# The EIRENE Code User Manual

## including: B2-EIRENE interface

**Manual version: 1.0.0**
**(The manual version is synchronised with the code and licence (EPL) versions)**
https://www.eirene.de/EPL

**Produced on the April 28, 2023**

*0riginal author:*
**D. Reiter**

*Current EIRENE-NGM contact:*
**D.V. Borodin,**
https://www.eirene.de/Contact/contact.html

**Institut für Energie- und Klimaforschung – Plasmaphysik**
**Forschungszentrum Jülich GmbH**
**P.O.B. 1913**
**D-52425 Jülich, GERMANY**

*Since April 2023 the manual is maintained via*
https://jugit.fz-juelich.de/eirene/eirene/Manual

# EIRENE

## THE GREEK GODDESS OF PEACE

The Greek author Hesiodos wrote a genealogy of the gods, The Theogony, in the 8th century B.C.. According to him, EIRENE and her sisters, Eunomia and Dike, were the daughters of Zeus and Themis. These sisters were called the horae, the Greek word for the right time, hora.

## Abstract

The EIRENE neutral particle (incl. photons) transport code [**kn:Reiter92a**] is described. This code resorts to a combinatorial discretization of general 3 dimensional computational domains. It is a multi-species code solving simultaneously a system of time dependent (optional) or stationary (default) linear or non-linear kinetic transport equations of almost arbitrary complexity. A rather crude model for transport of ionized particles in an externally specified magnetic field lines is also included. EIRENE is coupled to external databases for atomic and molecular data and for surface reflection data, and it calls various user supplied routines, e.g. for exchange of data with other (fluid-) transport codes. The main goal of code development was to provide a tool to investigate neutral gas transport in magnetically confined plasmas. But, due to its flexibility, it also can be used to solve more general linear kinetic transport equations, by applying a stochastic rather than a numerical or analytical method of solution. In particular, options are retained to reduce the model equations to the theoretically important case of the one speed transport problem.
Major applications of EIRENE are in connection with plasma fluid codes, in particular with the various versions of the B2 code [**kn:Braams**]. The semi-implicit iterative coupling method of B2-EIRENE [**kn:Reiter91b**], [**kn:Reiter92b**] and it's implementation (code segment: EIRCOP) are also described. Both, in its stand alone form and also non-linearly integrated into plasma fluid transport solvers the EIRENE code has become a widely spread "community code" since the nineties of the last century. A short overall summary of the code package, including some historical notes, is given in [**kn:Reiter2004**].

# Foreword to 2nd edition

The first edition of this EIRENE code user manual was published as KFA report JÜL-2599 in March 1992, [**kn:Reiter92a**]. It basically was a collection of my notes on the meaning of the various input data options for EIRENE. Over the years too many such input flags had accumulated to memorize the meaning of each individual one.
Since the EIRENE code became a quite popular tool also for many other colleagues, it was decided to provide these notes as a kind of "user's manual" in a somewhat completed and edited form.
In the meantime the distribution of the EIRENE code has become even wider, and it seems timely to update the previous manual, although most of it's content is still a relevant source of information for a third party application of the code. Apart from several minor corrections, e.g., of spelling errors and unclear language, the major new features as compared to the previous edition are the following:

- time dependent mode + snapshot estimators (section 2.13)
- internally consistent "I-integral" approach for coupled neutral (kinetic) - plasma (fluid) simulations (partially disabled again shortly after its implementation, because of spurious oscillatory behaviour of some derivatives of polynomial fits for rate coefficients)
- individual background "ion" temperatures TI(IPLS) for each background species (e.g., for neutral background particles)
- simulation of self collisions in **BGK!** (**BGK!**) approximation
- some new collision processes and data, such as net re-combination energy losses (radiation + potential), elastic neutral - ion collisions, multi-step break-up of molecules
- RAPS graphics interfaces
- user defined geometry block. In this new mode of operation, geometry level LEVGEO=10, EIRENE knows nothing (not even the dimensionality) about the geometrical aspects of the problem. Everything concerning grids, cell volumes, flight times within cell etc. is in user supplied routines. No default graphics options are available, of course, in this mode of operation. (This option was developed and is used in context with EMC3-EIRENE since about 1999)
- Particle tracing routines FOLATM and FOLMOL for neutral atoms and neutral molecules are combined into one single routine FOLNEUT for neutral particles.

  The particle tracing routine FOLION for charged test particles has been updated considerably, including now a simple approximation to the Fokker Planck collision operator (a "Krook collision operator") in 1997.

An increasing number of EIRENE 3rd party applications is running cases in which EIRENE is coupled to a fluid (**CFD!** (**CFD!**)) plasma model. Here it provides sources and sinks due to

interaction of "kinetic sub-components" with the macroscopic flow, which are iteratively coupled ("operator splitting").
The first such coupled **CFD!** - EIRENE code was SOLXY-EIRENE ([**kn:Gerhauser88a**]) and had led to implementation of the so called "correlated sampling" option in EIRENE, as necessary to achieve convergence measured by comparison with solution using an analog analytic neutral particle model.
The widely used B2-EIRENE code system [**kn:Reiter91b**] is another such example of a coupled **CFD!**-Monte Carlo diffusion-advection-reaction code.
A new section describing this code coupling part of the EIRENE code has been added. It describes the "sandwich" file EIRCOP which was written to permit linkage between EIRENE and a **CFD!** plasma fluid model (one, two, or three dimensional). Initial and boundary value problems can be treated in a rather self-consistent manner.
For B2-EIRENE this code interfacing segment was developed largely under support of a KFA-EURATOM contract ([**kn:Net**]), and first applications have been published for ITER configurations, see [**kn:Reiter91b**] and [**kn:Maddison91**]. Main new features in EIRENE for this project was implementation of multiply connected 2D curvilinear grids into EIRENE, so that the **CFD!** and Monte Carlo code operate on identical meshes, without any interpolation necessary. Also the "short-cycle" option, a semi-implicit correction scheme for Monte Carlo source terms between large **CFD!** $\Delta t$ cycles, without new or at least with strongly reduced MC runs in between, was implemented.
Subsequently a large number of application runs, in particular those carried out at IPP-Garching (Ralf Schneider et al.) and AEA-Culham Laboratories (Geoff Maddison) have lead to identification of many critical issues and limitations of the code and to permanent improvements until these days. Typical convergence behaviour of the combined code system ("saturated residuals") is described in the new section 1.8, written in collaboration with G.P.Maddison, AEA Culham Laboratories.
Detlev Reiter, Spring 1998

# Foreword to 3rd edition

During the year 2001 a major revision of the EIRENE code has been carried out, largely in order to implement a somewhat more modern FORTRAN structure into the code. EIRENE code versions from this particular year 2001 then have sometimes been referred to as "EIRENE-FACELIFT" (= $\text{EIRENE}_{2001}$ in the notation of this manual).
In particular a dynamical allocation of storage (rather than the hitherto necessary pre-assignment of storage in the PARAMETER statements collected in the previous PARMUSR file) is now in place. This dynamic memory management has led to a replacement of the previous "Common-Blocks" now by "Modules", and to the entire elimination of the "Equivalence" statements used for storage economy in earlier versions.
Further main upgrades are related to:

- incident-species dependent surface interaction models, pumping speeds, sputter models etc., can now be controlled by input flags, rather than the previous quite difficult implementation via the USR-routines.

- integration volumes of volume averaged tallies can now be larger than the grid cell volumes, i.e., the plasma background and geometrical discretization, which is not very storage sensitive, can be made much finer than the neutral particle volume averaged tally profile output.
- discretization of general multiply connected 3D volumes by tetrahedron-grids.
- photons as new type of test particle species. The photon_dummy module allows simple photon transport (one-speed Boltzmann-equation) with spatial profiles of spontaneous emission, Doppler line broadening, for optically thin lines only, with inclusion of (multiple) surface reflection. The full photon module additionally allows for various other line broadening mechanisms and photon re-absorption, stimulated emission, radiation trapping (excitation-hopping), iteration with neutral gas excitation population kinetics, etc.
- The code interface module EIRCOP has been supplemented by another quite similar program, which, however, is based upon a general (unstructured) discretization of 2D geometries by triangles. With this option the non-linear options (neutral–neutral and neutral-photon interactions), have become accessible to the iterative B2-EIRENE mode of operation (e.g. forming the version SOLPS4.3 currently used for ITER and DEMO divertor design studies).

EIRENE versions 2003 and younger have been compiled and successfully tested at FZ-Jülich on the following systems and compilers:
Linux: Suse 11.1 and all predecessors down to Suse 6.***:

- pgf90 Portland Compiler Version 5.2-4 — 10.1
- ifort Intel Fortran Compiler Version 8.1 — 11.1
- lf95 Lahey Fortran Compiler Version 6.2
- nagf95 NAG Fortran Compiler Version 5.0 — 5.2

AIX: AIX 4.3 and 5.2

- xlf95 XL Fortran for AIX Version 8.1

Windows: Windows 2000 und Windows XP

- Compaq Visual Fortran Version 6.6

Detlev Reiter, Spring 2010

# Contents

# Chapter I

# Introduction and General Information

This manual describes the input required by the EIRENE code to run a Monte Carlo study for a fully 3 dimensional simulation of linear transport (i.e., of test particles) in a prescribed background medium. Although the geometry of the problem and the interaction between test particle species and the background are in principle not subject to any restrictions, the aim of code development was to provide a tool for investigating neutral gas transport in tokamak plasmas. Consequently the choice of preprogrammed options has been made mainly with this application in mind. A large variety of problems in this field can be run without having to resort to any problem specific routines but instead by an appropriate setting of logical and numerical input flags.
Any user of EIRENE should be aware that this code is a moving target as is this manual.
Therefore it is possible that there are some inconsistencies between this description and a particular version of the code. The user should always first check subroutine INPUT, where most of the data are read in using hard wired formats, as most input errors will lead to a rapid exit in the initialization phase of a run.
This manual was written by the author of the code who often may not have been able to anticipate difficulties in understanding the use of EIRENE. He therefore gratefully acknowledges any suggestions to make this manual more informative and clear than it might be at the present status.
This code description consists of the following parts:

- In the first part an introduction is given to the general linear transport problem and its solution by Monte Carlo methods. Most of the terminology used in later sections is introduced there.

- In part two a description of the formatted input file required by EIRENE to run on a specific problem is given. It mainly consists of explanations of the meanings of the various input flags.

- In part three the most important problem specific routines are explained. At present we have restricted this part to routines for evaluation of "user requested tallies", namely the subroutines UPTUSR, UPCUSR and UPSUSR. Other often applied routines such as SAMUSR (user supplied source sampling distribution) or REFUSR (user supplied surface interaction model) are briefly described.

- In part four the package EIRCOP for interfacing EIRENE with other codes (e.g., the B2-EIRENE package) is described. Here mainly the location of the storage on the EIRENE work array RWK for plasma data and geometrical information is given. Also the implementation of the method of (semi-implicit) corrections (see section 1.8) at each plasma

code time-step to the terms transferred from EIRENE to plasma fluid transport codes is described here.

# Chapter 1

# The neutral gas transport equation; Monte-Carlo terminology

### General Remarks

To introduce the terminology used throughout this report, we briefly recall the basic definitions and principles of a Monte Carlo linear transport model, following the lead of many textbooks on Monte Carlo methods for computing neutron transport (see e.g., [**kn:Spanier69a**]). The original derivation of the EIRENE Monte Carlo model, building on these neutronics concepts and terminology, was carried out in D. Reiter's PhD thesis (in German) [**kn:Reiter84a**] in the early eighties of the last century. We begin with the linear transport equation for the pre-collision density, written as integral equation (linear non-homogeneous Fredholm integral equation of $2^{nd}$ kind). Distinct from standard terminology in the (analytic) transport theory we do not discuss analytic properties of the various terms in the equation, but, instead, point out their probabilistic interpretation, as needed for a Monte-Carlo solution of that equation. Next (section 1.3) we sketch the Monte Carlo procedure to solve such equations, by referring to the two most often applied techniques: "track-length - and collision based estimators". In the third subsection we briefly describe the treatment of boundary conditions (models for interaction of the particles with surrounding surfaces) and discuss some special models which are in use for the neutral gas transport in fusion plasma devices. Then (section 1.5) we discuss the most important source function (non-homogeneous part of the integral equation) and its implementation in a Monte Carlo algorithm, namely the surface source of neutral particles due to recombination of ions incident on solid surfaces at the boundary or inside of the computational area. In section 1.6 we comment on the implementation of geometry within the framework of a Monte Carlo code in general terms and for the EIRENE code in particular. In section 1.7 the time dependent mode of operation of the EIRENE code is described. It merely amounts just to an increase in the dimensionality by one, by adding a time co-ordinate and treating it, formally, in a rather symmetric fashion with the other spatial coordinates. See also reference [**kn:Reiter94**].
Two kinds of non-linearities may be accounted for:
In section 1.8 the nonlinear behavior resulting from background data, which depend on neutral particle transport (sources and sinks), is discussed. The algorithm of the B2-EIRENE code system [**kn:Reiter92b**] is described. More details on this can be found in the report [**kn:Maddison94**].
In the final section of this introductory section 1.9 the non-linear **BGK!** formalism and the **DMCS!** (**DMCS!**) method for self collisions between neutral particles, as implemented in the EIRENE code, is described.

