i,k}$, $P_{0i,k}$ and $\Lambda^{(k)}_{ij}$ signifies that these coefficients have been calculated taking into account that they entered cell i from adjacent cell k. The rest of the coefficients in Eq. (11) are the same as before. It should be noted that Eq. (11) reduces to Eq. (5) when first collision effects are ignored, i.e., setting $c_{i,k} = c_i$, $\Lambda^{(k)}_{ij} = \Lambda_{ij}$, $P_{0i,k} = P_{0i}$ and using Eq. (9) for the total escape probability P_i.

In the present version of the GTNEUT code, $c_{i,k}$ is calculated taking into account the ancestry of the first collision neutrals. The rest of the first collision coefficients however, $P_{0i,k}$ and $\Lambda^{(k)}_{ij}$, are assumed to be equal to P_{0i} and Λ_{ij} respectively, i.e., they do not include a first-collision correction. We should note however that, in the code, the first collision terms appear explicitly, so adding first collision effects to these coefficients should be straightforward once the appropriate methodology is developed.

Similar arguments can be made for the neutrals due to internal or external sources. The volumetric source S^i_{ext} can be non-uniform and the emitted neutrals can have an anisotropic distribution and various initial energies. In this case, the contribution from the original source neutrals, before they suffer their first collision with the background plasma ions, can be separated out and Eq. (6) can be written as:

$$\Gamma^s_{i,j} = S^i_{\text{ext}}\left[P^0_{0i}\Lambda^{s,0}_{ij} + \left(1 - P^0_{0i}\right)c^0_i P_i \Lambda_{ij}\right] \tag{12}$$

where the superscripts "s" and "0" indicate that the relevant quantities are calculated taking into account the properties of the source.

2.4. Boundary conditions

As discussed in the beginning of Section 2, the computational domain of interest is bounded by material walls and, in some cases, by a plasma region (referred to as "*core plasma*" in this context) in which the explicit knowledge

of the neutral distribution is not necessary. In this section, we discuss the boundary conditions imposed by these two types of boundaries.

2.4.1. Reflection from a plasma region

If one or more sides of cell i interface with a core plasma region *kpl*, an albedo boundary condition is used to express the flux from the plasma region *kpl* into cell i:

$$\Gamma_{kpl,i} = \alpha_{kpl}\Gamma_{i,kpl}. \tag{13}$$

The albedo coefficient α_{kpl} is calculated from a numerical fit to detailed Monte Carlo albedo simulations for different values of the charge exchange fraction parameter c_{kpl}. The fit is valid for the entire region of interest ($0 \leqslant c_{kpl} \leqslant 1$) and is in excellent agreement with semi-analytic results from transport theory [18].

This fit replaces our earlier diffusion theory based approximation (Eq. (30) in Ref. [7]) which was inaccurate for smaller values of c_{kpl}, becoming negative for $c_{kpl} < 0.57$.

2.4.2. Reflection from material walls

Laboratory plasmas are usually surrounded by the material surfaces of the confining vessel. In a tokamak device, these surfaces would include the first wall, the divertor plates or limiters and any other structures in close proximity to the plasma.

When ions or neutrals interact with a material surface, one of the three following possible outcomes can occur [19]. They can (a) be directly back-scattered or reflected while retaining a significant fraction of their original (impact) energy, (b) reach thermal equilibrium with the lattice atoms and be re-emitted as molecules at low energies, of the order of the wall temperature, and (c) become permanently trapped inside the material wall. The thermal molecules of the second group, after dissociation, become *Franck–Condon* atoms with energies of a few electron volts [19]. The present version of the GTNEUT code does not treat molecules explicitly, assuming instead that they dissociate at the point at which they are introduced into the plasma, and subsequently treated as *Franck–Condon* atoms with an input-assigned energy.

Accordingly, the particle flux from the wall segment *kw* to an adjacent plasma cell i, $\Gamma_{kw,i}$, can be written in terms of the flux $\Gamma_{i,kw}$ from cell i onto the wall segment *kw* as follows:

$$\Gamma_{kw,i} = \Gamma_{\text{ext}}^{kw} + R_n^{kw}\Gamma_{i,kw} + \left(1 - R_n^{kw}\right)\left(1 - f_{\text{abs}}^{kw}\right)\Gamma_{i,kw}. \tag{14}$$

The first term in Eq. (14), Γ_{ext}^{kw}, represents an external flux contribution (e.g., a gas puffing fueling source), the second term represents the fraction of particles directly reflected or back-scattered and the last term represents the neutrals that are re-emitted as molecules from the material surface and treated here as Franck–Condon atoms. R_n^{kw} is the particle reflection coefficient, which depends on the impact energy of the ions and the material properties of the surface, and f_{abs}^{kw} is a wall absorption coefficient taking into account any particles that remain trapped in the wall.

Since most coefficients in Eqs. (7) and (11) depend on the neutral energy through their dependence on the neutral mean free path λ or on the various ion-neutral reaction rates, it is important that they are evaluated at the appropriate energy of each of these three groups of wall-originated neutrals. The externally launched particles are assumed to have energy E_0 which is treated as an input variable, the back-scattered "fast" particles have an energy equal to $T_i R_E^{kw}/R_n^{kw}$, where T_i is the ion temperature of the plasma region adjacent to the wall segment *kw* and R_E^{kw} is the energy reflection coefficient, and the "slow" neutrals are assumed to emerge at an input-specified low energy corresponding to the Franck–Condon energy for atoms. The particle and energy reflection coefficients R_n^{kw} and R_E^{kw} of each wall segment are calculated using standard fits that are valid for a wide range of wall materials, particle species and impact energies [20,21].

In addition to the realistic wall reflection model described above, a simpler reflection model is also available in which the reflection coefficient is an input variable and neutrals are returned to the plasma at their original energies.

In this case, Eq. (14) for the flux from the wall segment *kw* into cell i becomes:

$$\Gamma_{kw,i} = R^{kw}\Gamma_{i,kw} + \Gamma^{kw}_{\text{ext}} \tag{15}$$

where R^{kw} is the input-specified reflection coefficient. This simple model is useful for setting up vacuum interfaces ($R^{kw} = 0$, $\Gamma^{kw}_{\text{ext}} = 0$) or symmetric boundaries ($R^{kw} = 1$, $\Gamma^{kw}_{\text{ext}} = 0$) as well as for compatibility with the original TEP implementation, which did not include a realistic wall reflection model [7].

Finally, when GTNEUT is coupled to a plasma fluid code, the reflection of the plasma ions from the material walls should also be taken into account. Since there are no significant differences between the reflection of incident atoms or ions [19], the same approach followed for the reflection of neutral atoms is used to evaluate the neutral flux from wall segment *kw* into the adjacent plasma cell i due to an ion flux Γ^{kw}_{ion}. In this case, Eq. (15) for the neutral particle flux entering cell i from the wall segment *kw* becomes:

$$\Gamma_{kw,i} = \Gamma^{kw}_{\text{ext}} + R^{kw}_n\left(\Gamma_{i,kw} + \Gamma^{kw}_{\text{ion}}\right) + \left(1 - R^{kw}_n\right)\left(1 - f^{kw}_{\text{abs}}\right)\left(\Gamma_{i,kw} + \Gamma^{kw}_{\text{ion}}\right) \tag{16}$$

where the incident ion flux Γ^{kw}_{ion} should be provided by the plasma fluid code.

2.5. Final form of equations

Using Eqs. (13), (14) and (16) to eliminate the fluxes from plasma regions and wall segments into adjacent cells in favor of the fluxes from the cells into these regions, and separating out the first collision contributions as shown in Eqs. (11) and (12), the partial current balance equation for the interface between cells i and j (Eq. (7)), can be written in the following form:

$$\begin{aligned}
\Gamma_{i,j} = {} & \sum_{k}^{k\neq kw,kpl} \Gamma_{k,i}T^{i}_{k,j} + \sum_{k}^{k\neq kw,kpl} \Gamma_{k,i}\left(1 - \sum_{l} T^{i}_{k,l}\right) c_{i,k}\left[P_{0i,k}\Lambda^{(k)}_{ij} + (1 - P_{0i,k})c_i P_i \Lambda_{ij}\right] \\
& + \sum_{kpl} \alpha_{kpl}\Gamma_{i,kpl}T^{i}_{kpl,j} + \sum_{kpl} \alpha_{kpl}\Gamma_{i,kpl}\left(1 - \sum_{l} T^{i}_{kpl,l}\right) c_{i,kpl}\left[P_{0i,kpl}\Lambda^{(kpl)}_{ij} + (1 - P_{0i,kpl})c_i P_i \Lambda_{ij}\right] \\
& + \sum_{kw} \Gamma^{kw}_{\text{ext}}T^{i,0}_{kw,j} + \sum_{kw} R^{kw}_n\Gamma_{i,kw}T^{i,f}_{kw,j} + \sum_{kw}\left(1 - R^{kw}_n\right)\left(1 - f^{kw}_{\text{abs}}\right)\Gamma_{i,kw}T^{i,s}_{kw,j} \\
& + \sum_{kw} \Gamma^{kw}_{\text{ext}}\left(1 - \sum_{l} T^{i,0}_{kw,l}\right) c^{0}_{i,kw}\left[P^{0}_{0i,kw}\Lambda^{kw,0}_{ij} + \left(1 - P^{0}_{0i,kw}\right)c_i P_i \Lambda_{ij}\right] \\
& + \sum_{kw} R^{kw}_n\Gamma_{i,kw}\left(1 - \sum_{l} T^{i,f}_{kw,l}\right) c^{f}_{i,kw}\left[P^{f}_{0i,kw}\Lambda^{kw,f}_{ij} + \left(1 - P^{f}_{0i,kw}\right)c_i P_i \Lambda_{ij}\right] \\
& + \sum_{kw}\left(1 - R^{kw}_n\right)\left(1 - f^{kw}_{\text{abs}}\right)\Gamma_{i,kw}\left(1 - \sum_{l} T^{i,s}_{kw,l}\right) c^{s}_{i,kw}\left[P^{s}_{0i,kw}\Lambda^{kw,s}_{ij} + \left(1 - P^{s}_{0i,kw}\right)c_i P_i \Lambda_{ij}\right] \\
& + \sum_{kw} R^{kw}_n\Gamma^{kw}_{\text{ion}}T^{i,f}_{kw,j} + \sum_{kw}\left(1 - R^{kw}_n\right)\left(1 - f^{kw}_{\text{abs}}\right)\Gamma^{kw}_{\text{ion}}T^{i,s}_{kw,j} \\
& + \sum_{kw} R^{kw}_n\Gamma^{kw}_{\text{ion}}\left(1 - \sum_{l} T^{i,f}_{kw,l}\right) c^{f}_{i,kw}\left[P^{f}_{0i,kw}\Lambda^{kw,f}_{ij} + \left(1 - P^{f}_{0i,kw}\right)c_i P_i \Lambda_{ij}\right] \\
& + \sum_{kw}\left(1 - R^{kw}_n\right)\left(1 - f^{kw}_{\text{abs}}\right)\Gamma^{kw}_{\text{ion}}\left(1 - \sum_{l} T^{i,s}_{kw,l}\right) c^{s}_{i,kw}\left[P^{s}_{0i,kw}\Lambda^{kw,s}_{ij} + \left(1 - P^{s}_{0i,kw}\right)c_i P_i \Lambda_{ij}\right] \\
& + S^{i}_{\text{ext}}P^{0}_{0i}\Lambda^{s,0}_{ij} + S^{i}_{\text{ext}}\left(1 - P^{0}_{0i}\right)c^{0}_{i}P_i\Lambda_{ij}.
\end{aligned} \tag{17}$$

In Eq. (17), cell i is assumed to be bounded by an arbitrary number of wall segments *kw*, plasma regions *kpl* and other cells k. The various superscripts appearing in the wall-originated components ("0" for external fluxes, "ion" for background plasma ion fluxes, "f" for the fast, back-scattered component and "s" for the slow, low energy Franck–Condon neutrals) indicate that the various coefficients have been calculated at the neutral energies characteristic of each group.

The total ionization rate I_i in cell i can be obtained by summing the ionization contributions from each generation of reactions, as shown in Fig. 3. Evaluating the infinite sum of the elements in the leftmost column (labeled *ionized*) of Fig. 3, we can show that

$$I_i = \sum_k \Gamma_{k,i}\left(1 - \sum_l T^i_{k,l}\right)(1-c_i)/\left[1 - c_i(1-P_{0i})\right].$$

This result can be generalized by considering contributions from wall segments, core plasma and external sources and taking explicitly into account first collision effects by assigning region-dependent secondary neutral fractions $c_{i,k}$ and escape probabilities $P_{0i,k}$ as discussed in Section 2.3. In this case, the total ionization rate is equal to:

$$\begin{aligned}
I_i = {} & \sum_k^{k\neq kw} \Gamma_{k,i}\left(1 - \sum_l T^i_{k,l}\right)\left[(1-c_{i,k}) + \frac{c_{i,k}(1-c_i)(1-P_{0i,k})}{1-c_i(1-P_{0i})}\right] \\
& + \sum_{kw} \Gamma^{kw}_{\text{ext}}\left(1 - \sum_l T^{i,0}_{kw,l}\right)\left[\left(1-c^0_{i,kw}\right) + \frac{c^0_{i,kw}(1-c_i)(1-P^0_{0i,kw})}{1-c_i(1-P_{0i})}\right] \\
& + \sum_{kw} R^{kw}_n\left(\Gamma_{i,kw} + \Gamma^{kw}_{\text{ion}}\right)\left(1 - \sum_l T^{i,f}_{kw,l}\right)\left[\left(1-c^f_{i,kw}\right) + \frac{c^f_{i,kw}(1-c_i)(1-P^f_{0i,kw})}{1-c_i(1-P_{0i})}\right] \\
& + \sum_{kw}\left(1-R^{kw}_n\right)\left(1-f^{kw}_{\text{abs}}\right)\left(\Gamma_{i,kw} + \Gamma^{kw}_{\text{ion}}\right)\left(1 - \sum_l T^{i,s}_{kw,l}\right)\left[\left(1-c^s_{i,kw}\right) + \frac{c^s_{i,kw}(1-c_i)(1-P^s_{0i,kw})}{1-c_i(1-P_{0i})}\right] \\
& + \left(1-P^0_{0i}\right)S^i_{\text{ext}}\left[1 - c^0_i + \frac{c^0_i(1-c_i)(1-P_{0i})}{1-c_i(1-P_{0i})}\right]
\end{aligned} \tag{18}$$

where, as in Eq. (17), the first summation is over all non-wall adjacent cells, the next three summations represent walls with external fluxes, fast and slow reflected neutrals, and the last term is from any volumetric sources.

Once the ionization rate is known, the neutral density n_{0i} in each cell i can be calculated from:

$$n_{0i} = \frac{I_i}{n_i \langle \sigma v\rangle^{\text{tot}}_{\text{ion}} V_i} \tag{19}$$

where n_i is the background ion density in cell i, V_i is the volume of cell i and $\langle \sigma v\rangle^{\text{tot}}_{\text{ion}}$ is the total ionization rate (electron impact and ion impact ionization). Eq. (19) is actually an approximation, since the different ionization terms in Eq. (18) may correspond to different neutral energies. However, since the dominant ionization rate is, by far, the electron impact ionization rate which is independent of the neutral energy, and the ion impact ionization rate depends only weakly on the neutral energy, Eq. (19) is a very good approximation.

2.6. Extension to two energy groups

As discussed in Section 2.2, the "local ion temperature" approximation is a reasonable assumption when the neutral mean-free-path λ is comparable to or less than the characteristic dimension Δ of the region under consideration, and when there are no strong gradients in the background plasma properties. When either of these conditions is not met, however, the possibility of introducing errors in the calculation due to assigning the wrong energy to parts of the neutral population must be considered.

To remedy this situation, the TEP methodology was modified by introducing two distinct energy groups: a "*slow*" energy group, consisting of the neutral atoms at the Franck–Condon energy formed by the dissociation of molecules re-emitted from the wall or injected as a gas fueling source, and a "*fast*" energy group including the neutrals that are in thermal equilibrium with the background ion population. Directly reflected neutrals are assumed to be part of the fast group, since they retain a significant fraction of their original energy. Making the plausible simplifying assumption that every charge exchange or elastic scattering reaction moves slow neutrals to the fast neutrals group and that no fast neutrals are "scattered" from the fast to the slow group, the particle balance equations for the two groups and for internal regions (i.e., regions not bounded by material surfaces) are:

$$\Gamma^f_{i,j} = \sum_k \Gamma^f_{k,i} T^{i,f}_{k,j} + \sum_k \Gamma^f_{k,i} \left(1 - \sum_l T^{i,f}_{k,l}\right) c_{i,k} P_i \Lambda_{ij} + \sum_k \Gamma^s_{k,i} \left(1 - \sum_l T^{i,s}_{k,l}\right) c^s_i P_i \Lambda_{ij}, \tag{20}$$

$$\Gamma^s_{i,j} = \sum_k \Gamma^s_{k,i} T^{i,s}_{k,j} \tag{21}$$

where, as before, the superscripts "*s*" and "*f*" correspond to the slow and fast energy groups. The third term in Eq. (20) represents the slow neutrals that entered the fast group, after charge-exchanging with the background plasma ions. From Eq. (21), we can see that the slow neutrals group propagates only through uncollided fluxes, and does not have any contribution from charge exchange reactions.

For neutrals originating from wall segments due to reflection, re-emission or external sources, the fast and slow fluxes into the adjacent plasma region are:

$$\begin{aligned} \Gamma^f_{kw,i} &= R^{kw}_n \Gamma^f_{i,kw}, \\ \Gamma^s_{kw,i} &= \Gamma^{kw}_{\text{ext}} + \left(1 - R^{kw}_n\right)\left(1 - f^{kw}_{\text{abs}}\right)\Gamma^f_{i,kw} + \left[R^{kw}_n + \left(1 - R^{kw}_n\right)\left(1 - f^{kw}_{\text{abs}}\right)\right]\Gamma^s_{i,kw}. \end{aligned} \tag{22}$$

It can be seen from Eq. (22) that only the directly reflected neutrals of the fast energy group are credited to the fast group. The Franck–Condon neutrals resulting from the dissociation of re-emitted or gas fueling molecules go into the slow group.

Simulations with the two-energy group option have shown very good agreement with Monte Carlo and experiment [10].

3. Code implementation

The TEP methodology outlined in the previous section has been implemented into the 2-D neutral transport code GTNEUT. In this section, details of this implementation are discussed with emphasis on the input preparation, the solution methodology including code performance considerations, the output file format and the overall structure of the code.

3.1. Input

The main GTNEUT input variables are contained in the namelist *inp* which is included in the input file *toneut*. A complete list of these input variables, along with a short explanation, is included in Appendix A. An optional namelist, *inp1*, also included in the *toneut* input file, contains variables that are used in the automatic input generation for rectangular configurations. This option is discussed in more detail below in this section. If the user decides to use the DEGAS atomic rates database instead of the (default) Janev rates [12], two additional data files are required. This is discussed in more detail in Section 3.4 and in Appendix A.

The first step in creating a GTNEUT input file is to describe the geometry of the configuration. One of the strengths of the TEP methodology and its implementation into the GTNEUT code is the ability to model the

complicated and irregular geometries that are often encountered in realistic neutral transport simulation problems. To accomplish this task, GTNEUT employs a coordinate-free unstructured grid.