## 1.1 The linear Boltzmann equation for the distribution function $f$

EIRENE solves a multi species set of coupled "Boltzmann"-type equations, in arbitrary 3D geometries. Strictly speaking it is the Boltzmann equation generalized from its original single species form with bi-linear collision kernel for elastic collisions, to a far more complicated collision integral, which also represents "chemical reactions". This generalisation is also often referred to as **WCU!** (**WCU!**) equation in literature [**kn:WCU**]. In this present section we start with the Boltzmann equation. We then deal with linearizations of collision kernels by fixing the phase space distribution of one of the two collision partners (which we refer to as "bulk" or "background" or even simply "plasma" species). Generalization to the full **WCU!** type multi-species collision kernels, which in the same linearized form provide the mathematical description of the equations solved by EIRENE, is then discussed next, section 1.1.1.
Relaxation of the linearization assumptions is needed only when there is self-interaction amongst the species considered by EIRENE, as it is required e.g. when radiation transfer is included (coupling between "photons" and atoms) or if neutral–neutral collisions are relevant. In this latter case so called **BGK!** approximations to the full collisions integral are employed, and the non-linearity is dealt with by successive linearization (i.e. by iteration).
The strongest non-linearity in EIRENE applications is typically the back-reaction of the "plasma-background" (the host medium) on the neutral gas and radiation fields. This, however, is dealt with in an operator splitting cycling procedure (see chapter 4) between a plasma solver ("diffusion-advection sub-module" solver, for given reaction terms) and EIRENE ("reaction part" solver, for given diffusion advection solution, i.e. for given background medium), in which EIRENE, within each single cycle, may still be operated in linear mode.
The term *$\mu$-space* in this report refers to the phase space of a single particle. The quantity of interest is then the *one particle distribution function f*

$$f(\mathbf{r}, \mathbf{v}, i, t) \quad \text{or} \quad \tilde{f}(\mathbf{r}, E, \mathbf{\Omega}, i, t) \quad \text{or} \quad f(x),$$

where the state $x$ of the $\mu$-space is characterized by a position vector $\mathbf{r}$, a velocity vector $\mathbf{v}$, a species index $i$ ($i$ stands for, e.g., H, D, T, $D_2$, DT, He, $CH_n$, ...) and the time $t$.
The number density $n_i(\mathbf{r})$ for species $i$ then reads:

$$n_i(\mathbf{r}, t) = \int d^3v \; f(\mathbf{r}, \mathbf{v}, i, t)$$

Instead of $\mathbf{v}$ we sometimes utilize the kinetic energy $E$, $E = m/2 \cdot v^2$, and the unit (speed) vector $\mathbf{\Omega} = \mathbf{v}/|\mathbf{v}|$ in the direction of particle motion. Hence:

$$f(\mathbf{r}) d^3v = \tilde{f}(E, \mathbf{\Omega}) dE d\Omega \quad \text{where} \quad d^3v = v^2 dv d\Omega \quad \text{and} \quad d\Omega = \sin\theta d\theta d\phi$$

and

$$f(\mathbf{r}, \mathbf{v}, i, t) = \left(\frac{m}{v}\right) \tilde{f}(\mathbf{r}, E, \mathbf{\Omega}, i, t)$$

The distribution function in the form of $\tilde{f}$ clearly remains meaningful also in the case of massless particles (photons), i.e. for applications of EIRENE to radiation transfer problems.
The notation used now may appear to be unnecessarily complicated, but we intent to prepare here already the more general discussion in section 1.1.1 on the **WCU!** equation.

We start with the original (elastic collisions only) "Boltzmann Equation" [**kn:Cercignani**], but by assuming additionally that collisions are discontinuous events (i.e.: finite range interaction potentials, or, at least, that proper cut-off procedures are applicable). This additional assumption allows us to separate, in the Boltzmann collision integral given below, the collisional "loss" and "gain" terms into two separate integrals. Otherwise cases could arise in which only the net collision term $\iiint \ldots (f' f'_b - f f_b)$ would lead to finite results, but not the collision term in the form as given in (1.a) below: $\iiint \ldots (f' f'_b) - \iiint \ldots (f f_b)$.
Further: Lets consider only one specific species $i_0$, now sometimes omitting this species index. We assume that there are only collisions of this species $i_0$ with only one further species, here labeled $b$. Generally $b = i_0$ is also possible, but anticipating that later, when discussing **linear** Boltzmann models, these species $b$ will be referred to as background species, plasma species, etc., with an externally given (fixed) distribution $f_b(\mathbf{x}, \mathbf{v}, t)$. Restriction, for the time being, to elastic collisions means that exactly one particle of each of these two species $i_0, b$ will also be present after the collision event (i.e., chemical reactions, inelastic collisions with change of internal energy, are excluded for the time being, but their description within a "Boltzmann-like" framework is discussed below, section 1.1.1) on the so called "**WCU!**" equation.
The familiar Boltzmann equation for the distribution function $f = f_{i_0}$ for a particular species (a particular component of a gas mixture) $i_0$ reads

$$\begin{aligned}\left[\frac{\partial}{\partial t} + \mathbf{v}\cdot\nabla_{\mathbf{r}} + \frac{\mathbf{F}(\mathbf{r},\mathbf{v},t)}{m}\cdot\nabla_{\mathbf{v}}\right] f(\mathbf{r},\mathbf{v},t) = &\int\int\int \sigma(\mathbf{v}',\mathbf{v}';\mathbf{v},\mathbf{v})|\mathbf{v}'-\mathbf{v}'|f(\mathbf{v}')f_b(\mathbf{v}') \\ &- \int\int\int \sigma(\mathbf{v},\mathbf{v};\mathbf{v}',\mathbf{v}')|\mathbf{v}-\mathbf{v}|f(\mathbf{v})f_b(\mathbf{v}) \\ &+ Q(\mathbf{r},\mathbf{v},t) \end{aligned} \tag{1.a}$$

There is one such equation for each "particle species $i$" considered, but for elastic collision without exchange between species. Always only two of them (here: $i_0$ and $b$) are directly coupled to each other.
$Q$ $(= Q_{i_0})$ is any external source (particles of species $i_0$ injected per unit volume in phase space and per unit time).
Integration in the first term on the right hand side ("gain term") of this equation is over the (pre-collision) velocities $\mathbf{v}', \mathbf{v}'$ as well as over one of the two post-collision velocities: $\mathbf{v}$, the velocity of species $b$. We write more suggestively $\sigma(\mathbf{v}', \mathbf{v}'; \mathbf{v}, \mathbf{v}) = \sigma(\mathbf{v}', \mathbf{v}' \to \mathbf{v}, \mathbf{v})$ for the (multi-) differential cross section for a binary particle collision process. The product of this $\sigma$ with the relative pre-collision velocity $|\mathbf{v}' - \mathbf{v}'|$ is the transitional (collision) probability leading from velocities prior to the collision $(\mathbf{v}', \mathbf{v}')$ to the pair $(\mathbf{v}, \mathbf{v})$ of velocities after the collision.
In a multi-component gas mixture the first two arguments in $\sigma$, namely the velocities $\mathbf{v}', \mathbf{v}'$ in the first integral, correspond to the species $i_0$ and $b$, respectively. Via elastic collisions these are turned into the post-collision velocities $\mathbf{v}, \mathbf{v}$, again for the same species $i_0$ and $b$, respectively. The first integral, therefore, describes the total rate of collisional transitions into the velocity space interval $[\mathbf{v}, \mathbf{v} + d\mathbf{v}]$ for species $i_0$
In the second integral on the right hand side ("loss term") the role of pre- and post-collision velocities is exchanged, but the integration remains over the same three velocities $\mathbf{v}', \mathbf{v}'$ and $\mathbf{v}$, i.e. integration is now over all post-collision velocities and over all incident velocities $\mathbf{v}$ of collision partner $b$. Hence this integral describes the total collisional rate of loss for species $i_0$ from that

phase space interval $[\mathbf{v}, \mathbf{v} + d\mathbf{v}]$. Furthermore, $m = m_{i_0}$ is the particle mass and $\mathbf{F}(\mathbf{r}, \mathbf{v}, t)$ is a volume force field. The fact that the cross section $\sigma$ remains the same in the forward (gain) and reverse (loss) integral after exchanging the pre-collision with the post-collision velocity arguments follows from time-reversal (detailed balancing) in case of elastic collisions.
The four velocity arguments in both collision integral terms are not truly independent: the conservation laws for total energy and momentum in a collision event must be fulfilled.

$$\begin{aligned} m_{i_0}\mathbf{v}' + m_b\mathbf{v}' &= m_{i_0}\mathbf{v} + m_b\mathbf{v} \\ \frac{1}{2}m_{i_0}v'^2 + \frac{1}{2}m_bV'^2 &= \frac{1}{2}m_{i_0}v^2 + m_bV^2 + \Delta_E \end{aligned} \tag{1.1}$$

Here $\Delta_E$ is the exchange of internal energy in the collision (on expense of the energy of relative motion), and, by definition, $\Delta_E = 0$ for elastic collisions. We may assume that cross section $\sigma$ contains appropriate delta function factors such that the three integrations over velocity spaces reduce to integrations over lower dimensional manifolds on which these conservation laws are fulfilled. The modulus of the relative velocity $g$ is unchanged in elastic collisions, $g_{i_0,b} = |\mathbf{v} - \mathbf{v}| = |\mathbf{v}' - \mathbf{v}'| = g'_{i_0,b}$, and hence the collision is just a rotation of this relative velocity, by an angle $\vec{\Omega}$. 6-fold integration over full velocity spaces $\mathbf{v}', \mathbf{v}'$ in $\mathfrak{R}^6$ reduce to just the 2 fold integration over $\Omega : d\Omega = \sin\chi d\chi d\epsilon$ with $\chi$ being the deflection angle and $\epsilon$ the scattering angle.
Writing

$$D_t^{i_0} = \frac{\partial}{\partial t} + \mathbf{v}\cdot\nabla_{\mathbf{r}} + \frac{\mathbf{F}_{i_0}(\mathbf{r}, \mathbf{v}, t)}{m_{i_0}}\cdot\nabla_{\mathbf{v}}$$

for the left hand side (convective part) of (1.a), leads to the Boltzmann equation in more commonly used text book form:

$$\begin{aligned} D_t^{i_0} f_{i_0}(\mathbf{r}, \mathbf{v}, t) = &\int_{\mathfrak{R}^3,\Omega} d\mathbf{v}' d\Omega \;\; \sigma(\mathbf{v}, \mathbf{v}; \mathbf{v}', \mathbf{v}')|\mathbf{v} - \mathbf{v}| \left[f_{i_0}(\mathbf{v}')f_b(\mathbf{v}') - f_{i_0}(\mathbf{v})f_b(\mathbf{v})\right] \\ &+ Q_{i_0}(\mathbf{r}, \mathbf{v}, t) \end{aligned} \tag{1.a1}$$

and the velocities $\mathbf{v}', \mathbf{v}$ are uniquely determined by $\mathbf{v}, \mathbf{v}'$ and scattering angles $\Omega$ via mass and energy conservation (1.1).
The integral in this equation, or, equivalently, the first two terms on the right hand side of (1.a), comprise the Boltzmann collision integral $\frac{\delta f_{i_0}}{\delta t}|_{coll}$. If there are more than just one possible type of collision partners (species) "$b$", then the collision integral has to be replaced by a sum of collision integrals over all collision partners $b$, including, possibly, $b = i_0$ (self collisions)

$$\frac{\delta f_{i_0}}{\delta t}|_{coll} = \frac{\delta f_{i_0}}{\delta t}|_{gain} - \frac{\delta f_{i_0}}{\delta t}|_{loss} = \sum_b \frac{\delta f_{i_0}}{\delta t}|_{coll_b}. \tag{1.2}$$

All these collision operators are bi-linear in the distribution functions $f_{i_0}, f_b$. The first term on the right hand side is due to scattering into the element $d\mathbf{v}$ of velocity space and we shall abbreviate it by defining the collision kernel ("redistribution function") $C_{i_0}$ for $i_0$ particles alone, i.e. with an operator acting on (pre-collision) $f_{i_0}$ alone, already containing the proper integrations over pre- and post-collision velocities $\mathbf{v}', \mathbf{v}$ of species $b$ collision partners:

$$\frac{\delta f_{i_0}(\mathbf{v})}{\delta t}|_{gain} = \int d^3v' C_{i_0}(\mathbf{v}' \to \mathbf{v})|\mathbf{v}'|f_{i_0}(\mathbf{v}') = \int d^3v' \sum_b C_{i_0,b}(\mathbf{v}' \to \mathbf{v})|\mathbf{v}'|f_{i_0}(\mathbf{v}') \tag{1.3}$$

In more general situations (including chemical reactions) in addition to being an integral over particle "$b$" velocity distributions, the kernels $C$ can be quite complicated integrals due to collision kinetics, as it then involves not only multiple differential cross sections, but also, possibly, particle multiplication factors, e.g. in case of fission by neutron impact, dissociation of molecules by electron impact, or stimulated photon emission from excited atoms. It can also include absorption, in which case the post-collision state must be an extra "limbo" state outside the phase-space considered. Due to both particle multiplication and/or absorption the collision kernel $C$ is not normalized to one, generally.
The second term on the right hand side of the Boltzmann equation $\frac{\delta f}{\delta t}|_{loss}$ is, formally, much simpler, because the function $f_{i_0}(\mathbf{v})$ can be taken out of the integrals. It is often convenient to even take the product $|\mathbf{v}| \cdot f_{i_0}$ before the integral. The remaining integral is then just the total "macroscopic cross section" $\Sigma_t$ (dimension: 1/length), i.e., the inverse local mean free path. Apart from dependence on individual velocity $\mathbf{v}$ of particle $i_0$ it is solely defined by total cross sections averaged over velocity distributions of particles "$b$", i.e. by "reaction rates", and is independent of particle multiplication factors, since we only consider binary collisions (exactly two pre-collision partners always, which lead to the loss of exactly one particle of species $i_0$ from phase space).
This term is then, often, taken on the left hand side of the Boltzmann equation with a positive sign, in the form:

$$\frac{\delta f_{i_0}}{\delta t}|_{loss} = \Sigma_{t,i_0}(\mathbf{r}, \mathbf{v})|\mathbf{v}| f_{i_0}(\mathbf{v}) \tag{1.4}$$

Note, however, that in case of "self collisions" within the community of particles $i_0$, the macroscopic cross section $\Sigma_t$ itself is a (nonlinear) function of the dependent variable $f_{i_0} : \Sigma_{t,i_0}(\mathbf{r}, \mathbf{v}, f_{i_0})$, since it then contains also an integration (over velocities $\mathbf{v}$ of the collision partners) of $\sigma(\ldots, \mathbf{v} \rightarrow \ldots) \times f_{i_0}(\mathbf{v})$.