As discussed in Section 2, the computational domain of interest consists of a number of internal cells bounded by the core plasma and the material walls (including any pumping interfaces which are treated as wall segments with prescribed reflection coefficients). These three elements, *internal cells*, *plasma regions* and *wall segments*, constitute the building blocks of the GTNEUT input geometry, and their properties and interrelationships must be specified. A list of the most important geometry-related input parameters is shown in Table 1 (see Appendix A for a complete list).

Each geometric element is assigned a numerical index starting with the internal plasma cells (1 to *nCells*), followed by the plasma regions and the wall segments. Although a valid boundary is obviously required, it is not necessary that it consist of *both* plasma regions and wall segments. In fact, the simplest problem that GTNEUT can solve is a group of internal plasma cells bounded by vacuum, i.e., wall segments with zero reflection coefficients.

Wall segments are always one-sided, while internal cells and plasma regions can have an arbitrary number of sides specified by the value of the input variable *nSides*. The sides of each multi-sided element are numbered consecutively, with the convention that side 1 is the side whose right endpoint (i.e., the point with the largest value of the horizontal coordinate) is the bottom vertex of the cell (i.e., the point with the smallest value of the vertical coordinate) and then moving clockwise. For example, referring to Fig. 1, $lside(1,i)$ is the length of side 1 of cell i, $angle(1,i)$ is the angle between sides 1 and 2 while $angle(5,i)$ is the angle between sides 5 and 1. Lengths and angles must be specified only for internal cells and not for plasma regions.

The location of each geometric element is specified in terms of its neighbors via the 2-dimensional array $adjCell(k,i)$ which returns the index of the cell adjacent to the kth side of cell i. Referring again to Fig. 1, $adjCell(1,i) = k$ and $adjCell(4,i) = j$. This array should be specified for all geometric elements (cells, plasma regions and wall segments). A consistency check is performed by the subroutine *checkinp* after reading the input file, to ensure that the entries of the *adjCell* array are internally consistent.

After the geometry of the problem has been fully specified, the properties of the first wall must be described. Depending on the choice of wall reflection model, which is controlled by the input variable *irefl*, the user must specify either an array of wall reflection coefficients *Rwall*(1: *nWallSegm*) (*irefl* $= 0$) or the material properties of each wall segment (*irefl* $= 1$), in which case a realistic wall reflection model is employed as discussed in Section 2.4.2. The material of each wall segment is fully described by the elements of the atomic mass array *awall* and the atomic number array *zwall*. It is still possible to assign input-specified wall reflection coefficients to selected wall segments, even when the wall reflection model option has been selected (*irefl* $= 1$). This is useful for wall segments that represent pumping surfaces or vacuum interfaces, and can be accomplished by setting the value of *zwall* for the desired wall segments to a negative number (e.g., -1). For these wall segments, reflection is controlled by the input-specified array *Rwall* and not by the internal wall reflection model.

Following the specification of the geometry and the properties of the first wall, the properties of the background plasma (electron and ion temperatures and densities, as well as ion species information) must be specified for

Table 1
Important geometric input variables

`nCells`	Number of internal cells.
`nPlasmReg`	Number of plasma regions.
`nWallSegm`	Number of wall segments.
`iType(i)`	Type of geometric element i (0 for internal cell, 1 for plasma region and 2 for wall segment).
`nSides(i)`	Number of sides for internal cell i.
`lside(k,i)`	Length of kth side of internal cell i.
`angle(k,i)`	Angle between sides k and $k+1$ of cell i.
`adjCell(k,i)`	Index of element (internal cell, wall segment of plasma region) that is adjacent to the kth side of cell i.

each internal cell as well as for the core plasma cells. The relevant input variables and their units are described in Appendix A.

Finally, recycling and other neutral gas sources must be specified. The most common option is to specify the external flux Γ_{ext}^{kw} (*#particles/s*), which corresponds to the GTNEUT input array *g_ex*, for all wall segments that are expected to have finite recycling or gas puffing sources. A non-zero value of *g_ex* for wall segment *kw* causes GTNEUT to launch neutrals with energy equal to *eneut* from that wall segment into the adjacent cell. Alternatively, an ion flux striking a wall segment can be specified (*g_ion*). GTNEUT then assumes that all ions are recycled as neutrals with energies determined by the wall reflection model and the wall temperature array *twall*. As discussed in Section 2.4.2, the energy *twall* should be characteristic of Franck–Condon atoms, i.e., a few electron volts. This option is most useful when GTNEUT is coupled with an edge fluid code, in which case the fluid code should provide the *g_ion* array, or when experimental data for ion fluxes are available. Volumetric sources (such as plasma recombination) can also be prescribed in each internal cell, using the array *S_ext*. The energy of these volumetric neutrals is specified by the input variable *eneut_v*.

3.1.1. Automatic input generation

The flexibility of the coordinate-free geometry specification model of GTNEUT has one drawback: manual input preparation, where the lengths, angles and relative positions of the sides of each cell with respect to its neighbors must be individually specified, can be laborious and error prone, especially for new geometries and configurations. For this reason, programs that can generate part of or the entire *toneut* input file using information generated by another application have been developed. Such automatic input generation interface routines have been developed for the plasma edge fluid code UEDGE [22], the MHD Equilibrium and Fitting code EFIT [23] and the DEGAS Monte Carlo neutral transport code [1]. Although these routines are considered research tools and are not part of the standard GTNEUT distribution, the GTNEUT developers will assist potential users to develop interfaces for their applications.

In addition, and in order to facilitate testing and benchmarking of the code, an internal optional automatic input capability has been implemented in GTNEUT. This option, which is activated if the input variable *i_inp* is set equal to 1, generates a rectangular NX $\times$ NY grid where NX and NY are the number of cells in the horizontal and vertical directions respectively. The background plasma parameters (n_e, T_e, etc.) can be either fixed or have top-to-bottom or left-to-right linear variation. The input variables specifying this automatic grid generation are contained in the namelist *inp1*, which is also included in the main input file *toneut*. A detailed description of these input variables is included in Appendix A, and a sample problem is discussed in Section 4.1.

3.2. Solution methodology and performance

The linear system of equations for the interface fluxes of the internal cells described by Eq. (17), has the form:

$$\mathbf{A} \cdot \boldsymbol{\Gamma} = \mathbf{S} \tag{23}$$

where $\mathbf{A}$ is a coefficient matrix with elements consisting of the various transmission coefficients and escape probabilities, $\boldsymbol{\Gamma}$ is the vector of the unknown interface fluxes and $\mathbf{S}$ is the source vector consisting of the various recycling ion and neutral fluxes and any volumetric source terms. Knowledge of the interface fluxes following the solution of the linear system of equations (Eq. (23)), allows us to construct all other quantities of interest such as the fluxes from wall segments and core plasma regions, the neutral densities in each cell, ionization rates, etc.

Since each side of each internal cell contributes one equation for the interface flux from this cell into the cell adjacent to this side, the total number of equations and unknowns (*nEqs* in the code) is equal to the total number of sides of all the internal cells. Therefore, the coefficient matrix $\mathbf{A}$ has dimensions *nEqs* $\times$ *nEqs*.

While in our discussion of the TEP theory in Section 2 the interface fluxes were introduced as $\Gamma_{i,j}$, with the indices i and j denoting the origin and destination of the neutrals represented by each flux (*from* cell i *into* the adjacent cell j), the fluxes in the code implementation are defined with respect to the side of cell i that they traverse

to enter the adjacent cell j. The correspondence between the theoretical and internal code fluxes is therefore given by:

$$\widetilde{\Gamma}_{i,k} \quad \Rightarrow \quad \Gamma_{i,adjCell(k,i)} = \Gamma_{i,j} \tag{24}$$

where the tilde in the first flux denotes a GTNEUT flux, and the assumption that cell j is adjacent to the kth side of cell i has been made. This choice significantly reduces the storage requirements for the fluxes which otherwise would have to be dimensioned as *nCells* $\times$ *nCells*, with most elements being zero. The vector of the unknown fluxes $\boldsymbol{\Gamma}$ in Eq. (23) is arranged as:

$$\boldsymbol{\Gamma} = \begin{Bmatrix} \widetilde{\Gamma}_{1,1} \\ \widetilde{\Gamma}_{1,2} \\ \vdots \\ \widetilde{\Gamma}_{1,nSides(1)} \\ \vdots \\ \widetilde{\Gamma}_{i,k} \\ \vdots \\ \widetilde{\Gamma}_{nCells,1} \\ \vdots \\ \widetilde{\Gamma}_{nCells,nSides(nCells)} \end{Bmatrix} = \begin{Bmatrix} \Gamma_{1,adjCell(1,1)} \\ \Gamma_{1,adjCell(2,1)} \\ \vdots \\ \Gamma_{1,adjCell(nSides(1),1)} \\ \vdots \\ \Gamma_{i,adjCell(k,i)} \\ \vdots \\ \Gamma_{nCells,adjCell(1,nCells)} \\ \vdots \\ \Gamma_{nCells,adjCell(nSides(nCells),nCells)} \end{Bmatrix}. \tag{25}$$

The coefficient matrix **A** is very sparse since each equation, corresponding to a single row of matrix **A**, involves only fluxes from contiguous cells. For example, a typical problem with 100 four-sided cells results in 400 equations and unknown fluxes. However, only 1960 of the 160 000 elements of the coefficient matrix **A** (1.2%), are non-zero. Therefore, the use of efficient sparse linear solvers in GTNEUT is essential for increased performance and reduced storage requirements.

GTNEUT employs the Unsymmetric-pattern Multifrontal Package UMFPACK to solve the system of linear equations of Eq. (23). UMFPACK is a set of routines for the direct solution of sparse linear systems using the unsymmetric multifrontal method [24]. The UMFPACK Fortran version 2.2.1 used by GTNEUT is functionally equivalent to the routine MA38 from the Harwell Subroutine Library (HSL). The UMFPACK routines can be obtained from http://www.cise.ufl.edu/research/sparse/umfpack.

An alternative, non-sparse, linear solver based on the widely available LAPACK library [25] is also included in GTNEUT. This was implemented so that potential users can still run the code without having to install the UMFPACK library. The user can select which solver to use by setting the input variable *i_solver* to the appropriate value (see Appendix A). For small problems with fewer than 100 cells, the performance of the non-sparse solver is comparable to that of the UMFPACK, although it drops rapidly as the size of the problem increases. However, the storage requirements of the non-sparse solver increase quadratically with the size of the problem, limiting its practical use to problems with fewer than 600 cells, depending on the hardware platform.

Users can substitute their favorite sparse matrix solver in place of UMFPACK, by replacing the *solvers* routine by their own and by modifying the sparse matrix storage scheme in the *setup* routine, to be consistent with the storage scheme of their sparse matrix solver.

3.2.1. Extension to multi-species

The present version of GTNEUT can handle only one neutral species. However, adding more species to the code (other hydrogenic atomic or molecular species, helium, impurities and even excited states of the same species) is straightforward and can be accomplished by following these steps:

(a) extend the flux vector $\boldsymbol{\Gamma}$ (Eq. (25)) to include the fluxes of the new species. The new fluxes satisfy balance equations similar to Eq. (17), with possibly additional terms describing interactions between the various species (e.g., charge exchange between neutral carbon and background hydrogenic ions). Interaction terms will affect the balance of the previously installed species as well and appropriate terms should be added to their balance equations;
(b) modify the routine *calcmfp* to calculate the mean-free-paths and related parameters (charge exchange fractions, etc.) of the new species. Routines providing the various atomic and molecular rates for the new species should also be provided;
(c) calculate the first-flight transmission coefficients and escape probabilities of the new species. Since the transmission coefficients depend only on the neutral mean free path and the geometry, this computationally expensive step can be avoided by interpolating (or using a table lookup) the already computed transmission coefficients;
(d) modify the routine *setup* to define the new elements of the sparse coefficient matrix and modify existing elements by adding inter-species interaction terms; and
(e) include any new contributions to the ionization rate (Eq. (18)).

3.2.2. Performance considerations

One of the advantages of the TEP method is its computational speed. Our benchmarking simulations have indicated that GTNEUT is faster than Monte Carlo by one to two orders of magnitude [17]. To maintain and improve this computational speed advantage as the code is upgraded and new features and capabilities are added, it is important to identify the most computationally intensive parts of the code. While reasonable effort has been made to select computationally efficient algorithms during the development of the code, most of our emphasis has been on improving the TEP methodology and implementing new capabilities rather than on numerical optimization per se. Therefore, significant room exists for improvements to further increase the computational speed of GTNEUT.

The code has been profiled under various computing platforms and compilers. This profiling revealed that more than 85% of the computation is spent calculating the first-flight transmission coefficients (Eq. (3)) and the associated Bickley–Naylor function (Eq. (4)). Standard methods are employed for the numerical evaluation of the multidimensional integral of Eq. (3), which is treated as a product of one-dimensional integrals. Gauss–Legendre quadratures are used for the spatial integration along the ξ coordinate [26], while the angular part is evaluated using the Simpson rule. The Bickley–Naylor function, Ki_3, is evaluated from an approximate fit [27] and not by direct integration of Eq. (4). The performance of the integral evaluation can be improved by employing multidimensional quadrature algorithms or quasi-Monte Carlo methods [28,29], or by evaluating the integrals using mean-chord-length approximation techniques [30].

In cases where GTNEUT is coupled to a plasma fluid code, performance can be improved by pre-computing the various transmission coefficients for a range of mean free paths during the first call to GTNEUT, and then employing table lookup interpolation to obtain the desired coefficients for subsequent calls. This approach assumes that the geometry remains unchanged during the coupled plasma-neutrals simulation while the properties of the background plasma change, in which case the transmission coefficient integrals depend only on the neutral mean free path. Finally, since the evaluation of each transmission coefficient integral depends only on the geometry and on the background plasma parameters (which determine the neutral mean free path), parallelization of this part of the code for appropriate platforms is relative straightforward with significant potential savings in computation time.

3.3. Output

The main GTNEUT output file is the text file *neut.out*. It contains the neutral density, total ionization rate and ionization rate density in each cell as well as the total, uncollided and collided fluxes at each interface. The order

in which these quantities are printed out is controlled by the input integer array *prntOrdr*. If the first element of this array is negative, then the various quantities are printed in their natural order, i.e., following the internal cell index numbering scheme. Since the numbering of the cells is more or less arbitrary, especially for automatic input generation cases, the default printing order can be overridden by specifying the elements of the *prntOrdr* array. This allows the user to group together regions of interest, as well as to facilitate the plotting of the data.

At the end of the *neut.out* file, the results of a global particle balance are included. This particle balance, performed by calling the subroutine *pbalance*, evaluates and lists the total number of particles entering the solution region and the total number of particles lost via ionization, escape from the system or wall capture. It then evaluates the relative error, which is a measure of the roundoff error of the simulation.

An optional output file, *neut.dbg*, is also generated if the debug input flag, *idbug*, is equal to 1. This file contains the values of several important parameters for each cell (*geometric details, neighbors, reaction rates, mean free paths, transmission coefficients and escape probabilities*) as well as the non-zero elements of the coefficient matrix and source vector. The *neut.dbg* file is very useful for troubleshooting GTNEUT simulations, but it can become quite large for problems with a large number of cells.

Finally, the file *umferr.dat* includes any error messages or warnings from the UMFPACK library. This file exists only if the sparse matrix option has been selected (*i_solver* $= 1$).

3.4. Code structure

GTNEUT has a simple structure, consisting of a main routine and a number of subroutines and functions. The subroutine calling sequence with a brief description of each routine is shown in Table 2.

4. Test problems

The GTNEUT code has been used in several neutral transport simulations, ranging from artificial model problems devised to test aspects of the TEP methodology and to perform benchmarks with Monte Carlo, to realistic modeling of fusion-relevant configurations and analysis of experiments [7–10]. Two such problems have been included in the GTNEUT distribution to help potential users become familiar with the code, and are described in this section.

4.1. 5×4 *rectangular geometry model problem*

The first problem consists of a 20 cell rectangular configuration (5×4) created by using the automatic input generation capability of GTNEUT. Fig. 4 shows the problem geometry, including the cell and wall segment indices. A uniform background plasma is assumed with $n_e = n_i = 10^{19}$ m^{-3} and $T_e = T_i = 10$ eV. The 1.0 m $\times$ 0.8 m rectangular region is bounded by carbon walls on three sides, while a vacuum interface is imposed on the right vertical boundary (wall segments 10–13). A unit strength (1 #/s) surface source of 2 eV deuterium neutral atoms is imposed on wall segments 2 and 3. The mean free path for neutrals in the fast energy group is $\lambda = 0.11$ m, resulting in a mean free path to cell dimension ratio $\lambda/\Delta = 0.55$.

The results of this simulation are included in the *neut5x4.out* and *neut5x4.dbg* output files. In Fig. 5, the neutral density is plotted versus the cell index. For comparison, the predictions of the DEGAS Monte Carlo code for this problem are also shown. It should be emphasized that the horizontal axis in Fig. 5 is categorical, representing the cell index. Therefore, the oscillatory appearance of the curve is an artifact caused by our arbitrary choice of the cell numbering scheme. A more physical representation of the results is shown in Fig. 6, where contours of constant neutral density are shown.

Table 2
Subroutine calling tree for the GTNEUT code

Routine	Brief description and subroutines called

main	The main routine. It opens the various input and output files, reads the namelists **inp** and **inp1** and calls the various subroutines to perform the TEP calculation. The following subroutines are called:
	`degasread`
	`rectinp`
	`checkInput`
	`calcrefln`
	`calcmfp`
	`escape`
	`calctransm`
	`setup`
	`solverf, solvers`
	`postsolver`
	`pbalance`
	`output`
	`zstop`

`degasread`	This routine is called only when $iatdat = 1$. It reads the files `ehr1.dat` and `cxionh.dat` containing atomic rate data from the DEGAS code. This option has been retained to facilitate benchmarks with the original DEGAS Monte Carlo code. For regular simulations, the Janev rates [12] should be used ($iatdat = 0$).
`rectinp`	This routine is called when $i\_inp = 1$, and generates the automatic part of the GTNEUT input for model problems with a rectangular grid (see Section 3.1 and Table A.3 in Appendix A).
`checkInput`	This routine checks the input variables for inconsistencies and performs a number of auxiliary tasks. It makes sure that the size of the problem does not exceed the dimensioning of various arrays, it checks for incompatible input choices, it ensures that the geometry is correct (the sum of the angles of each cell should be equal to $(n-2)\pi$ where n is the number of sides), it checks to make sure that the assignment of neighbors is consistent (if $adjCell(k,i) = j$ then $adjCell(l,j) = i$ where l is one of the sides of j). In addition, it converts units and normalizes various parameters. If it detects an error, an appropriate message is printed on the terminal.
`calcrefln`	This routine calculates various wall reflection parameters. If the wall reflection model is on ($irefl = 1$) it calls the subroutine `reflect`.
`reflect`	Calculates particle and energy reflection coefficients using fits that depend on the projectile and target properties (material, energy, etc.).
`calcmfp`	This routine calculates the neutral mean free path λ_i and charge exchange fraction c_i (Eq. (8)) in each internal cell. Depending on the value of the input parameter *iatdat*, it calls the functions `svione`, `svioni`, `svcxi` and `svefj` which are grouped together in the file `svjanev.f`, or the routines `svdegas` and `calcxswms`. It also calculates the albedo coefficient α_{pl} using the numerical fit to Monte Carlo simulation data in the function *albdfit*.
`svione, svioni, svcxi`	Functions that evaluate the electron impact ionization reactivity, the ion impact ionization reactivity and the charge exchange reactivity using the database assembled by R.K. Janev [12].
`svefj`	Evaluates the electron impact ionization rate using the older Freeman and Jones fit. Implemented for compatibility with older codes and used only when $ifjsv = 1$ and $iatdat = 0$.
`svdegas`	Evaluates various atomic rates using data from the original DEGAS code (i.e., not the more recent DEGAS-2 code [2]). Used when $iatdat = 1$.
`calcxswms`	Calculates various atomic rates using data compiled by E.W. Thomas and W.M. Stacey [13]. Used when $iatdat = 2$.
`escape`	Evaluates the first-flight and total escape probabilities P_{0i} and P_i (Eqs. (9), (10)) using a rational approximation. Evaluates the directional escape probability factor Λ_{ij}.
`calctransm`	Setup routine for the evaluation of the various first-flight transmission coefficients. Calls the subroutine `TransmCoeff`.