### The linear form of Boltzmann transport equation

If all distributions $f_b(\mathbf{v})$ of the collision partners $b$ of species $i_0$ are assumed to be given, then the kernel $C = C_{i_0}$ defined in (1.3) is linear and the expression given there defines a linear integral ("collision") operator acting on the distribution $f_{i_0}(\mathbf{v})$. For the linear case, i.e. for $f_b$ given for all collision partners other than $i_0$, and for self collisions ($i_0 + i_0 \rightarrow \ldots$) being excluded, also the "extinction coefficient" $\Sigma_t = \Sigma_{t,i_0}$ defined by (1.4) is independent of the dependent variable $f = f_{i_0}$, and this term (out-scattering) just describes the loss of particle flux of $i_0$ particles due to any kind of interaction of them with any species $b$ from the background "host" medium.
With these formal substitutions the Boltzmann equation for distribution function $f(\mathbf{v}) = f_{i_0}(\mathbf{v})$ takes a form which is often more convenient, in particular in linear transport theory (from now on omitting subscript $i_0$ on all terms):

$$[\frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{r}} + \frac{\mathbf{F}(\mathbf{r}, \mathbf{v}, t)}{m} \cdot \nabla_{\mathbf{v}}]\ f(\mathbf{r}, \mathbf{v}, t) + \Sigma_t(\mathbf{r}, \mathbf{v})|\mathbf{v}| f(\mathbf{v}) = \int d^3v'\ C(\mathbf{v}' \rightarrow \mathbf{v})|\mathbf{v}'| f(\mathbf{v}') + Q(\mathbf{r}, \mathbf{v}, t) \tag{1.b}$$

which, in the linear case (no self collisions, $f_b$ given externally) is a linear equation for the distribution function $f = f_{i_0}(\mathbf{x}, \mathbf{v}, t)$ for species $i_0$.

Clearly a computationally crucial simplification is provided in this linear case, which means neglect of all interactions within the community of species "$i$", retaining only: $i_0 + b \rightarrow \cdots$ events. In practice for neutral particle (species group $i$) transport in plasmas (species group $b$) this mostly means neglect of neutral–neutral interaction, retaining only neutral-plasma collisions (for given plasma conditions). For the status of options in EIRENE to deal with "non-linear" "self-collisions" ($i_0 + i_0 \rightarrow \cdots$) or with "cross collisions" ($i_0 + i_1 \rightarrow \cdots$) amongst the species for which the kinetic equation is solved, see below, section 1.9. Note that also combined (consistent) neutral particle and radiation transport models fall into the class of non-linear Monte Carlo applications, except for optically thin media in which case the photon gas and the neutral particle gas are decoupled.

**stationary kinetic transport equation**

Often the characteristic time constants for neutral particle transport phenomena are very short (μs), compared to those for plasma transport (ms). We can, therefore, often neglect explicit time dependence in the equations describing the neutral particles. This is done in most applications. The extension to time-dependent problems is rather straight forward and the procedure in the EIRENE code for such cases is described in section 1.7 below. This is because in a Monte Carlo transport procedure time can be treated in a completely symmetric fashion with the spatial coordinates.
For stationary (time-independent) problems the (scalar) transport flux ("angular flux") $\Phi$, where

$$\Phi(x) := |\mathbf{v}| \cdot f(\mathbf{r}, \mathbf{v}, i), \tag{1.5a}$$

with, again, $|\mathbf{v}|\Omega = \mathbf{v}$, is sometimes used as dependent variable, in preference to the distribution $f(x)$. For example the *total particle flux* is obtained from this "angular flux" by integrating it over all velocity space angles $\Omega$, i.e. as the zeroth $\Omega$ - moment of $\Phi(\mathbf{r}, \mathbf{v}, i)$.
In particular for stationary ($\partial/\partial t = 0$) and force free ($\mathbf{F} = 0$) problems, as e.g. often encountered in linear transport theory such as neutronics, radiation transport, neutral particle transport in plasmas, etc., the transport equation then reduces to the more compact form:

$$\nabla_{\mathbf{r}}\Phi(\mathbf{r}, \mathbf{v}, t) + \Sigma_t(\mathbf{r}, \mathbf{v})\Phi(\mathbf{r}, \mathbf{v}) = Q(\mathbf{r}, \mathbf{v}, t) + \int d^3v'\, C(\mathbf{r}; \mathbf{v}' \rightarrow \mathbf{v})\Phi(\mathbf{r}, \mathbf{v}') \tag{1.c}$$

Alternatively, in computational domains with non-vanishing collisionality (i.e., if $\Sigma_t(x) \neq 0$ everywhere) the (pre-) collision density $\Psi$ is used, i.e.,

$$\Psi(x) = \Sigma_t(x) \cdot \Phi(x) = \nu_t(x) \cdot f(x), \qquad \Sigma_t = \Sigma_t(\mathbf{r}, E, \mathbf{\Omega}, i) = \nu_t(\mathbf{r}, E, \mathbf{\Omega}, i)/|\mathbf{v}| \tag{1.5b}$$

where, again, the "macroscopic cross section" $\Sigma_t$ is the total inverse local mean free path (dimension: 1/length), and $\nu_t$ is the collision frequency (dimension: 1/time). This cross section can be written as a sum $\Sigma_t = \sum \Sigma_k$ over macroscopic cross sections for the different types (identified by the index $k$) of collision processes. Further details about this "macroscopic cross section" and its relation to the conventional ("microscopic") cross sections are given below, section 1.3.5.
In closing this section we note:
All these equations (1.a) to (1.d) given below are equivalent, of course, so are their corresponding **WCU!** generalizations discussed in the next section 1.1.1. Which particular form is used in a

particular discussion depends upon the issue which is considered. For example, the collision estimator for evaluating moments (responses) of the solution to the Boltzmann equation can be shown to be unbiased quite conveniently when using (1.d), whereas the track-length estimator, used for the same purpose in EIRENE, is most easily understood with the from (1.d). Time dependent cases are best discussed utilizing the form (1.b), etc.

### 1.1.1 The WCU! generalisation of the Boltzmann equation

The mathematical generalization from the classical Boltzmann equation (for a system undergoing only elastic collisions), equation (1.a) to the semi-classical Boltzmann equation (the **WCU!** equation, [**kn:WCU**]) for accommodating also chemical reactions (including, for example, vibrational relaxation or exchange of internal energy $\Delta E$ as special cases, but also chemical reactions $A+B \rightarrow C+D$, etc.) is obtained by a proper generalization of the species labels. The transitions are now symbolized as $\{i'l'\}, \{b'l'_1\} \rightleftarrows \{il\}, \{bl_1\}$. These generalized species indices label both the chemical species $(i, b)$ and/or the internal quantum state $(l, l_1)$ corresponding to modes of vibration, rotation or excitation. I.e. we regard two particles (objects) as different, with a different label, if either they belong to a different chemical species or they differ by their internal (electronic, or ro-vibrational, etc) state, as appropriate.
Now, by analogy to the discussion of the original Boltzmann equation (1.a), (1.a1), we consider then the phase-space balance equation for a given species/internal mode $\{i_0 l_0\}$. Then the sum in the collision integral is over all collision labels $\{i'l'\}, \{b'l'_1\}, \{bl_1\}$ and over all velocities of these three "species". Even more generally: in case there are more than two post-collision particles, i.e. more than two particles in the exit channel of the collision, integration (velocities) and summation (generalized species indices) in the **WCU!** collision integral is over all involved particles, except over $\{i_0 l_0; \mathbf{v}\}$, the parameters of the dependent variable $f_{i_0}(l_0, \mathbf{v})$. Employing the notations as in (1.a1) the full **WCU!** prototypical system of kinetic transport equations then reads (written here for two post-collision particles only)

$$
\begin{aligned}
D_t^{i_0 l_0}\, f_{i_0 l_0}(\mathbf{r}, \mathbf{v}, t) = & \sum_{i', l', b, l_1, b', l'_1} \int_{\Re^3, \Omega} d\mathbf{v}' d\Omega \;\; B_{i_0 l_0, b l_1}^{i' l', b' l'_1}(\mathbf{v}, \mathbf{v}) \times \\
& \times \left[ f_{i' l'}(\mathbf{v}') f_{b' l'_1}(\mathbf{v}') - f_{i_0 l_0}(\mathbf{v}) f_{b, l_1}(\mathbf{v}) \right] \\
& + Q_{i_0 l_0}(\mathbf{r}, \mathbf{v}, t)
\end{aligned}
\tag{1.6}
$$

with

$$
B_{il, bl_1}^{i' l', b' l'_1}(\mathbf{v}, \mathbf{v}) = \sigma_{il, bl_1}^{i' l', b' l'_1}(\mathbf{v}, \mathbf{v}; \mathbf{v}', \mathbf{v}') \; |\mathbf{v} - \mathbf{v}| \tag{1.7}
$$

The cross sections in the corresponding collision integrals $\sigma_{il, bl_1}^{i' l', b' l'_1}(\mathbf{v}', \mathbf{v}', \mathbf{v}, \mathbf{v})$ are multiple differential for scattering at a certain solid angle and post collision energies with simultaneous transition from $(il, bl_1)$ to $(i'l', b'l'_1)$. Again, as already in the Boltzmann equation restricted to elastic collisions, the pre- and post-collision velocities $\mathbf{v}', \mathbf{v}'$ and $\mathbf{v}, \mathbf{v}$, respectively, are not independent, but constrained by momentum and energy conservation, and the corresponding constrains are contained in the multi-differential cross sections, e.g. in form of delta functions.
For more than two post-collision objects (e.g. fission, dissociative excitation, stimulated radiation emission, etc.), this notation is readily generalized by adding more superscript labels and more post-collision velocities.

The extra discrete label $l$ introduced here compared to the Boltzmann equation may either be regarded as species index: $i_0 \rightarrow i_0 l_0$ or as other additional discrete independent variable (e.g. polarization, in case of radiation transport): $f_{il}(\mathbf{v}) \rightarrow f_i(l, \mathbf{v})$, whichever is more convenient.
As noted above, further generalizations to include particle splitting, absorption or fragmentation into more than two post collision products are straight forward, but can more conveniently be formulated in the $C$-collision kernel formulation introduced above and used below to relate the transport equation to a Markovian stochastic (discrete time) process.

**linear form of WCU! equation: the generic EIRENE equation**

As above, linearization will be again be obtained by grouping generalized labels $\{il\}$ and $\{bl_1\}$ into two disjunct classes, and allowing for reactions only between initially (prior to the collision) one member $\{i_0 l_0\}$ from class "$\{il\}$", the community of "test particles" and one member $\{b_0 l_{1_0}\}$ from the disjunct class "$\{bl_1\}$", the "bulk", "background" medium, or in our case, simply "plasma" species.

## 1.2 The linear integral equation for the collision density $\Psi$

By formally integrating the characteristics for (1.c) the same transport equation can also be written in integral form. This formal procedure is outlined below in section 1.2.1.
The resulting integral equation is often most conveniently written for the (ingoing-) collision density $\Psi$ (1.5b) rather than for transport flux $\Phi$:

$$\Psi(x) = S(x) + \int dx' \Psi(x') \cdot K(x' \rightarrow x). \tag{1.d}$$

This equation has the general form of the backward integral equation of a Markovian jump-process and it is therefore particularly well suited for a Monte Carlo method of solution. The formal relation between the integro-differential form (1.c) and this integral form is very useful to generalize the Monte Carlo procedure, e.g., to time-dependent equations, and to Boltzmann-Fokker-Planck equations (which contain diffusive contributions or diffusive approximations for some processes, in addition to the jump processes described by the Boltzmann collision integral). It also allows to make connection to the so called "Green's-functions Monte Carlo" concept (originally developed for quantum mechanical problems involving solutions to the Schrödinger equation). A corresponding discussion is postponed to section 1.2.1. A direct intuitive interpretation of the integral equation is already sufficient to understand the Monte Carlo method of solution and shall be given first.
In (1.d) $x'$ and $x$ are the states at two successive collisions (jumps). The integral $\int dx'$ in (1.d) is to be understood as an integral over all initial coordinates, i.e. over the entire physical space, the full velocity space and a summation over all species indices. The transition kernel $K$ is usually decomposed, in our context, into a collision- and a transport kernel, i.e., $C$ and $T$, where

$$K(\mathbf{r}', \mathbf{v}', i' \rightarrow \mathbf{r}, \mathbf{v}, i) = C(\mathbf{r}'; \mathbf{v}', i' \rightarrow \mathbf{v}, i) \cdot T(\mathbf{v}, i; \mathbf{r}' \rightarrow \mathbf{r}). \tag{1.8}$$

The kernel $C$ is (excluding normalization) the conditional distribution for new co-ordinates ($\mathbf{v}$,i) given that a particle of species $i'$ and with velocity $\mathbf{v}'$ has undergone a collision at position $\mathbf{r}'$.