(*continued on next page*)

Table 2 (*continued*)

Routine	Brief description and subroutines called
`TransmCoeff`	Evaluates the first flight transmission coefficients $T^{i}_{k,j}$, $T^{i,0}_{kw,j}$, $T^{i,s}_{kw,j}$, $T^{i,f}_{kw,j}$ (see Section 2). Calls subroutine `calRectParms` and `calcRect`.
`calRectParms`	Calculates various geometric parameters that are needed in the evaluation of the first-flight transmission coefficients.
`calcRect`	Evaluates the multidimensional integral for the calculation of the first-flight transmission coefficient (Eq. (3)). Calls one of the Gaussian quadrature routines `qgauss20`, `qgauss40`, `qgauss60`, `qgauss80` or `qgauss100` depending on the value of the input variable *iquad*. Uses the real function `t_ij`.
`qgaussxxx, where xxx = 20, 40, 60, 80 or 100.`	Gaussian quadrature routines based on Legendre polynomials for the evaluation of the spatial part of the integral of Eq. (3). The index `xxx` denotes the number of integration points (20–100).
`t_ij`	Function to evaluate the integrand for the calculation of the integral of Eq. (3). Performs the angular part of the integration using a Simpson rule and evaluates the Bickley–Naylor function (Eq. (4)) using either a fit (for Ki_3) or direct integration (Ki_4). Calls the subroutine `simpson` and the function `bickley`.
`simpson`	Evaluates the integral of a function using Simpson's rule.
`bickley`	Evaluates the Bickley–Naylor functions Ki_3 and Ki_4.
`setup`	Evaluates the non-zero elements of the coefficient matrix **A** and the source vector **S** (Eq. (23)) and stores them in the sparse vector `a_sparse` and index vector `i_sparse` using the UMFPACK storage scheme.
`solvers`	Calls the UMFPACK routines to solve the linear system of equations defined by Eq. (23) if the input variable $i\_solver = 1$.
`solverf`	Calls the LAPACK routine DGESV to solve the linear system of equations defined by Eq. (23) if the input variable $i\_solver = 0$. This option is limited to small problems.
`postsolver`	Evaluates the various fluxes, neutral densities, ionization rates, etc.
`pbalance`	Performs a global particle balance and estimates the round off error of the simulation.
`output`	Writes information and data to the various output files.
`zstop`	This routine is called when we need to terminate the run and write a short message to the terminal.

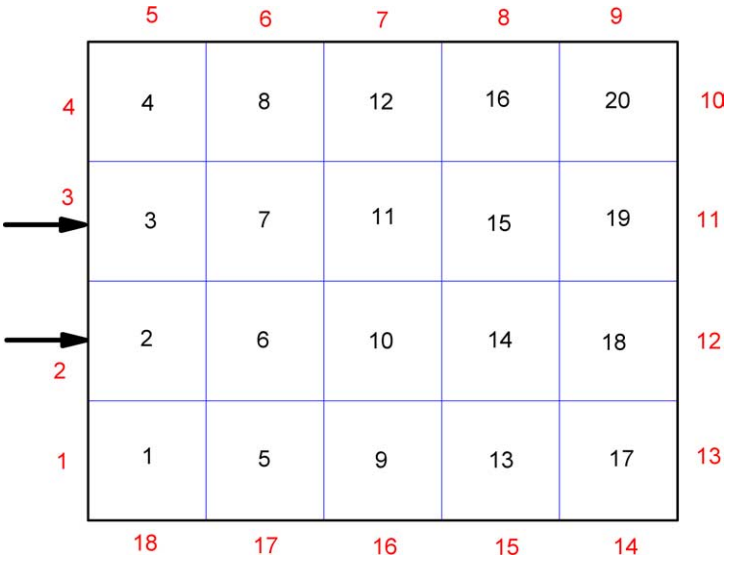

Fig. 4. Geometry configuration for the first test case.

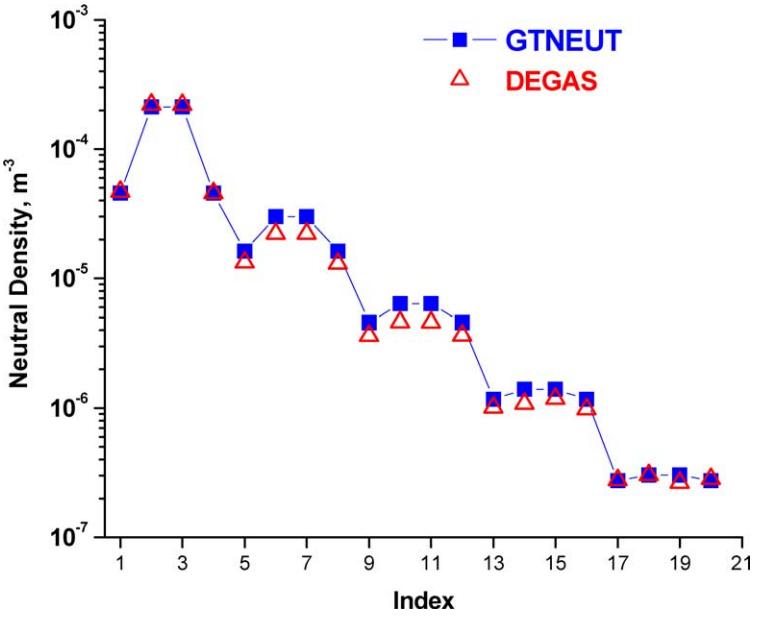

Fig. 5. Neutral density vs. cell index for the 5 × 4 test problem shown in Fig. 4. DEGAS results are also shown for comparison.

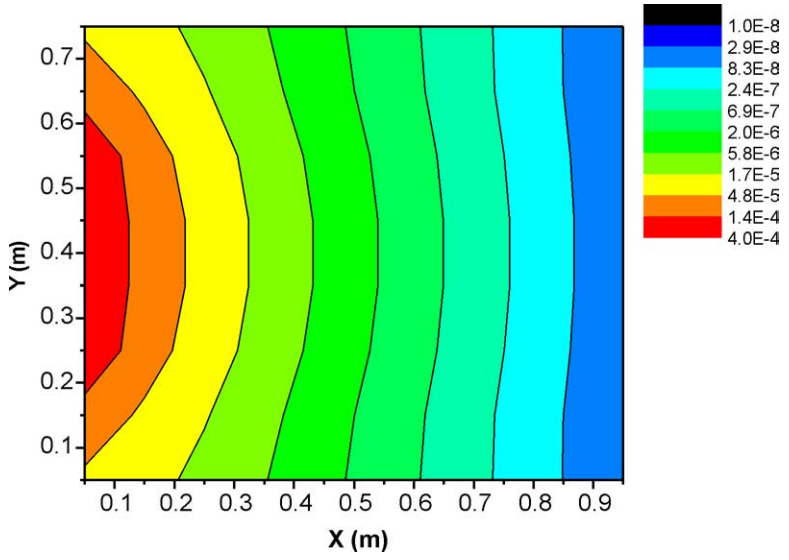

Fig. 6. Contours of constant neutral density for the test problem of Fig. 4.

4.2. DIII-D test problem

The second test case included in the GTNEUT distribution, is a realistic geometry neutral transport simulation based on a recent discharge of the DIII-D tokamak experiment [31]. The geometric part of the GTNEUT input has been generated directly from the EFIT MHD equilibrium information [23], using an interface routine that was

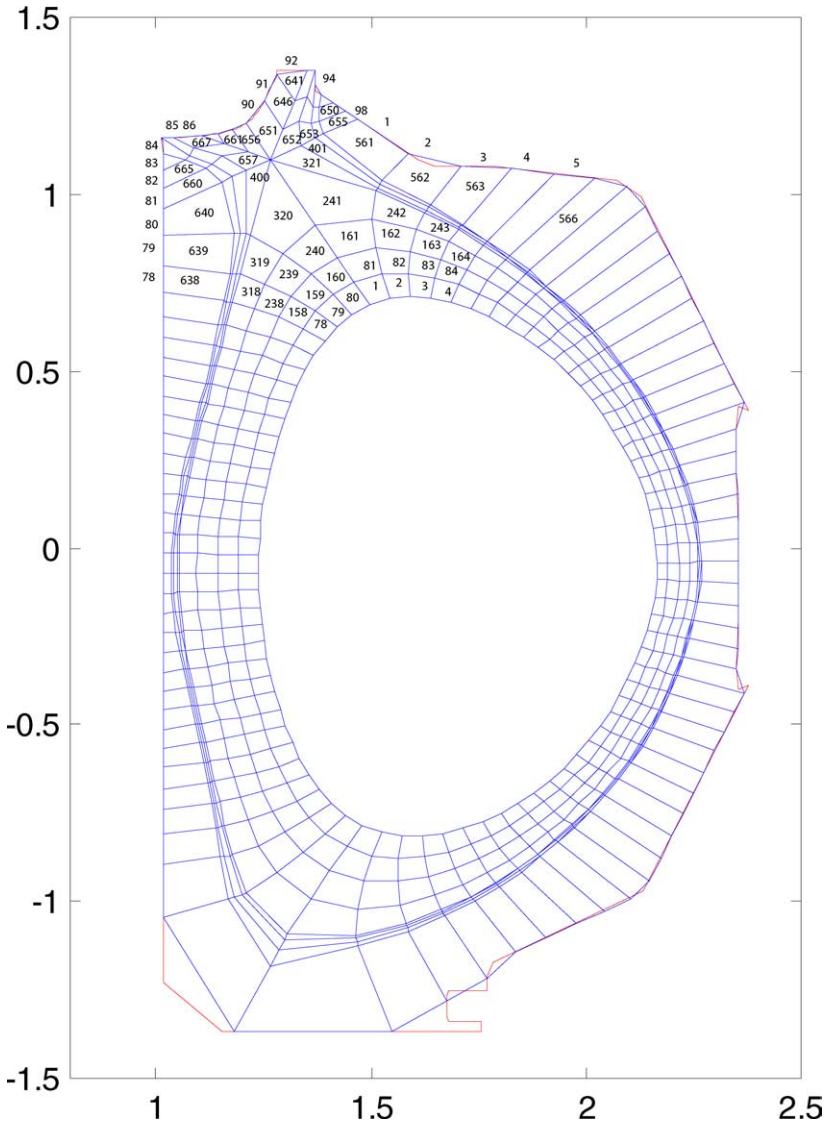

Fig. 7. Geometry of DIII-D test problem. Wall segments 84, 86, 93 and 95 represent the divertor plates where recycling is taking place, while segments 85 and 94 represent the two pump openings.

developed to facilitate input preparation. The resulting geometric configuration of the Upper Single Null (USN) plasma consisted of 670 cells, 90 core plasma regions and 98 wall segments and is shown in Fig. 7.

Wall segments 85 and 94 represent the two DIII-D upper pumps (dome and baffle pumps respectively) and are assigned zero reflection coefficients. The rest of the wall segments are assumed to be made of carbon. Neutrals and ions are assumed to recycle at the divertor plates, represented by wall segments 84, 86, 93 and 95. External neutral particle sources, *g_ex*, equal to 2.5×10^{22} #/s are imposed on each of these four wall segments, so that the total number of neutral atoms injected into the system is equal to 10^{23} #/s. The properties of the background plasma are based on preliminary experimental measurements.

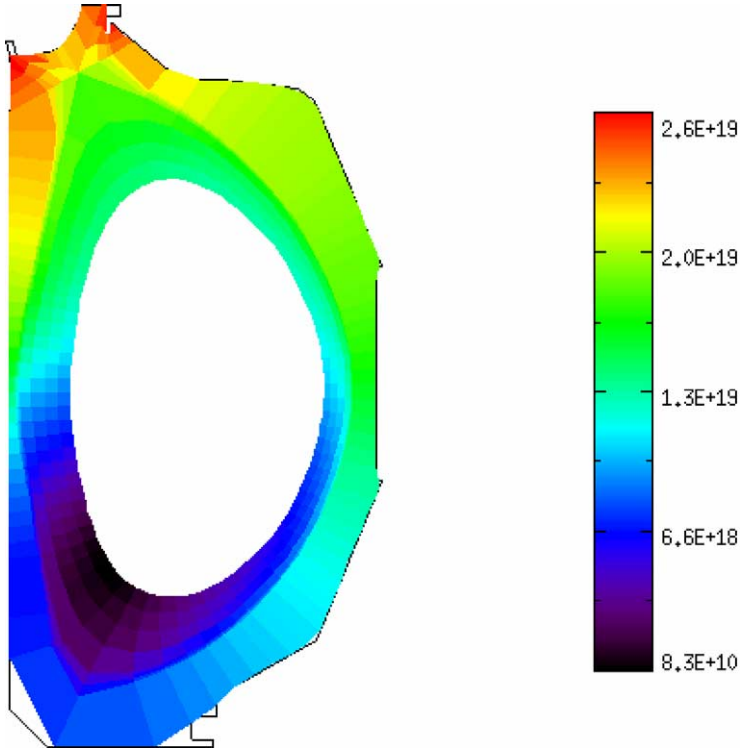

Fig. 8. Neutral density distribution for the DIII-D test case.

It should be emphasized that the DIII-D test problem is included to demonstrate the capability of our code to handle realistic geometries relevant to fusion experiments, and is not supposed to represent a comparison between theory and experiment since no neutral density measurements were available for this discharge.

The results of this simulation are included in the output file *neutDIIID.out*. The neutral density distribution is shown in Fig. 8. It can be seen that the neutral density peaks at the top of the cross section, near the divertor plates. It can also be seen that the neutral density attenuates rapidly as we move inside the separatrix, since the densities of the confined plasma are higher.

It is worth pointing out that our solution region extends to the actual first wall of the device. This can be very important in realistic simulations, since the almost vacuum regions near the wall (plenums) can allow direct streaming of neutrals from the divertor region to the midplane. This also highlights the advantages of the TEP methodology and the GTNEUT code, being perhaps the only non-Monte Carlo method that can handle the complex geometries encountered near the material walls as well as be valid in both long and short mean free path regimes.

5. Conclusions and discussion

The TEP methodology and its implementation in the 2-D neutral transport code GTNEUT have been discussed. The main advantages of the GTNEUT code are its computational speed and its ability to accurately treat the transport of neutral particles in regions with complex geometries and strongly varying mean free paths. In addition, since the TEP methodology is deterministic, GTNEUT simulations are free of the numerical noise that is inherently

present in Monte Carlo neutral transport simulations. This last advantage makes GTNEUT an ideal tool for coupling with edge plasma fluid codes, where computational speed and absence of numerical noise are essential for a rapid convergence between the two parts (neutral and plasma) of the simulation.

As desirable as computational efficiency is, it should never be attained at the expense of accuracy. For this reason, extensive tests of the TEP methodology have been carried out by benchmarking the GTNEUT code against Monte Carlo for a variety of problems. These range from artificial model problems to realistic simulations of neutral transport in fusion-relevant configurations, including the analysis of neutral density experiments [8–10, 17]. These benchmark simulations have confirmed the correctness and accuracy of the TEP methodology and of the GTNEUT code in virtually all cases of practical interest. Sizeable discrepancies from Monte Carlo have only been observed in artificial model problems, especially constructed to test the limits of the applicability of the TEP methodology.

While the present version of the code is a mature computational tool for carrying out neutral transport simulations, several improvements are planned for the future. These include both extensions of the GTNEUT capability requiring little or no theoretical development and refinements in the TEP methodology.

Among the planned extensions of the capabilities of the code is the addition of multi-species capabilities including molecular species (already discussed in Section 3.2.1), the extension of our two-group energy treatment to full multi-group and the development and implementation of additional tools to facilitate input preparation.

Regarding our plans to refine the TEP methodology, our goal is to address certain issues that were identified in our tests against Monte Carlo for model problems designed to test limiting cases of the methodology. Two of the basic assumptions of the TEP methodology are the assumption of an isotropic neutral distribution function in both the inward and outward half-spaces at the interfaces between the computational regions, and the assumption of a uniform charge exchange collision source within the volume of each cell. The first assumption, also known as the DP_0 approximation, has been shown to be a good approximation since charge exchange and elastic scattering collisions tend to isotropize the neutral distribution function. However, departures from anisotropy are possible, especially in long mean free path regions where anisotropies driven by wall reflection, presence of vacuum regions, pumps, etc. would persist across regions. Extending the original DP_0 approximation to include linearly (DP_1) and quadratically (DP_2) anisotropic distributions appears to resolve this issue, as evidenced by comparisons with Monte Carlo for model problems designed to accentuate the anisotropy effects [32].

The second assumption, i.e., the uniformity of the charge exchange collision source, is embodied in the rational approximation that we employ for the first flight collision probability P_{0i} (Eq. (10)) and in the treatment of the geometry factors Λ_{ij} which lack any specific directionality, being instead proportional to the fractional perimeter of each interface. This assumption may become questionable in regions where the neutral mean free path λ is much smaller than the characteristic dimension of the cell Δ. In these regions, the first collision source is predominantly located near the incident interface, resulting in a preferential backscattering of these neutrals across that incident surface. From our benchmark simulation experience, the assumption of a uniform collision rate distribution is reasonable, as long as the ratio of the neutral mean-free-path to the characteristic dimension of the cell λ/Δ is not too small ($\lambda/\Delta \geqslant 0.25$ or so). For smaller values of this ratio, GTNEUT appears to underestimate the neutral attenuation away from the source. The discrepancy becomes significant only after several neutral mean free paths away from the source, in which case the neutral density has already attenuated by a few orders of magnitude. We do not anticipate this to be very restrictive in practice however. Our experience with using grids from edge fluid codes is that the ratio λ/Δ is usually $\geqslant 1$, since fluid codes tend to use fine grids near material surfaces in order to resolve the gradients of the background plasma parameters. Future versions of GTNEUT may employ adaptive grid technology to ensure that $\lambda/\Delta \geqslant 0.25$ or implement corrections on the calculation of P_i and Λ_{ij} to get around this limitation. Such corrections are already being developed and tested [32] and, after becoming satisfactorily validated, will be implemented in the production version of the code. Regarding the geometric factor Λ_{ij}, a better approximation for it which takes into account the field line geometry and the background ion flow field was suggested in our original paper on the TEP methodology (Ref. [7], Eq. (29)). This formulation has not been implemented in our stand-alone version of the code since the details of the

field geometry and background ion flow are not usually available. However, when GTNEUT is coupled with an edge fluid code this information should be available and more accurate expressions for Λ_{ij} can easily be implemented.