This kernel can further be decomposed into:

$$C(\mathbf{r}', \mathbf{v}', i' \to \mathbf{v}, i) = \sum_k p_k \, C_k(\mathbf{r}'; \mathbf{v}', i' \to \mathbf{v}, i) \ , \ p_k = \frac{\Sigma_k}{\Sigma_t} \tag{1.9}$$

with summation over the index $k$ for the different types of collision processes under consideration and $p_k$ defined as the (conditional) probability for a collision to be of type $k$. The normalizing factor

$$c_k(x') = \sum_i \int d^3v \, C_k(\mathbf{r}', \mathbf{v}', i' \to \mathbf{v}, i) \ , \ C_k = \frac{1}{c_k} C_k \tag{1.10}$$

gives the mean number of secondaries for this collision process. The function $C_k$ then is a conditional probability density. The particle absorption process can conveniently be described by adding an "absorbing state" $x_a$ to the $\mu$-space (generally referred to as "one-point compactification of this space" in the language of mathematical topology). This "limbo state", once it is reached, is never left again if the kernels $T$ or $C$ are employed as transition probabilities. Formally, an additional collision kernel $C_a(x \to x_a)$ and an absorption probability $p_a = \Sigma_a/\Sigma_t$ must be included in the collision kernel. The quantity $\Sigma_a$ comprises all collision processes with no next generation particles within the community of particles considered by the coupled set of kinetic transport equations. (Ionisation of a neutral atom is a loss, if the resulting ion is not considered further, or dealt with by a **CFD!** code outside the Monte Carlo procedure).
The kernel $T$ describes the motion of the test particles between the collision events. Let, again, $\mathbf{\Omega}'$ denote the unit vector in the direction of particle flight $\mathbf{v}'/|\mathbf{v}'|$, and let $\mathbf{\Omega}'_2$ and $\mathbf{\Omega}'_3$ be two further unit vectors such that these three vectors form an orthonormal basis at the point $\mathbf{r}'$. Neither velocity nor species change along the transition described by $T$, i.e. $\mathbf{v}' = \mathbf{v}$ and $i' - i$. Omitting the corresponding delta functions in velocity space and the Kronecker delta $\delta_{i'i}$ the transport kernel $T$ then reads as follows:

$$\begin{aligned}(T(l) =) \ T(\mathbf{v}', i'; \mathbf{r}' \to \mathbf{r}) &= \Sigma_t(\mathbf{r}, \mathbf{v}', i') \cdot \exp\left[ - \int_0^{l=\mathbf{\Omega}'(\mathbf{r}-\mathbf{r}')} ds \Sigma_t(\mathbf{r}' + s\mathbf{\Omega}', \mathbf{v}', i') \right] \\ &\quad \cdot \delta(\mathbf{\Omega}'_2(\mathbf{r} - \mathbf{r}')) \cdot \delta(\mathbf{\Omega}'_3(\mathbf{r} - \mathbf{r}')) \cdot H(\mathbf{\Omega}'(\mathbf{r} - \mathbf{r}')) \qquad (1.11a) \\ &= \Sigma_t(\mathbf{r}, \mathbf{v}', i') \cdot F(\mathbf{v}', i'; \mathbf{r}' \to \mathbf{r}) \quad 0 \le l \le \infty \qquad (1.11b)\end{aligned}$$

with $H(x) = 0$ if $x < 0$, and $H(x) = 1$ if $x \ge 0$, the Heaviside step function (the unit step function). The two remaining delta functions restrict the motion to a path in the direction of the initial velocity $\mathbf{v}'$.
Thus, although strictly being a conditional (on $x'$) distribution in phase space, for an infinite medium $T$ can be interpreted as the distribution density for the distance $l$ for a free flight starting from $\mathbf{r}'$ to the next point of collision $\mathbf{r} = \mathbf{r}' + l \cdot \mathbf{\Omega}'$. We shall frequently omit the arguments $\mathbf{v}', i'$ to simplify notation, because neither initial velocity nor species change during a free flight.
For a finite medium this distribution can be generalized to (writing shorter $l$ for $(\mathbf{r}' + l\mathbf{\Omega}', \mathbf{v}', i')$:

$$T(l) = \begin{cases} \Sigma_t(l) & \cdot \exp\left[ - \int_0^l ds \Sigma_t(s) \right], \quad l < l_{max} \\ \delta(l - l_{max}) & \cdot \exp\left[ - \int_0^{l_{max}} ds \Sigma_t(s) \right], \quad l \ge l_{max} \end{cases} \tag{1.12}$$

Here $l_{max}$ denotes the distance along the flight direction from $\mathbf{r}'$ to the boundary for the computational domain, to any internal surface at which the test flight shall be stopped, e.g. for scoring

surface fluxes there, or even to the cell boundary. The "Transport Kernel" $T$ has dimension: [1/Dimension of phase space] ($T(x' \rightarrow x)dx$ is a probability). The function $F$ (Dimension: [length times Dimension of $T$]) defined in expression (1.11b), and analogously from (1.12) for a finite medium, will turn out to be the relevant Green's function for the transport problem, see section 1.2.1. The integral

$$\alpha(\mathbf{r}', \mathbf{r}) = \int_0^{\mathbf{\Omega}(\mathbf{r}-\mathbf{r}')} ds \Sigma_t(\mathbf{r}' + s\mathbf{\Omega})$$

in equation (1.11a) is well known as "optical thickness of the medium" in linear transport theory. The inhomogeneity $S$ in equation (1.d) is, excluding normalization, the distribution density of first collisions, whereas the integral term in equation (1.d) describes the contribution to $\Psi$ from all higher generations. The quantity $S$ can be written as:

$$S(x) = \int dx' Q(x') \cdot T(x' \rightarrow x), \tag{1.13a}$$

with a source density $Q$. As the problem is linear, $Q$ can be normalized to 1 and, thus, $Q$ can be considered a distribution density in $\mu$ -space for the "*primary*" birth points of particles, as, e.g., opposed to the "*secondary*" birth point distribution (or "post collision density") $\chi$ of particles after a collision event

$$\chi(x) = \int dx' \Psi(x') \cdot C(x' \rightarrow x). \tag{1.13b}$$

It can be shown that a unique solution $\Psi(x)$ exists subject to appropriate boundary conditions and under only mild restrictions (basically on the constants $c_k$ and $p_a$) to ensure that the particle generation process stays sub-critical.
Usually, a detailed knowledge of $\Phi$ or $\Psi$ is not required, but only a set of "responses", R, defined by

$$R = \quad < \Psi | g_c > \quad = \int dx \Psi(x) \cdot g_c(x) \quad \left( = \quad < \Phi | g_t > \quad = \int dx \Phi(x) \cdot g_t(x) \right), \tag{1.14}$$

where $g_c(x)$, $g_t(x)$ are given "detector functions".
For example all terms in the plasma fluid equations resulting from neutral plasma interaction can be written in this way. This can be seen by considering a numerical grid, composed of M mesh cells (spatial and/or temporal), for the numerical solution of the fluid equations. The detector functions for many responses needed for fusion plasma applications are hard wired in EIRENE, generalization to any arbitrary response by resorting to "user defined detector functions" is described in section 3.2.1 for track-length estimates, and in section 3.2.2 for collision estimates.
Lets therefore define an entire set of detector functions $g_m$, one for each mesh cell of an external code, each including a characteristic function

$$g_m = g \times ch_m(\mathbf{r}, t), \quad m = 1, 2, \ldots, M, \tag{1.15a}$$

i.e., $ch_m(\mathbf{r}, t) = 1$ inside the numerical mesh cell (or time interval) labeled with the cell index $m$, and $ch_m(\mathbf{r}, t) = 0$ outside this cell. Thus profiles of cell volume averaged responses are readily obtained in a single Monte Carlo run.

Estimates of surface fluxes, or point estimates e.g. in time, are also included in this concept if proper use is made of delta functions to reduce dimensionality of the response:

$$g_{m,\alpha} = g \times ch_m(\mathbf{r}, t) \times \delta^{\alpha}(\mathbf{r}, t), \quad m = 1, 2, \ldots, M, \tag{1.15b}$$

Here, e.g., a surface cell $m$ would discretize a surface $S$ characterized by a surface delta function $\delta^S$.

Equation (1.14) shows that Monte Carlo estimates (tallies) fall into the category of *extensive* quantities (number of particles, total energy, momentum, etc., per cell). *Intensive* quantities (flow velocity, temperature, etc.) obtained by dividing two extensive quantities are typically difficult to estimate with Monte Carlo methods (see: "ratio estimates", correlation between nominator and denominator, ...). If, however, the extensive quantity in the denominator is known exactly, such as e.g. the cell volume in a computational mesh, then, of course, obtaining an intensive quantity from an extensive quantity with Monte Carlo is a trivial re-scaling.

Monte Carlo estimates of the (intensive) volumetric source terms in the fluid equations due to the trace particles (neutral particles, but also trace ions described by the Monte Carlo procedure) are such cell averages, surface averages, time averages or point estimates (e.g. in time), averaged over a sub-manifold with reduced dimensionality. (This works, of course, only if the probability for particle histories crossing this sub-manifold is not zero). Surface averages in EIRENE are described in section 3.2.4, point averages in time are described in section 3.2.3. This option to reduce dimensionality of responses usually excludes point estimates in real space, for which special estimators not mentioned here would be required.

Nevertheless, in concluding, depending upon the numerical algorithm in the fluid code, one may then have to interpret these Monte Carlo estimates properly, e.g. they may have to be interpolated to the grid points, or be properly rescaled in case of time-dependent applications.

### 1.2.1 The Green's function concept

We return to Equation (1.11b), where the function $F(\mathbf{v}, i; \mathbf{r}' \rightarrow \mathbf{r})$) was defined. For a finite domain, see (1.12), and introducing again the shortcut $l$ for $(\mathbf{r}' + l\boldsymbol{\Omega}, \mathbf{v}, i)$ this becomes:

$$F(l) = \begin{cases} \exp\left[-\int_0^l ds \Sigma_t(s)\right], & l < l_{max} \\ 0 & l \geq l_{max} \end{cases} \tag{1.16}$$

This function $F$ is the Green's function of the left hand side (the "convective part") of the transport equation.

to be written

## 1.3 Monte Carlo solution of equation 1

A statistical solution to equation (1.b) is straight forward, because it is formulated in probabilistic terms as follows.
A discrete Markov chain is defined using $Q$ as an initial distribution and $L = T \cdot C$ (order of $C$ and $T$ reversed compared to $K$, $K = C \cdot T$) as a transition probability. Histories $\omega^n$ from this stochastic process are generated according to $\omega^n = (x_0, x_1, x_2, \ldots, x_n)$, (where $x_j = x_a$ for all $j \geq n$ and $x_i \neq x_a$ for all $i < n$), with $x_n$ being the first state after transition into the absorbing state $x_a$. $x_0$ denotes the initial state distributed as described by $Q$. Thus, the length n of the chain $\omega^n$ is itself a random variable. A random sampling procedure to generate such chains is carried out in Monte Carlo codes by converting machine generated (pseudo-) random numbers $\xi_{i_1}, \xi_{i_2}, \ldots$ into random numbers with the distributions $Q$, $T$ and $C$. Having computed N chains $\omega_i$, $i = 1, 2, \ldots, N$, the responses $R$ (1.14) with respect to detector function $g$ are estimated as the arithmetic mean of functions ("statistics", or "estimators") $X_g(\omega)$, i.e.

$$R \approx \tilde{R} = \frac{1}{N} \sum_{i=1}^{N} X_g(\omega_i). \tag{1.17}$$

The proper choice of the number of histories $N$ depends on the variance $\sigma^2(X_g)$ of the estimator $X_g$ and is highly problem specific. In EIRENE it can range from $N = 2$ for conditional expectation estimators and point sources in phase space, up to values of several millions for $N$.