Coupling GTNEUT to edge fluid codes may also require the calculation of plasma-neutral energy and momentum source or loss terms, to be used in the energy and momentum balance equations of the fluid code. While these quantities are not explicitly evaluated in the present stand-alone version of the code, it is easy to do so using already calculated quantities. The plasma-neutral energy exchange is the easiest to evaluate, since energy is a scalar quantity and the ionization and charge exchange rates are already computed by the code. Momentum losses are more difficult to compute and would require knowledge of the background ion flow field. However, if we assume most of the fluid-neutral momentum exchange occurs during the first collision, then momentum losses can be easily implemented. The details of ion-neutral momentum exchange in subsequent collisions (according to our multi-generational model shown in Fig. 3) would be more difficult to take into account, since these collisions are treated in an average sense.

Another issue that merits discussion is the significance of our two-dimensional symmetry assumption and any limitations it might impose when modeling configurations that are not characterized by cylindrical symmetry. Since GTNEUT will most likely be used to model neutral transport in toroidal configurations, the present discussion is focused on toroidal geometries. As long as the neutral mean free path is much smaller than the characteristic dimension of the system in the toroidal direction, toroidal effects are not expected to be significant. This is supported by the results of our benchmarks with the DEGAS Monte Carlo code which showed negligible differences between the DEGAS simulations in toroidal and cylindrical geometry modes and GTNEUT [9]. However, since neutrals travel in straight-line trajectories between collisions, toroidal effects may become significant in very low aspect ratio configurations *and* long neutral mean free paths. In this case, a neutral traveling along the ignorable coordinate without interacting with the background plasma may eventually cross a cell boundary while, under our cylindrical symmetry assumption, it would have remained within its cell. This is not expected to be a significant limitation however unless the neutral mean free path becomes comparable or larger than the major radius, which is unlikely for most cases of practical interest.

In addition, even if toroidicity effects are not important in the above sense, our assumption of symmetry along the axial or toroidal coordinate makes it difficult to simulate certain configurations with neutral sources characterized by strong toroidal asymmetries (e.g., localized gas injection valves or recycling from local structures). If the toroidal extent of the sources is large or if the sources are evenly distributed at a discrete number of toroidal locations, then GTNEUT could still predict the average neutral densities using an equivalent toroidally-symmetric source and conserving the total number of injected particles.

It should be noted here that the TEP methodology can be readily extended to three-dimensional geometries, albeit at a computational cost. Such extensions have been developed for the needs of neutron transport simulations in three dimensional fuel lattices using the interface method [33]. While a 3-D version of GTNEUT has not been very high in our code-development priority list, this may change if future benchmarks and comparisons with experiments suggest that such an extension would enhance the usability of our code.

Finally, and as discussed in Section 3.2.2, the implementation of more efficient algorithms and approximations for the evaluation of the multidimensional integrals that are necessary for the calculation of the first-flight transmission coefficients will allow GTNEUT to maintain its computational speed advantage, even as new capabilities and features are being implemented.

Acknowledgements

The author wishes to acknowledge sincere thanks to W.M. Stacey, R. Rubilar and D. Zhang, whose invaluable contributions helped make the development of the GTNEUT code possible.

Appendix A. Input variables

In this appendix, the GTNEUT input variables are listed including their storage dimensions, units (if any) and a short description. Additional comments can be found in the MAIN routine.

The GTNEUT input variables are included in two namelists, **inp** (which includes most of the main input variables) and **inp1** which includes the input variables needed for the automatic rectangular grid input generation.

The dimensioning of the various arrays in the code is controlled by a number of constants defined in a PARAMETER statement in the include file `neutGlob.inc`. These constants are listed in Table A.1. The dimensioning of arrays related to the two linear solvers (UMFPACK and LAPACK) is controlled by parameter statements in the routines **solvers** and **solverf** respectively.

The main input variables of GTNEUT are included in the namelist **inp** and are described in Table A.2.

The input variables controlling the automatic input generation capability of GTNEUT (`i_inp` = 1) are included in namelist **inp1** and are described in Table A.3 below. Only the Lx, Ly, NX and NY variables are mandatory when `i_inp` = 1. The rest of the input variables, which assign background plasma parameters, sources and wall reflection coefficients assuming simple symmetries (e.g., uniform background plasma, linear horizontal or vertical variations, etc.), are optional and can be overridden by or combined with their equivalent individual cell input variables of namelist **inp**. See subroutine `rectinp` for more information.

Table A.1
Parameter constants in GTNEUT

Constant	Description
`maxCell`	Maximum number of internal cells.
`maxWall`	Maximum number of wall segments.
`maxPlas`	Maximum number of plasma regions.
`maxSides`	Maximum number of sides for cells and plasma regions.
`maxTot`	`maxCell` + `maxPlas` + `maxWall`.
`maxCPl`	`maxCell` + `maxPlas`.
`maxEqs`	Maximum number of equations = `maxCell` ∗ `maxSides`.

Table A.2
Main GTNEUT input variables

Variable and dimension	Units	Description
`i_inp`		Flag determining the input geometry 0: original coordinate-free format 1: automatic input generation for rectangular regions (see variables in **inp1** namelist).
`nCells`		Number of internal cells.
`nPlasmReg`		Number of core plasma regions.
`nWallSegm`		Number of wall segments.
`iType(maxTot)`		Type of geometric element. Must be defined for cells, core plasma regions and wall segments. Valid options are: 0: internal cell, 1: core plasma region, 2: wall segment.
`nSides(maxTot)`		Number of sides of each geometric element (wall segments must have only one side).
`lside(maxSides, maxCell)`	m	$lside(k,i)$ is the length of kth side of cell i.
`angle(maxSides, maxCell)`	degrees	$angle(k,i)$ is the angle between sides k and $k+1$ of cell i.

(*continued on next page*)

Table A.2 (*continued*)

Variable and dimension	Units	Description
`adjCell(maxSides, maxCell)`		*adjCell*(k,i) is the index of the cell that is adjacent to the kth side of cell i.
`scalFact`		Scale factor multiplier for lengths. If *scalFact* > 0, then *lside*$(k,i) =$ *scalFact* $\times$ *lside*(k,i).
`aion`	amu	Atomic mass of background ions.
`zion`		Atomic number of background ions (only needed by the wall reflection model).
`aneut`	amu	Atomic mass of neutrals (present version of GTNEUT can handle only a single, hydrogenic neutral species).
`eneut`	keV	Energy of externally launched neutrals.
`eneut_v`	keV	Energy of volumetric neutrals (*S_ext*).
`i_e0`		Flag determining the neutral energy assumptions. If *i_e0* $= 1$, then a constant neutral energy equal to *eneut* is used throughout the code. If *i_e0* $= 2$, then the local ion temperature approximation is used. GTNEUT should be normally run with *i_e0* $= 2$. The constant energy option has been retained for tests and benchmark simulations. Notice that *i_e0* $= 1$ and *irefl* $= 1$ are incompatible options and a warning is printed out.
`v0fact`		The neutral velocity is equal to $\sqrt{v0fact \times E_0/m_0}$.
`elecTemp(maxCPl)`	keV	Background plasma electron temperature. Must be specified for all internal cells and core plasma regions.
`ionTemp(maxCPl)`	keV	Background plasma ion temperature. Must be specified for all internal cells and core plasma regions.
`elecDensity(maxCPl)`	m^{-3}	Background plasma electron density. Must be specified for all internal cells and core plasma regions.
`ionDensity(maxCPl)`	m^{-3}	Background plasma ion density. Must be specified for all internal cells and core plasma regions.
`S_ext(maxCell)`	#/s	External volumetric neutral source in each cell (S^i_{ext} in Eq. (17)).
`g_ex(maxWall)`	#/s	*g_ex*(*kw*) is the external neutral particle flux entering the plasma from wall segment *kw* (Γ^{kw}_{ext} in Eq. (14)).
`g_ion(maxWall)`	#/s	*g_ion(kw)* is the ion flux striking wall segment *kw* from the adjacent edge plasma cell i ($\Gamma^{\text{ion}}_{i,kw}$ in Eq. (16)). This variable is normally used when GTNEUT is coupled with an edge fluid code.
`irefl`		Flag specifying the wall reflection model. If *irefl* $= 0$, then wall reflection is controlled by the input array *Rwall* (*kw*). If *irefl* $= 1$, then the reflection coefficient is calculated using a material-based wall reflection model. *Rwall* can still be used when *irefl* $= 1$ to model vacuum regions, etc. for wall segments having negative *zwall*. See the routine *calcrefln* for more details.
`Rwall(maxWall)`		Input-specified reflection coefficient of wall segments (R^{kw} in Eq. (15)). Used when *irefl* $= 0$ (*no material-based refection model*) or to specify vacuum or pumping wall segments when *irefl* $= 1$.
`fwabsorb(maxWall)`		*fwabsorb*(*kw*) is the absorption coefficient of wall segment *kw* (f^{kw}_{abs} in Eq. (14)). Used only when *irefl* $= 1$.
`awall(maxWall)`	amu	*awall*(*kw*) is the atomic mass of the material of wall segment *kw*. Used only when *irefl* $= 1$.
`zwall(maxWall)`		*zwall*(*kw*) is the atomic number of the material of wall segment *kw*. Used only when *irefl* $= 1$. If *irefl* $= 1$ and *zwall*(*kw*) < 0, then the reflection coefficient of wall segment *kw* is determined by the input variable *Rwall*(*kw*). Useful for setting up vacuum regions, etc.
`twall(maxWall)`	keV	*twall*(*kw*) is the temperature of wall segment *kw*, and is needed to determine the energy of the "slow" neutrals emitted from the surface during the reflection process. Since the current version of GTNEUT treats these slow neutrals as Franck–Condon atoms, *twall* should be in the range of a few electron volts.
`iatdat`		Flag determining which atomic rate library to use. Current options are: 0: Janev's database, 1: DEGAS rates, 2: Thomas/Stacey rates. The recommended option is 0.

(*continued on next page*)

Table A.2 (*continued*)

Variable and dimension	Units	Description
`ifjsv`		If *ifjsv* > 0, use the older Freeman–Jones fits for the electron impact ionization rates. This option is available only for the default (Janev) rates (*iatdat* = 0) and has been implemented for benchmarks with other neutral codes which use the Freeman–Jones rates. It is recommended to use *ifjsv* = 0 and *iatdat* = 0.
`leh0`		This flag is only relevant if *iatdat* = 1 (DEGAS rates). If equal to 1, electron impact ionization rates are density dependent. If equal to 2, they are not.
`lchex`		This flag is only relevant if *iatdat* = 1 (DEGAS rates). If $\leqslant 2$, the charge exchange rates depend on neutral energy. If = 3, the charge exchange rates do not depend on neutral energy.
`iescp`		Flag determining how to calculate the escape probability P_0. If equal to 0, use the original Wigner formulation. If equal to 1, use the modified Sauer approximation (Eq. (10)). The recommended value is 1.
`iquad`		Flag determining the number of grid points for the ξ integration for the first-flight transmission coefficient (Eq. (3)). *iquad* = 1, 20 grid points, *iquad* = 2, 40 grid points, *iquad* = 3, 60 grid points, *iquad* = 4, 80 grid points, *iquad* = 5, 100 grid points.
`nph`		Number of grid points for the angular (ϕ) integration for the first-flight transmission coefficient (Eq. (3)).
`ifrstcol`		If equal to 1, take into account first collision effects (Eq. (11))
`prntOrdr`		Array affecting the printing order of various output arrays. If *prntOrdr* < 0, then natural order is used. See subroutine *output* for more details.
`i_solver`		Flag determining which linear solver is used to solve the TEP system of equations. If equal to 0, use the LAPACK routine DGESV. If equal to 1, use the UMFPACK sparse system library.
`isparsitr`		Number of steps for iterative improvement of sparse system solution (relevant only if `i_solver` = 1).

Table A.3
Optional input variables in namelist inp1. Since the same options apply to similar variables (electron and ion densities, electron and ion temperatures) these variables are grouped together separated by a comma, although the description refers only to the first entry

Variable and dimension	Units	Description
`Lx`	m	Length of horizontal side of rectangular region.
`Ly`	m	Length of vertical side of rectangular region.
`NX`		Number of cells in horizontal direction.
`NY`		Number of cells in vertical direction.
`ne_fixed, ni_fixed`	m^{-3}	Uniform background electron density. If `ne_fixed` is negative and if `igradneh` *and* `igradnev` are equal to zero, then the electron density in each cell is specified by the values of the array `elecDens` (`inp` namelist).
`igradneh, igradnih`		If equal to1, the electron density varies linearly in the horizontal (x) direction from `ne_lft` at the left vertical boundary to `ne_rgt` at the right vertical boundary.
`ne_lft, ni_lft`	m^{-3}	Electron density at the left vertical boundary, to be used when `igradneh` = 1.
`ne_rgt, ni_rgt`	m^{-3}	Electron density at the right vertical boundary, to be used when `igradneh` = 1.
`igradnev, igradniv`		If equal to 1, the electron density varies linearly in the vertical (y) direction from `ne_btm` at the bottom horizontal boundary to `ne_top` at the top horizontal boundary.
`ne_btm, ni_btm`	m^{-3}	Electron density at the bottom horizontal boundary, to be used when `igradnev` = 1.

(*continued on next page*)

Table A.3 (*continued*)

Variable and dimension	Units	Description
`ne_top, ni_top`	m^{-3}	Electron density at the top horizontal boundary, to be used when `igradnev` = 1.
`te_fixed, ti_fixed`	keV	Uniform background electron temperature. If `te_fixed` is negative and if `igradteh` *and* `igradtev` are equal to zero, then the electron temperature in each cell is specified by the values of the array `elecTemp` (`inp` namelist).
`igradteh, igradtih`		If equal to 1, the electron temperature varies linearly in the horizontal (x) direction from `te_lft` at the left vertical boundary to `te_rgt` at the right vertical boundary.
`te_lft, ti_lft`	m^{-3}	Electron temperature at the left vertical boundary, to be used when `igradteh` = 1.
`te_rgt, ti_rgt`	m^{-3}	Electron temperature at the right vertical boundary, to be used when `igradteh` = 1.
`igradtev, igradtiv`		If equal to 1, the electron temperature varies linearly in the vertical (y) direction from `te_btm` at the bottom horizontal boundary to `te_top` at the top horizontal boundary.
`te_btm, ti_btm`	m^{-3}	Electron temperature at the bottom horizontal boundary, to be used when `igradtev` = 1.
`te_top, ti_top`	m^{-3}	Electron temperature at the top horizontal boundary, to be used when `igradtev` = 1.
`S_0`	#/s	Volumetric neutral source (same for all cells).
`r_lft, r_rgt, r_btm, r_top`		Wall reflection coefficient for left, right, bottom and top boundaries. The wall reflection model (*irefl* = 1) overrides these input coefficients.
`g_lft, g_rgt, g_btm, g_top`	#/s	External neutral fluxes at the left, right, bottom and top boundaries. These values are added to any finite entries of the *g_ex* array.
`flx_lft, flx_rgt, flx_btm, flx_top`	#/s	Ion fluxes at the left, right, bottom and top boundaries. These values are added to any finite entries of the *g_ion* array.

References

[1] D. Heifetz, D. Post, D. Petravic, et al., J. Comput. Phys. 46 (1982) 309.
[2] D.P. Stotler, C.F.F. Carney, Contrib. Plasma Phys. 34 (1994) 392.
[3] D. Reiter, The EIRENE Code, Version: Jan. 92, Users Manual, EURATOM-KFA Institut für Plasmaphysik, Report JUL-2599 (1992).
[4] E.L. Vold, A.K. Prinja, F. Najmabadi, R.W. Conn, Fusion Technol. 22 (1992) 208.
[5] D.A. Knoll, P.R. McHugh, S.I. Krasheninnikov, D.J. Sigmar, Phys. Plasmas 3 (1996) 293.
[6] P.M. Valanju, J. Comput. Phys. 88 (1990) 114.
[7] W.M. Stacey, J. Mandrekas, Nucl. Fusion 34 (1994) 1385.
[8] W.M. Stacey, J. Mandrekas, R. Rubilar, Fusion Sci. Technol. 40 (2001) 66.
[9] R. Rubilar, W.M. Stacey, J. Mandrekas, Nucl. Fusion 41 (2001) 1003.
[10] J. Mandrekas, R.J. Colchin, W.M. Stacey, et al., Nucl. Fusion 43 (2003) 314.
[11] E.E. Lewis, W.F. Miller Jr., Computational Methods of Neutron Transport, ANS, La Grange Park, 1993.
[12] R.K. Janev, et al., Elementary Processes in Hydrogen–Helium Plasmas, Springer-Verlag, Berlin, 1987.
[13] E.W. Thomas, W.M. Stacey, Phys. Plasmas 4 (1997) 678.
[14] D.R. Schultz, S.Yu. Ovchinnikov, S.V. Passovets, in: R.K. Janev (Ed.), Atomic and Molecular Processes in Fusion Edge Plasmas, Plenum Press, New York, 1994, p. 279.
[15] E.P. Wigner, E. Creutz, H. Jupnik, T. Snyder, Appl. Phys. 26 (1955) 260.
[16] A. Sauer, Nucl. Sci. Eng. 16 (1963) 260.
[17] R. Rubilar, Neutral particle transport in the plasma edge and divertor region, Ph.D. Thesis, Georgia Inst. of Technology, Atlanta, GA, 2000.
[18] P. Rafalski, Nucl. Sci. Eng. 19 (1964) 378.
[19] P.C. Stangeby, The Plasma Boundary of Magnetic Fusion Devices, Institute of Physics Publishing, Philadelphia, 2000.
[20] W. Eckstein, J. Nucl. Mater. 248 (1997) 1.
[21] E.W. Thomas, R.K. Janev, J. Smith, J. Nucl. Instrum. Methods B69 (1992) 427.
[22] T.D. Rognlien, et al., Contr. Plasma Phys. 34 (1994) 362.
[23] L.L. Lao, J.R. Ferron, R.J. Groebner, et al., Nucl. Fusion 30 (1990) 1035.
[24] T.A. Davis, I.S. Duff, ACM Trans. Math. Software 25 (1999) 1.
[25] E. Anderson, Z. Bai, C. Bischof, et al., LAPACK Users' Guide, third ed., SIAM, Philadelphia, 1999. Library can be obtained from http://www.netlib.org.

[26] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, Dover, New York, 1972.
[27] F.G. Lether, J. Quant. Spectrosc. Radiat. Transfer 43 (1990) 187.
[28] B.D. Keister, Comput. Phys. 10 (1996) 119.
[29] A. Papageorgiou, J.F. Traub, Comput. Phys. 11 (1997) 574.
[30] J.M. Sicilian, M.M. Anderson Jr., Nucl. Sci. Eng. 57 (1975) 78.
[31] J.L. Luxon, Nucl. Fusion 42 (2002) 614.
[32] D. Zhang, J. Mandrekas, W.M. Stacey, Contrib. Plasma Phys. 44 (2004) 45.
[33] G. Marleau, R. Roy, A. Hébert, Nucl. Sci. Eng. 104 (1990) 209.