### 1.3.1 Unbiased estimators

One possible choice for $X(\omega)$ is the so called "collision estimator" $X^c$, which, for a chain no. $i$, $\omega_i^n$ of length $n$ reads:

$$X_g^c(\omega_i^n) = \sum_{l=1}^{n} g_c(x_l) \cdot \prod_{j=1}^{l-1} \frac{c(x_j)}{(1 - p_a(x_j))} \quad , \tag{1.18}$$

and the summation index $l$ runs from the first point of collision $x_1$ until the last (absorption) at $x_n$. The product in the sum accounts cumulatively for branching, and absorption rates along the Markov chain $\omega_i^n$.
This estimator is, for example, used in the DEGAS code [**kn:Heifetz**]. It can be shown [**kn:Spanier69a**] in a tedious but mathematically strict way that the statistical expectation E($X^c$) produces:

$$R = E(X^c) = \int d(\omega) X_g^c(\omega) h(\omega) \tag{1.19}$$

with $h(\omega)$ being the probability density for finding a chain $\omega$ from the Markov process defined above, i.e. $X^c$ is an unbiased estimator for response $R$.
Other estimators ("track-length type estimators") are employed frequently. These estimators are unbiased as well but have higher moments different from those of $X^c$. Instead of evaluating the detector function $g_c(x)$ at the points of collisions, $x_l$, (as $X^c$ does), they involve line integrals of $g_t(x)$ along the trajectories, e.g.,

$$X_g^t(\omega_i^n) = \sum_{l=0}^{n-1} \left\{ \int_{x_l}^{x_{l+1}} ds \; g_t(s) \right\} \cdot \prod_{j=1}^{l-1} \frac{c(x_j)}{(1 - p_a(x_j))}, \tag{1.20}$$

with $R = E(X^t) = E(X^c)$. In literature mostly the detector function $g_t$ is taken to be constant within a grid cell. The line integral in (1.20) then collapses to the product $g_t s_l$, with $s_l$ being the distance ("track-length") travelled in cell during step $l$. The mild generalisation of the track-length estimators to detector functions, which can vary along the track even within a cell, is derived in [**kn:Reiter84a**] and applied in EIRENE as early as 1984 in [**kn:Reiter84b**].
It has been shown, e.g., in [**kn:Spanier69a**], that the collision estimator, derived not for the pre- but for the post-collision density integral equation results in a track-length type "conditional expectation estimator" $X^e$, which, together with $X^c$ and the "traditional" track-length estimator $X^t$, may be used as one further option in the EIRENE code.
This estimator $X^e$ is obtained from $X^t$ by extending the line integration, which is restricted to the path from $x_l$ to $x_{l+1}$ in formula (1.20), to the line segment from $x_l$ to $x_{end}$. Here $x_{end}$ is the nearest point on a boundary along the test flight originating in $x_l$. I.e., the line integration may be extended into a region beyond the next point of collision, into which the generated history would not necessarily reach. This "conditional expectation estimator" reads:

$$X_g^e(\omega_i^n) = \sum_{l=0}^{n-1} \left\{ \int_{x_l}^{x_{end}} ds\; g_t(s) \cdot \exp\left( - \int_0^s ds' \Sigma_t(s') \right) \right\} \cdot \prod_{j=1}^{l-1} \frac{c(x_j)}{(1 - p_a(x_j))}, \tag{1.21}$$

If $x_{end}$ is taken to be the nearest point on each mesh cell boundary, then the estimator (1.21) reduces, as a special case, to the method employed by the NIMBUS code [**kn:Cupini**]. However, the estimator (1.21) is more general, as the integration may be extended over arbitrarily many cells. The length of the integration path is controlled in EIRENE by an input flag WMINC, see section 2.10.

### 1.3.2 Scaling of tallies

With increasing $N$, the number of Monte Carlo histories, the unbiased estimators $X$ given in previous section 1.3.1 provide arbitrarily precise approximations for the responses $R_g = \langle \Psi | g \rangle$, for detector function $g$ and dependent variable $\Psi$ (1.d). The precise meaning and physical dimensions of $g$ and $\Psi$ depend upon the problem at hand, e.g. also on symmetry (ignorable spatial coordinates, or time).
Because the inhomogeneous part of the transport equation (1.d) $\mathcal{S}$ was assumed to be normalized to one (for source sampling),

$$\mathcal{S}(\mathbf{r}, \mathbf{v}, t) = \frac{1}{s} S(\mathbf{r}, \mathbf{v}, t), \quad s = \int d^3r d^3v dt\; S(\mathbf{r}, \mathbf{v}, t) \tag{1.22}$$

this normalization factor $s$ has to be multiplied to estimators (1.17) to turn responses $R_g$ (1.14) into estimates with correct dimensionality and in absolute units.
Mostly the responses of interest are *intensive quantities*, such as density, pressure, collision density (number of collision per unit time and volume) rather than the *extensive quantities* obtained by the Monte Carlo phase space integration in (1.14), total energy, momentum or number of particles. The volume, e.g. of a grid cell, is also an extensive quantity. Thus by dividing the estimate by the appropriate cell-volume $V_m$ (in space-time), see (1.15), finally the profiles of cell averaged intensive quantities (e.g. density profiles) are obtained. The final unbiased estimate, in absolute

units, for any of the unbiased estimators $X_g$ for detector function $g$ discussed above, then is:

$$R_g = \langle \Psi | g \rangle \approx \tilde{R}_g(N) = \frac{s}{V_m \times N} \sum_{i=1}^{N} X_g(\omega_i), \quad N \gg 1 \tag{1.23}$$

with $N$ being the number of Monte Carlo histories. Correct interpretation of the results of an EIRENE run hence requires knowledge of the ratio $s/V_m$. The total source strength scaling factor $s$, i.e. the integral of the inhomogeneous part $S$ of the governing Fredholm integral equation, is input to EIRENE, variable "FLUX" in input block 7 (see section 2.7). The space-time cell volumes $V_m$ are either automatically calculated by EIRENE in subroutine "VOLUME", for cells belonging to standard grid (input block 2, section 2.2), or are provided from external considerations by a number of options, e.g. input blocks 3b, 5, or 8. In this latter case great care is needed, however, that the cell volumes are proportional to the volumes *as seen by the test flights* for otherwise not only unscaled profiles, but even wrong (biased) profile shapes will be obtained.

#### 1.3.2.1 Scaling in problems with ignorable coordinates

As seen in the previous paragraph, the absolute values of estimates are scaled with the ratio $s/V$ of source strength to cell volume. Depending on ignorable coordinates in any particular problem, or on whether stationary (i.e.: time is an ignorable coordinate) or time dependent transport problems are considered, the dependent variable $\Psi$ in the governing integral equation (1.d), the source strength $s$ and "cell volume" $V$ may have different interpretations:

**stationary (time independent) problems**
**one ignorable spatial coordinate**

In stationary problems with one ignorable coordinate, say, the z-coordinate, the source strength $s$ (input flag "FLUX") is the flux, particles per unit length $dz$ in direction of $z$. Likewise, the volume $V$ is per unit length in the ignorable direction, i.e. if $dz = 1$ then $V$ is the cell area in the two remaining coordinates $x, y$. The unit length $dz$ of an ignorable coordinate (here: $z$-coordinate) used in a particular run is determined by the input flags in the corresponding input block for standard grid options, i.e. in input block 2A for the x-coordinate, input block 2B for the y-coordinate and input block 2C for the z-coordinate, see section 2.2. Note that the numerical value of source strength "FLUX" (block 7) corresponds to the choice of e.g. $dz$. If $dz = 1$ cm, then "FLUX" is the number of particles per unit time and per cm in z-direction. If $dz = 1$ m, then the same value of variable "FLUX" would correspond to the 100 times smaller source strength of "FLUX" particles per unit time and per meter. All resulting volume averaged output tallies would have a value 100 times smaller. A particle density might then also be interpreted as surface density (particle per unit area).

### 1.3.3 Statistical errors, Efficiency (FOM!)

The efficiency for Monte Carlo Codes is the inverse of the **FOM!** (**FOM!**) of the calculation, defined as

$$\text{FOM} = \textit{statistical variance} \cdot \textit{computing cost}. \tag{1.24}$$

Note that FOM should be approximately independent of the running time, because the number of histories generated is (excluding overhead) proportional to the CPU costs and inversely proportional to the statistical variance $\sigma^2$.

It is one of the major advantages of Monte Carlo methods over other numerical schemes that the error estimates, empirical variances $\tilde{\sigma}^2$, are directly provided by the method itself, not requiring any further considerations. The options to activate evaluation of statistical variance in an EIRENE run for any computed quantity (tally) are described in input block 9 (section 2.9). EIRENE provides numerical and graphical output for the "empirical relative standard deviation", in %.

$$\tilde{\sigma}_{g,rel}(N) = \tilde{\sigma}_g(N)/\tilde{R}_g(N) \times 100 \tag{1.25}$$

and $\tilde{R}_g^N$ is the Monte Carlo estimate for tally $R_g$ based on $N$ Monte Carlo histories (1.23), properly scaled to source strength $s$ (1.22). Here $R_g$ may be any volume or surface averaged tally for detector function $g$, either track-length, collision, or snapshot estimated. The variance $\sigma_1^2$ per history is obtained as the (unbiased) estimate ("empirical variance") as:

$$\sigma_1^2 \approx \tilde{\sigma}_1^2(N) = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \overline{X})^2 = \frac{1}{N-1}\left[\sum_{i=1}^{N} X_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} X_i\right)^2\right] \tag{1.26}$$

where $X_i = X_g(\omega_i)$ is the contribution of Monte Carlo history $\omega_i$ to estimator $X_g$ for tally $R_g$, and $\overline{X}$ is the arithmetic mean $\overline{X} = 1/N \sum X_i$ over all histories. $N$ is the number of (statistically independent) Monte Carlo histories. The subscript $g$ is omitted here and from now on. Note that the expression on the right hand side of (1.26) can be evaluated after each completed particle history, i.e. without storing all individual contributions $X_i$ first.
The variance per history is turned into the final estimate for the Monte Carlo variance by the "law of large numbers" (and the "central limit theorem" of probability theory):

$$\tilde{\sigma}_{MC}^2(N) = \frac{1}{N}\tilde{\sigma}_1^2(N) \tag{1.27}$$

For large $N$ the variance per history (= variance of a single observation) $\tilde{\sigma}_1^2$ converges to a constant (namely to $\sigma_1^2$), and the final Monte Carlo estimate of variance $\tilde{\sigma}_{MC}^2$ (1.27) has the expected probabilistic $1/N$ scaling, i.e. the empirical standard deviation, which is the Monte Carlo error estimate, scales as: $\tilde{\sigma}_{MC} \sim 1/\sqrt{N}$.

### Multiple cell crossings

One should note that except in simple 1D cases in general one single Monte Carlo history $\omega_i$ can contribute more than once to the estimate $\tilde{R}_g$ for a particular cell of the grid or surface segment. E.g. test particles can cross one cell $m$ [see (1.15b)] more than once, with each cell crossing "$j$" leaving a score (contribution) $X_{i,j}$ to estimator $X_g(\omega_i)$. Then:

$$X_i = \sum_{j=1}^{J_i} X_{i,j} \quad , \tag{1.28}$$

with $J_i$ being the number of contributions (e.g. cell crossings) to the estimator, from particle history no. $i$. The summation in (1.17) and hence also the second sum on the right hand side of (1.26) can still be accumulated "on the fly" while generating the Monte Carlo histories (i.e. at no extra CPU cost). However, because, of course $X_i^2 \neq \sum X_{i,j}^2$, the first sum on the right hand side of (1.26) cannot be evaluated "on the fly" (per event). Instead here EIRENE first accumulates the sum (1.28) $X_i$ for individual test flight "$i$", and updates the sum $\sum X_i^2$ only once after each

completed history (in subroutine STATIS, entry STATS1). I.e., variance estimates are made after completion of each history, inside the particle loop, see fig. 1.11. The full evaluation and scaling of expression (1.26) and (1.27) is then carried out after completion of all histories (per stratum), in entry STATS2 of subroutine STATIS.

**Linear functions of tallies**

Often algebraic functions of tallies (sums, products, ratios, . . . ) are formed after a Monte Carlo run to produce further (post processed) output quantities (see input blocks 10C and 10E in section 2.10). For those tallies standard deviations are not available in general, due to statistical correlation between individual tallies obtained from the same Monte Carlo run.
However, for linear functions of tallies (e.g. sums, differences), this is possible even without resorting to the covariance estimators (block 9, section 2.9) of EIRENE. Usually this was done in "problem specific" modules, e.g. in STATS1_COP for source terms arising in plasma fluid codes (see section 2.14). In 2012 a new routine (UPFCOP) was added, and is called inside the particle loop, scored per particle (not per event) and prior to STATS1.
Let a linear function $R_{lin}$ of default tallies $R_k$ be given by

$$R_{lin} = \sum_k a_k R_k, \tag{1.29}$$

($a_k$ are some constant scalar factors) then after each completed history $\omega_i$ in routine UPFCOP the sum of contributions

$$X_l(\omega_i) = \sum_k a_k X_k(\omega_i) = \sum_k a_k \sum_j X_{k,j}(\omega_i) = \sum_j \sum_k a_k X_{k,j}(\omega_i) \tag{1.30}$$

is formed, to finally build the linear function tally $R_l(N) = 1/N \sum_i X_l(\omega_i)$ by averaging over all histories, same as for default tallies. This is possible because in case of linear combinations of tallies the summation over tallies $k$ and event-contributions $j$ of a single history can be interchanged (1.30).
Finally the variances for $R_l$ are evaluated in STATS1, reusing the procedure used for default tallies, and the former problem specific routine STATS1_COP for variances from tallies needed for code coupling has hence been made redundant, as indicated in red color in fig. 1.11. Currently linear combination tallies $R_l$ are stored on array (tally) COPV. For them all printout and graphical output options are available. One example of such linear combination tallies is discussed in section 2.14 for interfacing EIRENE with **CFD!** codes, as usually the (kinetic) reaction source terms provided by EIRENE to plasma fluid codes are such linear functions of default tallies.

**Statistical independence of MC histories**

When the Monte Carlo histories from one EIRENE run are not strictly statistically independent, as e.g. in case of stratified source sampling (see paragraph below), then a modified formula for the statistical error estimates has to be used.

#### 1.3.3.1 Sampling, Non-analog sampling

One often encounters in literature the description of special, "new" Monte Carlo techniques, that are greatly superior to so called "standard methods".

It is true that one can devise very intelligent methods to optimize performance, but each optimization always only works well for one very particular problem, it can equally well entirely wreck the performance of an only slightly different case.
For a general purpose Monte Carlo solver for transfer problems, as EIRENE, therefore, no general advise can be given, although the performance often could greatly be improved by adapting the method to a particular problem. In particular, non of these "intelligent methods" have been (and will ever be) hard wired into EIRENE.
EIRENE contains a set of such methods, referred to as "non-analog" methods and controlled by the flags in input block 9, see section 2.9. Activating those must be accompanied by a very careful statistical analysis of results, not just the run time per particle, nor even the standard deviation alone, suffices.
We first note here that any random distribution function $f(x)$ arising during the course of generating the chains $\omega$ ("particle histories") can be replaced by another "non-analog" function, $g(x)$, if a weighting factor,

$$w(y) = f(y)/g(y) \tag{1.31}$$

is included in the formula for the estimator $X(\omega)$, y being an actual random number generated from $g$. This choice can in some cases increase the efficiency of the algorithm.
The only restrictions on the choice of non-analog sampling functions $g(x)$ (besides practical ones) are:

$$\text{if} \quad g(x) = 0, \qquad \text{then} \quad f(x) = 0; \tag{1.32}$$

and conditions to ensure that the non-analog process remains sub-critical as well.
The condition (1.32) is checked in an EIRENE run whenever non-analog distributions are applied, and, if violated, an error exit with the message: "violation of Radon–Nikodym condition" occurs. Note: a violation of condition (1.32) can not be detected otherwise, e.g., by monitoring overflow in the weighting factor $w$. Of course, values y resulting in a zero in the denominator of $w$ (1.31) have sampling probability zero according to distribution $g(x)$, and, hence, are never sampled. Still the results would be biased.
Note: when inspecting an EIRENE geometry plot with particle trajectories (or even a movie: NLMOVIE=TRUE, input block 1) in order to get an intuitive feeling for the particular transfer process considered, then all non-analog sampling options must first be turned off, (NLANA flag in input block 1), for otherwise the pictures may be grossly misleading.

We consider the procedures for random sampling from univariate distributions as known, and refer to the many textbooks on that, in particular to the "random sampling library" [**kn:Sampling**].
If none of the direct methods apply, then still either the non-analog method mentioned above, or the "rejection method" can be used.
Sampling from a multivariate distribution $f(x_1, x_2, \ldots, x_n)$ can always be reduced to a sequence of samplings from univariate distributions, by noting that:

$$f(x_1, x_2, \ldots, x_n) = f_1(x_1) \cdot f_2(x_2|x_1) \cdot f_3(x_3|x_1, x_2) \cdot \ldots \tag{1.33}$$

Here, $f_1$ is the marginal distribution obtained from $f$ by integrating over all but the first independent variables. It is a univariate distribution of $x_1$.
$f_2$ is a conditional marginal distribution obtained from $f$ firstly by integrating $f$ over all but the first 2 variables $x_1, x_2$, and secondly then taking the conditional distribution, conditional on $x_1$ (i.e., the univariate distribution of $x_2$ for given values of $x_1$).

Likewise, $f_3$ is a conditional marginal distribution, n-3 fold integration of $f$, and then distribution conditional on $x_1, x_2$, and so on.
Random sampling from $f(x_1, x_2, \ldots, x_3)$ then proceeds by sampling first $x_1$ from the (univariate) first factor in (1.33), then $x_2$ by sampling from the (univariate) second factor, and so on.
This transformation of a multivariate (k-) distribution to a uniform distribution on a k-hypercube is sometimes referred to as "Rosenblatt transformation" (M. Rosenblatt, Ann. Math. Statistics 23: 470-472, (1952)).
One particular example of this scheme is the "TRIM database surface reflection model" (see below, section 1.4.2). There, essentially, tables of the conditional marginal distributions mentioned here are pre-computed with the TRIM code. For convenient random sampling, these tables are stored for the inverted cumulative distribution, i.e., as conditional quantile functions.

#### 1.3.3.2 Stratified Source Sampling

The EIRENE code resorts to a "stratified sampling" technique. This technique is one the few Monte Carlo variance reducing techniques which are quite straight forward, easy to implement and, most importantly, behave in a well predicable way. It is therefore recommended to always use "source stratification" described below as much as possible. There are, however, non-trivial issues of load balancing in case of multi-core runs (i.e. Monte Carlo code parallelization), which are related to assignments of strata to cores, in particular if the average CPU-time per history strongly varies between strata. In such cases the "proportional allocation of weight" to source strata discussed below is technically difficult to achieve together with proper load balancing, due to the stochastic termination of the random walks in the various strata. Whereas without stratification extracting parallelism seems as trivial as constructing many independent samples in parallel, this "embarrassingly parallel" feature of linear Monte Carlo transport codes is lost, and parallelization in combination with stratified source sampling requires some attention.
Stratified sampling means that the primary source distribution $Q(\mathbf{r}, \mathbf{v}, t)$, see Equation (1.a), or more generally, for $S(x)$ in (1.d), can be decomposed into a sum of $M$ independent sources (this applies for both: the source distribution $S$ itself and the corresponding source strength $s$):

$$S = \sum_{k}^{M} S_k; \quad s = \sum_{k}^{M} s_k \tag{1.34}$$

and the solution is obtained by linear superposition of the solutions for each sub-source ("stratum") $S_k$. We use again the slightly more general notation $S$ for the inhomogeneous part of the Boltzmann equation rather than $Q$, which is the physical source of particles (objects), i.e. a special case of $S$.
The stratified sampling technique is a well known statistical procedure from experimental planning, see any textbook on statistics or Monte-Carlo methods.
It sometimes can significantly affect the efficiency of a Monte-Carlo run. This can go both ways, depending upon the particular problem and the particular stratification used. Therefore, as with the non-analog options, no general recommendation can be made.
The effects of source stratification, however, are usually more easy to assess (predict) than those resulting from general non-analog methods.
In order to make connection with the textbooks, in which "stratified sampling" is often discussed in connection with Monte Carlo integration procedures, the Monte Carlo solution of the transport

problem is interpreted as a Monte Carlo integration, and the "stratified sampling" in Monte Carlo integration becomes a "stratified source sampling" in transport problems in this interpretation. To see this we need to note here only the following (based on Fredholms alternative for the linear Boltzmann operator):
EIRENE provides estimates of linear functionals ("moments", "responses")

$$\langle g, \phi_S \rangle = \langle S, \phi_g^* \rangle \tag{1.35}$$

As described above, $g$ is the detector function ("weighting function") $\phi_S$ is the solution to the kinetic transfer equation (Boltzmann equation) in which $S$ was used as (inhomogeneous) source term. $\phi_g^*$ is the solution of the corresponding adjoint equation, with $g$ as source term (and $S$ as "weighting function".
The above equation means that EIRENE estimates can simply be regarded as Monte-Carlo estimates of multi-dimensional integrals, namely of the adjoint function $\phi_g^*$ integrated over phase space with $S$ defining a distribution (measure) in phase space "$\mu_S$".
It is then obvious that one can split that integral into a sum of integrals by arbitrarily decomposing the domain of integration, i.e., by decomposing the source $S$ into sub-sources ("strata") $S_k$ with

$$R_g = \int d\mathbf{x}\, \phi_S \cdot g = \langle S, \phi_g^* \rangle = \sum_k^M \langle S_k, \phi_g^* \rangle = \sum_k^M \int d\mu_{S_k} \phi_g^* \tag{1.36}$$

This is the concept of "stratified sampling", which, for Monte Carlo particle transfer procedures in particular actually turns out to be a "stratified source sampling".
Further details about this in our particular context are given in the description of input block 7, section 2.7 below. As pointed out above the evaluation of statistical error estimates has to account for the stratification, because with stratification the Monte Carlo histories are not strictly independent anymore.
Grouping the $N$ statistically independent test flights (random samples) into $M$ strata, leads to Monte Carlo particle numbers $N_k$ sampled from stratum $k$ (i.e., with birth points sampled only from source $S_k$) with:

$$\sum_{k=1}^{M} N_k = N \quad , \quad N_k > 0 \quad \text{for} \quad k = 1, \ldots, M \tag{1.37}$$

where, again, we omit the subscript $g$ for the particular tally (response function) from now on.
The choice of the $N_k$ is only restricted by the normalization condition in (1.37). This therefore also applies for the allocation of CPU time to individual strata for a given overall CPU time. This additional freedom can be used in EIRENE runs to affect the statistical error for a given CPU time. Unfortunately, as with non-analog sampling, this can go both ways. Fortunately, however, here a clear general recommendation can be given, see below in this section: "proportional allocation".
Lets denote by $\tilde{R}_k(N_k)$ the estimate for tally $R$ as obtained only from the $N_k$ samples from stratum $k$ (but scaled with the full source strength $s$). Likewise, we denote by $\tilde{\sigma}_{1k}^2(N_k)$ the variance of estimate $\tilde{R}_k$ per history, i.e. for a single sample, obtained by only using the $N_k$ samples from stratum $k$ (again implying the scaling with total source strength $s$). Then clearly

$$\tilde{R}^*(N) = \sum_k \frac{s_k}{s} \tilde{R}_k(N_k) \tag{1.38}$$

is the (only) unbiased weighted estimate of $R$ obtained from summing over all strata, i.e. from all $N$ samples. The asterisk denotes the estimate $\tilde{R}$ of $R$ obtained with stratification, $N = N_1 + \cdots + N_M$, and $\tilde{R}(N)$ is the estimate of the same quantity from a simple sample (no stratification) of same size $N$:

$$R = E\{\tilde{R}^*(N)\} = E\{\tilde{R}(N)\} \tag{1.39}$$

where $E\{X\}$ denotes the "expectation value" of random variable $X$. Furthermore:

$$\tilde{\sigma}_k^2(N_k) = \frac{1}{N_k}\tilde{\sigma}_{1k}^2(N_k); \quad \tilde{\sigma}^{2^*}(N) = \sum_k \frac{s_k^2}{s^2}\frac{1}{N_k}\tilde{\sigma}_{1k}^2(N_k) \tag{1.40}$$

where again the asterisk indicates the value obtained with stratification. The right equation follows from the sum rule for variances and because the estimates for the strata are mutually statistically independent. It is clear that the variance with stratification $\tilde{\sigma}^{2^*}(N)$ can certainly be different from the variance $\tilde{\sigma}^2(N)$ obtained from a simple random sample (without stratification) of the same size $N$. To see this we write the trivial identity:

$$\tilde{\sigma}^2(N) = \sum_{k=1}^{M} \frac{s_k}{s}\tilde{\sigma}^2(N) \tag{1.41}$$

For example, in case of "proportional allocation" of CPU-time to strata according to their source strength $s_k$, i.e., $N_k \propto s_k$ :

$$N_k = \frac{s_k}{s}N, \quad k = 1, 2, \ldots, M \tag{1.42}$$

one finds from (1.40) for the variance of the stratified sample:

$$\tilde{\sigma}^{2^*}(N) = \sum \frac{s_k}{s}\tilde{\sigma}_k^2(N_k) \tag{1.43}$$

which can be substantially smaller than the variance from the simple sample (1.41) of size $N$, if the subdivision into strata is made such that the variability within strata $\sigma_k^2$ is less then the variability in the entire population $\sigma^2$. Furthermore, one can show that with proportional allocation (1.42) the variance from the stratified sample is always less than or equal to the variance from the simple sample of same size $N$, no matter how the subdivision into strata was done:

$$\text{If} \quad N_k = \frac{s_k}{s}N; \quad k = 1, 2, \ldots, M \quad \text{then} \quad \tilde{\sigma}^{2^*}(N) \le \tilde{\sigma}^2(N) \tag{1.44}$$

i.e., stratified source sampling with proportional allocation is an "inherently safe working procedure" for variance reduction.
This important inequality can be derived after some lengthy algebra, leading finally to

$$\tilde{\sigma}^2(N) = \tilde{\sigma}^{2^*}(N) + \frac{1}{N}\sum_k [R_k(N_k) - R(N)]^2 \tag{1.45}$$

Clearly, the second term on the right side of this equation, which is the variance between strata, is always positive (or zero) and this fraction of the variance is eliminated from the Monte Carlo estimate by stratified sampling.

This fact is the reason for the EIRENE option of "proportional allocation of CPU time" described in input block 7, section 2.7.
In principle one can even optimize the CPU performance (i.e. minimize the variance per CPU time, for any given stratification $S_k$) by choosing the allocation $N_k$ to strata in an even more intelligent way (involving a-priori knowledge of the variability within strata $\sigma_k^2$, e.g. estimated from a previous iteration.) But these options are likely to be quite sensitive to the choice of the particular tally (response $R_g$). In a typical EIRENE application many such responses for a large variety of detector functions $g$ are estimated simultaneously in one single run. Therefore this "optimal allocation" schemes are not currently implemented.

#### 1.3.3.3 Testing the stratification

The proper implementation of the stratification, for all kinds of output tallies (such as volume averaged tallies, surface averaged tallies, spectral tallies) can be verified by including two: $M = 2$ (or more: $M$) identical strata, even with identical number of test-histories $N_k$ and identical random number seeds for those. The standard deviations of the sum over these identical strata must then be smaller by a factor $\sqrt{2}$ (or: a factor $\sqrt{M}$, respectively) *exactly*, rather than only approximately, as it would be the case without stratification by simply enhancing the number of test-histories for a stratum by a factor two (a factor $M$, respectively).

### 1.3.4 Source sampling

Source sampling in EIRENE is carried out in routine LOCATE.F. This routine converts uniform random numbers into random numbers with distribution $S_k$, the k-stratum part of the inhomogeneous term $S$. Again sequential sampling (1.33) from conditional distributions is applied.
In LOCATE.F the following sequence is used (stratum no. $istra = k$ is set in calling program):

$$S = S_1(\mathbf{x}) \times S_2(i|\mathbf{x}) \times S_3(\mathbf{v}|\mathbf{x}, i) \tag{1.46}$$

with $S_1(\mathbf{x}) = \sum_i \int d^3v S(\mathbf{x}, i, \mathbf{v})$ being the cumulative distribution of the spatial coordinate $\mathbf{x}$ of the "birth point" of the trajectory, obtained by integrating the source term $S$ over all velocities and summing over all species. $S_2(i)$ is the distribution of birth species index, obtained by integrating over all velocities and conditional on the already sampled spatial birth point. $S_3(\mathbf{v})$ finally is the velocity distribution at the birth-point of the trajectory, conditional on the already sampled species index $i$ and birth point location $\mathbf{x}$.
This procedure is employed for all types of sources, namely for re-sampling from a stored "census array" (e.g. initial condition in time dependent mode), for point sources, line sources, surface sources and volume sources.
If the sampled species $i$ belongs to the background species ($i = i_b$) rather than to the test particle species community, then, after completion of sampling from $S$ an appropriate "collision" is carried out (e.g. with a wall surface in case of surface sources), until a test particle species $i_t$ results. I.e., formally a sampling from kernel $C(\mathbf{x}, \mathbf{v}, i_b \rightarrow i_t, \dots)$ is carried out at the place of birth $\mathbf{x}$ to produce initial test particle coordinates.
Stratification of sources has been described in section 1.3.3.2.

#### 1.3.4.1 time-census sources

#### 1.3.4.2 Point sources

#### 1.3.4.3 Line sources

#### 1.3.4.4 Surface-sources

and see special section on "plasma surface recycling source sampling", section 1.5.

#### 1.3.4.5 Volume sources

special case: volume (radiative or three-body) recombination of plasma ions and electrons to neutrals. Also used for: spontaneous photon emission source in radiative transfer simulations.

### 1.3.5 Sampling a free flight from the transport kernel T

The conditional sampling distribution $T(\mathbf{r}' \rightarrow \mathbf{r})$ for the next point of collision (event), given a test flight is at position $\mathbf{r}'$, traveling with velocity $\mathbf{v}'$, is given by (1.11a).
A more intuitive argument for this distribution of free flight path lengths follows from the simple attenuation law of a mono-energetic beam incident on a thin slab of material, thickness $dl$, target material density $n_t$. Clearly the fraction of beam particles undergoing collisions in this slab is $\sigma n_t dl$, with "cross section" $\sigma$ (stationary target). This results in the attenuation law

$$dn = -nn_t\sigma dl \qquad \rightarrow \qquad n = n_0 \exp(-n_t\sigma l) \tag{1.47}$$

where $n$ is the number of beam particles, $n_0$ the number of beam particles at $l = 0$. It is then naturally supposed that

$$p(l)dl = [\exp(-n_t\sigma l)]n_t\sigma(dl) \tag{1.48}$$

is the probability for a first collision between $l$ and $l + dl$, and hence

$$P(l) = \int_0^l [\exp(-n_t\sigma l)]n_t\sigma(dl) = 1 - \exp(-n_t\sigma l) \tag{1.49}$$

is the corresponding probability distribution function for a first collision at distance $\leq l$, i.e. the same results as derived above formally with the Green's function argument.
Returning to (1.11a) we first again omit velocity and species index in the argument lists to keep the notation simple, but we note that certainly the mean free path $\lambda$ (or macroscopic cross section $\Sigma_t$) usually depend explicitly on the test particle energy and species. Let us introduce the distance of the flight $l$ in direction $\mathbf{\Omega} = \mathbf{v}/|\mathbf{v}|$, hence $\mathbf{r} = \mathbf{r}' + l\mathbf{\Omega}$, then the sampling distribution for $l$ is simply

$$T(l) = \Sigma_t(\mathbf{r}' + l\mathbf{\Omega}) \exp\left[-\int_0^l ds\Sigma_t(\mathbf{r}' + s\mathbf{\Omega})\right], \quad \Sigma_t = 1/\lambda_t \tag{1.50}$$

Distances $l$ are readily sampled from this exponential distribution using the inversion method and the cumulative distribution $G_T$ of $T$

$$\xi = G_T(l) = 1 - F(l) = 1 - \exp\left[-\int_0^l ds\Sigma_t(\mathbf{r}' + s\mathbf{\Omega})\right] \tag{1.51}$$

with $\xi$ (and hence also $1-\xi$) being a uniformly distributed random variable on [0,1]. Therefore, accumulating $\left[-\int_0^l ds\Sigma_t(\mathbf{r}'+s\mathbf{\Omega})\right]$ along a flight and comparing it with $\ln(\xi)$ with a uniform random number $\xi$ provides (by inversion) a random sample of the free flight distance $l$ with the proper distribution.
Sampling the distance $l$ of a free flight, for test particle species "$i$" with velocity $\mathbf{v}$, from the transport kernel $T$ requires knowledge of the ("monochromatic") mean free path $\lambda = \lambda(\mathbf{r}, |\mathbf{v}|)$ along the trajectory, and, hence, of the "macroscopic cross section" $\Sigma = \Sigma(\mathbf{r}, |\mathbf{v}|)$, which also may vary along the trajectory.
The "macroscopic cross sections" $\Sigma$ (by abuse of language, dimension: 1/length) with or without subscript $k$ (the label for the type of the collision process) or subscript $t$ (for "total", the sum over $k$), are defined as:

$$\Sigma = \frac{1}{\lambda} = \frac{\nu}{|\mathbf{v}|}, \qquad \lambda = \frac{|\mathbf{v}|}{\nu} \tag{1.52}$$

with the mean free path $\lambda$ and the collision frequency $\nu$:

$$\nu(\mathbf{r}, \mathbf{v}, i) = n_b \langle \sigma \cdot v_{rel} \rangle_b, \qquad v_{rel} = |\mathbf{v} - \mathbf{v}_b| \tag{1.53}$$

Here $n_b$ and $\mathbf{v}_b$ are the target particle density and velocity, respectively, and the brackets $\langle \dots \rangle_b$ denote averaging with the velocity distribution $f_b(\mathbf{r}, \mathbf{v}_b)$ of the background medium species $b$.
$\sigma$ is the total scattering cross section, $\sigma = \sigma(|\mathbf{v}_{rel}|)$.
Hence, e.g., for all isotropic target distributions $f_b$ one has:

$$\nu = \nu(\mathbf{r}, E, \mathbf{\Omega}, i) \qquad \text{and} \qquad \Sigma = \Sigma(\mathbf{r}, E, \mathbf{\Omega}, i).$$

Furthermore, let $f_b$ be a drifting Maxwellian, with temperature $T_b$ and drift $\mathbf{v}_b$: $f_b = f_b(T_b, \mathbf{v}_b)$.
Then:
$\nu = \nu(T_b, |\mathbf{v} - \mathbf{v}_b|, i)$
Finally, if the thermal speed $v_b^{therm}$ in this Maxwellian: $v_b^{therm} = \sqrt{T_b/m_b}$ is very large compared to typical values of $|\mathbf{v} - \mathbf{v}_b|$ (i.e.: nearly incompressible flow conditions), then:
$\nu \approx \nu(T_b, 0, i) = n_b \langle \sigma \cdot v \rangle_b$
and here $\langle \sigma \cdot v \rangle_b$ is the usual "Maxwellian rate coefficient" for the collision process, taken at temperature $T_b$ of the background particles (species $b$) and at zero velocity of the test particles (species $i$). For example, in case of neutral atom or molecule collisions with electrons this approximation is usually valid. From the point of view of electrons or ions, colliding with a cold background of neutrals or ions ($T_b \approx 0$, we have the opposite case: $|\mathbf{v}| \gg v_b^{therm}$ and then the rate coefficient reduced to the simple product $\sigma(v)v$ with $v$ being the test particle velocity.

#### 1.3.5.1 Alternative: fixed time step (or constant path length increment)

If the particle orbit between two events (the Green's function) is not given explicitly, e.g., not as simple straight line, then the orbit has to be integrated numerically, using small time steps $\Delta t$ or path length increments $\Delta\mathbf{x}$. The algorithm to determine the next point of collision then is different:
After a free flight time $\Delta t$ the probability of at least one collision during this period is the function $G_T(l_0)$ defined above, with $|\Delta\mathbf{x}| = l_0 = \int_0^{\Delta t} \mathbf{v}(t)dt \approx v\Delta t$ the fixed path length for this step. This follows from the definition of $G_T$ as cumulative distribution for the free flight length (1.51).

Clearly $0 \leq G_T(l) \leq 1$ for all positive $l$, by definition. Lets now assume constant mean free path $\lambda$, and constant velocity $\mathbf{v}$ during time step $\Delta t$, and $l_0 = |\mathbf{v}|\Delta t$, to simplify notation.
The procedure to find the free flight length (or time) is now as follows:
Rather than sampling $l$ by inversion, as in Equation (1.51), now a uniformly distributed random number $\xi$ on [0,1] is compared with the probability $G_T(l_0) = 1 - \exp[-l_0/\lambda]$.
If $\xi > G_T(l_0)$, no collision has happened during this step, and the next step is carried out.
If $\xi \leq G_T(l_0)$, then at least one collision has happened prior to completion of the step.
Strictly, in this case the exact position of the first collision during this step has to be identified, by inversion, the time step has to be shortened and the particle has to be moved to the sampled point of collision in the interval $\Delta\mathbf{x}$. However, various approximations are now usually made, to simplify coding.

- Ignore the fact that more than one collisions could have happened during $\Delta t$.
- replace $G_T(l_0)$ by $l_0/\lambda$ $(= \Delta t v \Sigma_t(\mathbf{v})$, see above) when comparing with uniform random number $\xi$

The second assumption results from Taylor expansion for sufficiently small $\Delta t$ such that $l_0/\lambda \ll 1$.
The first assumption is more difficult to justify generally:
If the post-collision test particle species is the same and its new velocity is not too different from that prior to the collision, then the probability for $n$ collisions in the same time $\Delta t$ is roughly $P_n = G_T(l_0)^n$ and the number of missed collisions in $\Delta t$ is

$$\sum_{i=2}^{\infty} P_i = \frac{G_T(l_0)^2}{1 - G_T(l_0)} \tag{1.54}$$

For sufficiently small $\Delta t$ (and hence $l_0$) this error can be made small.
If, however, collisions involve change in species, and/or significant changes in velocity, then the smallness of $l_0/\lambda$ must be guaranteed not only for the particle which is currently being considered, but for all particle species in the system and for all possible energies (of secondaries). E.g.: the excitation cross section for process $A \rightarrow A^*$ may be small, hence $l_0/\lambda \ll 1$ for species $A$, but the de-excitation (quenching) of state $A^*$ may be very rapid. Then the quenching time of species $A^*$ determines the permitted time-step for species $A$ in this approximate procedure. Other critical examples may be systems composed of both rather stable chemical species and short living radicals, e.g hydrocarbons.
The EIRENE code therefore only uses the correct full sampling procedure (1.51), even when particle orbits (e.g. of ions) are integrated numerically with time-steps.

### 1.3.6 Sampling from the collision kernel C

The general rules for sampling from multivariate distributions outlined in section 1.3.3, Equation (1.33) also apply here. In the EIRENE code sampling from the collision kernel is carried out in subr. COLLIDE, which is called when a collision is known to have occurred at a pre-collision particle state vector [position, velocity, species (and, optional: time)] $(\mathbf{r}', \mathbf{v}', i')$ by a sequence of random decisions regarding the type of collision process $k$, the species index of the post-collision test particle $i$, and the new post-collision velocity $\mathbf{v}$. Collisions are instantaneous in time. Branching and/or absorption is accounted for by statistical weights.

It is strictly not necessary, but usually most convenient and intuitively most clear, to decompose the kernel $C$ according to the physical nature of the various collision processes taken into account. The collision kernel $C$ was written as (see (1.9))

$$C(\mathbf{r}', \mathbf{v}', i' \to \mathbf{v}, i) = \sum_k p_k C_k(\mathbf{r}'; \mathbf{v}', i' \to \mathbf{v}, i) \ , \ p_k = \frac{\Sigma_k}{\Sigma_t}$$

with summation over the index $k$ for the different types of collision processes under consideration and $p_k$ defined as the probability for a collision to be of type $k$. The normalizing factor

$$c_k(x') = \sum_i \int d\mathbf{v} \, C_k(\mathbf{r}', \mathbf{v}', i' \to \mathbf{v}, i) \ , \ C_k = \frac{1}{c_k} C_k$$

was the mean number of secondaries for this collision process $k$. The normalized function $C_k$ then is a conditional probability density for post-collision state $(\mathbf{v}, i)$, given the pre-collision coordinates $(\mathbf{r}', \mathbf{v}', i')$.
Sampling from $C$ (given that a collision of some kind has occurred) proceeds most conveniently by firstly sampling the type $k$ of the process from the discrete distribution

$$(p_k, \quad k = 1, \ldots, K),$$

and then by sampling the post-collision state of the test-particle from $C_k$. Finally the weight of the test-flight after collision is increased (or reduced) by multiplying it with the normalization constant $c_k$.
The decomposition of $C$ into sub-kernels $C_k$ is somewhat arbitrary, and the criteria may also be computational aspects rather than the different physical nature of the various processes taken into consideration.
Sampling $(\mathbf{v}, i)$ from $C_k$ can be done along the same idea: first factoring out the discrete distribution for the species index $i$, and then by sampling $\mathbf{v}$ from $C_{k,i}$, the conditional distribution for $\mathbf{v}$ given the type of the process is $k$ and the species of the post-collision trajectory is $i$. Let $(n_{k,i}, i = 1, \ldots, I_k)$ be the number of post-collision particles of species $i$ from process $k$, and $n_k = \sum_i n_{k,i}$. I.e., omitting pre-collision state parameters $(\mathbf{r}', \mathbf{v}', i')$:

$$n_{k,i} = \int d\mathbf{v} \, C_k(\mathbf{r}', \mathbf{v}', i' \to \mathbf{v}, i)$$

Then, normalizing,

$$(\tilde{n}_{k,i} \quad i = 1, \ldots, I_k) = (n_{k,i}/n_k, \quad i = 1, \ldots, I_k)$$

is the discrete distribution for the post-collision particle species $i$, given a collision process of type $k$.
Finally, writing

$$C_k = \sum_i n_{k,i} w_{k,i}(\mathbf{v})$$

defines the post-collision velocity distribution $w_{k,i}$, given: a collision has occurred at pre-collision state $(\mathbf{r}', \mathbf{v}', i')$, then a type of process $k$ has been sampled, then a post-collision species $i$ has been identified. Sampling $(\mathbf{v})$ from $w_{k,i}(\mathbf{v})$ is carried out in routines VELOCX, VELOEL, VELOEI, VELOPI, depending on the type $k$ of process identified. In all cases first a transformation into the rest frame of the interacting background species $i'_b$ is carried out.

### 1.3.7 elastic collisions, VELOEL

In elastic collisions (EL) colliding particles retain their identity: $(i' = i, i_b = i_{b'})$. A binary collision is simulated by first identifying the velocity $\mathbf{v}_{b'}$ of the pre-collision partner from the background medium. As in all heavy particle collisions the distribution for finding $\mathbf{v}_{b'}$ is:

$$w_{k,i_{b'}}(\mathbf{v}_{b'}) \sim \sigma_{el}(|\mathbf{v}' - \mathbf{v}_{b'}|)|\mathbf{v}' - \mathbf{v}_{b'}| f_{i_{b'}}(\mathbf{v}_{b'})$$

with $f_{i_{b'}}(\mathbf{v}_{b'})$ the velocity distribution of background medium $b$. EIRENE typically assumes here either a drifting Maxwellian in the laboratory frame, or a mono-energetic isotropic distribution in the rest frame of background medium $b$. Next the scattering angle $\theta$ is found in the center of mass frame, either from the total cross section and the interaction potential, for from differential cross sections, for other, more approximate assumptions. Finally the post-collision velocity $\mathbf{v}$ is obtained by transforming back into the laboratory frame.

### 1.3.8 charge-exchange, VELOCX

In this case an exchange of identity is assumed: $i = i_{b'}, i_b = i'$, which corresponds to a scattering angle $\theta = \pi$ in the center of mass system. In this special case sampling is most convenient in the in the rest frame of background medium $b$, rather than in the center of mass frame. This means again first sampling $\mathbf{v}_{b'}$ from

$$w_{k,i_{b'}}(\mathbf{v}_{b'}) \sim \sigma_{cx}(|\mathbf{v}' - \mathbf{v}_{b'}|)|\mathbf{v}' - \mathbf{v}_{b'}| f_{i_{b'}}(\mathbf{v}_{b'})$$

and then, after transformation from the rest frame of species $b$ back into the laboratory frame, simply setting $\mathbf{v} = \mathbf{v}_{b'}$

### 1.3.9 electron-impact collisions, VELOEI

### 1.3.10 general heavy-particle-impact collisions, VELOPI

### 1.3.11 photon processes (emission, absorption, scattering)

documentation to be written

## 1.4 Surface Reflection Models

Within the terminology introduced in section 1.2 the effects of the interaction of neutral particles, photons (and escaping ions) with surfaces surrounding or inside the computational volume can be described by additional boundary conditions. Such kinetic boundary conditions are then often referred to as "bidirectional reflectance functions". It is, however, much more convenient for Monte Carlo applications to describe this interaction as just one special type of collision process in the collision kernel $C$ of equation (1.9), i.e. as

$$C = p_w \cdot C_w + \sum_k p_k C_k, \tag{1.55}$$

with $p_w(\mathbf{r}) = 1$, if $\mathbf{r}$ is a point on a surface, and $p_w(\mathbf{r}) = 0$ elsewhere. The transport kernel $T$ can be modified such that the maximum length $l_{max}$ for free flight to the next point of collision is the distance to the nearest surface along the track of the particle. The reflection kernel $C_w$ is further decomposed it into a kernel $C_{wf}$ (fast particle reflection), $C_{wt}$ (thermal particle re-emission) and $C_{wa}$ (particle absorption) with respective probabilities, $p_f$, $p_t$ and $p_a$, such that

$$C_w = C_{wf} + C_{wt} + C_{wa} = p_f C_{wf} + p_t C_{wt} + p_a C_{wa}, \quad p_f + p_t + p_a = 1. \tag{1.56}$$

Here $C$ are, as above, the normalized versions of the collision kernels $C$.
We discuss in detail only the fast particle reflection model $C_{wf}$ since $C_{wt}$ is generally a mono-energetic and cosine angular distribution or a Maxwellian flux distribution at wall temperature, and $C_{wa}$ only describes the transition into the "limbo state" $x_a$ "absorbed particle".
It is usual in neutral gas models to replace $C_{wf}$ with numerical or analytical fits to moments of $C_{wf}$, e.g., such as the particle and energy reflection coefficients $C_P$ and $C_E$. The particle - wall interaction is often supplemented by simple assumptions on the angular distribution (cosine or specular) of the particles leaving the surface.
One has for the particle reflection coefficient:

$$C_P = \sum_i \int d\mathbf{v}\, C_{wf}(\mathbf{r}; \mathbf{v}', i' \rightarrow \mathbf{v}, i) \tag{1.57}$$

Note that $C_P = p_f = c_{wf}$ in the terminology of (1.10).
The energy reflection coefficient is defined as:

$$C_E = \frac{1}{E'} \cdot \sum_i \int d\mathbf{v}\, E \cdot C_{wf}(\mathbf{r}; \mathbf{v}', i' \rightarrow \mathbf{v}, i) = \frac{1}{E'} \cdot \tilde{C}_E \tag{1.58}$$

Here $E = mv^2/2$ is the energy of the reflected particle. Hence $C_E = \bar{E}/E' \cdot C_P$, with $\bar{E}$ denoting the mean energy of the reflected particles.

### 1.4.1 The Behrisch matrix reflection model

One model, the matrix model due to Behrisch [**kn:Behrisch**], is hardwired in EIRENE (and in many other codes such as AURORA, BALDUR, TRANSP developed in the early eighties of the last century). This model has been used as a standard option in many benchmarks [**kn:Benchmark**]. The model is based upon a table of values for $C_{P_\perp}(E')$ and a stochastic matrix

$\underline{\underline{B}}_{\perp}$ for the transition $E' \rightarrow E$, for H atoms striking a stainless steel target at normal incidence. For other incident angles $\vartheta_{in}$ the moments are modified in EIRENE by the relations:

$$C_P(\vartheta_{in}) = 1 - (1 - C_{P_\perp}) \cdot \cos^{e1}(\vartheta_{in}) \tag{1.59}$$

and

$$C_E(\vartheta_{in}) = 1 - (1 - C_{E_\perp}) \cdot \cos^{e2}(\vartheta_{in}). \tag{1.60}$$

This latter moment equation can be satisfied by modifying the tabulated stochastic matrix $\underline{\underline{B}}_{\perp}(E' \rightarrow E)$ in many different ways. In EIRENE, we first sample $\mathcal{E}$ from $\underline{\underline{B}}_{\perp}$ (given $E'$) and then set

$$E = E' - (E' - \mathcal{E}) \cdot \cos^{e2}(\vartheta_{in}). \tag{1.61}$$

The linearity of E in $\mathcal{E}$ gives an expectation value $\bar{E}$ with $\bar{E}/E' \cdot C_P(\vartheta_{in}) = C_E(\vartheta_{in})$, with $C_P$ and $C_E$ as given by equations (1.59) and (1.60), respectively.
To describe the angular distribution of re-emitted particles, conditional on incident energy and angle, let $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ define an orthonormal basis at the strike-point of a test-particle on a surface. Here let $\mathbf{e}_x$ be the unit vector parallel to the outer surface normal and let the incident particle travel in the local xz-plane. Furthermore, let $\vartheta_r$ and $\varphi_r$ be the polar and azimuthal angles sampled from a cosine distribution around $-\mathbf{e}_x$. The speed unit vector $\mathbf{\Omega}$ of the reflected particle in EIRENE is then given by

$$\begin{aligned}
\Omega_x &= -\cos(\hat{\vartheta}_r) f_1(\vartheta_{in}) + \sin(\hat{\vartheta}_r) \sin(\varphi_r) f_2(\vartheta_{in}), \\
\Omega_y &= \sin(\hat{\vartheta}_r) \cos(\varphi_r), \\
\Omega_z &= \sin(\hat{\vartheta}_r) \sin(\varphi_r) f_1(\vartheta_{in}) + \cos(\hat{\vartheta}_r) f_2(\vartheta_{in}),
\end{aligned} \tag{1.62}$$

with

$$\begin{aligned}
\sin(\hat{\vartheta}_r) &= f_1(\vartheta_{in}) \sin(\vartheta_r), \\
f_1(\vartheta_{in}) &= \cos^{e3}(\vartheta_{in}), \\
f_2(\vartheta_{in}) &= (1 - f_1 \cdot f_1)^{\frac{1}{2}}.
\end{aligned}$$

In total we have defined three parameters $e1, e2, e3$. The first two parameters $e1$ and $e2$ account for particle and energy reflection coefficients that continuously increase with incident angle (recommended: $e1 = 1$, $e2 = 0.5$ from comparison with TRIM code calculations, see below).
The third parameter $e3$ can be used to take into account an increasing contribution of specular reflection at near grazing incidence. $e3 = 0$ gives a pure cosine distribution, $e3 = 1$ reproduces the distribution given in [**kn:Heifetz**]. In general this model describes a fraction of specular versus cosine reflection, which increases with $\vartheta_{in}$. And in the (purely mathematical) limit of $\vartheta_{in} = 90°$, it results in pure specular reflection. Furthermore, the parameter $e3$ controls the speed at which the specular part $a_{spec}(\mathbf{\Omega}; \vartheta_{in})$ of the angular distribution $a(\mathbf{\Omega}; \vartheta_{in}) = a_{spec}(\mathbf{\Omega}; \vartheta_{in}) + a_{cosin}(\mathbf{\Omega}; \vartheta_{in})$ increases with incident angle $\vartheta_{in}$.
A simpler, "fixed momentum accommodation coefficient model", in which the dependence of this decomposition on incident angle $\vartheta_{in}$ is absent, can also be activated, e.g. for testing, (see flag: AINTG in cards for surface reflection models, section 2.6).
The interaction of neutrals and ions with a solid surface for target-projectile combinations other than H onto stainless steel are modelled using a reduced energy scaling.

### 1.4.2 TRIM code database reflection models

Complementary to the surface model described above, EIRENE is coupled to computer generated reflection databases, produced with the TRIM code (see references [**kn:Eckstein**] and [**kn:Bateman**]). At present a discretized form of the reflection kernel $C_{wf}$, including all correlations between the five vector components of $\mathbf{v}'$ and $\mathbf{v}$, is available for the target-projectile combinations listed in table 1.1. The first 12 of these combinations comprised an old default "TRIM data file" of such combinations in an EIRENE run. Meanwhile a much extended list, also far beyond that of table 1.1, is available, see separate documentation of TRIM code bidirectional reflectance database, and the target projectile combinations to be used in any particular run are selected individually from this larger list (input block 6).
ZNML is a target material flag set for each surface in the input file, blocks 3a and 3b, and is described below in block 6b, "Data for local reflection models". These databases are, essentially, tables of conditional quantile functions obtained from large random samples of test particle trajectories in the solid.

Table 1.1: **Surface reflection database (TRIM Code).**

| No | Projectile | Target | ZNML |
|---|---|---|---|
| 1 | H | Fe | 5626. |
| 2 | D | Fe | 5626. |
| 3 | H | C | 1206. |
| 4 | D | C | 1206. |
| 5 | He | Fe | 5626. |
| 6 | He | C | 1206. |
| 7 | T | Fe | 5626. |
| 8 | T | C | 1206. |
| 9 | D | W | 18474. |
| 10 | He | W | 18474. |
| 11 | H | W | 18474. |
| 12 | T | W | 18474. |
| 13 | D | Be | 904. |
| 14 | D | Mo | 9642. |
| 15 | Ne | C | 1206. |
| 16 | Ne | Be | 904. |
| 17 | H | Cu | 6429. |
| 18 | H | Mo | 9642. |
| 19 | T | Mo | 9642. |
| 20 | He | Mo | 9642. |

We consider these "database -" reflection models to be the most complete option available at present and recommend to further extend the databases to additional projectile - target combinations, and in particular to sputtering and self-sputtering data (such as, e.g., C onto C, or Be, etc.).

For an INTOR benchmark case for neutral particle Monte Carlo Codes [**kn:Benchmark**] the choice of a particular surface reflection model had no strong impact on the results. This was shown by running the EIRENE code with all three models ("Behrisch Matrix", TRIM database and MARLOWE database (at that time available as a third option. Meanwhile the MARLOWE database is not available anymore).
It was found that this had little influence on global parameters as, e.g., the pumping efficiency for the particular geometry (5.4 %, 5.3 % and 5.2 % respectively). On the other hand, it can easily be imagined that in other cases (different plasma conditions, more detailed geometries) the choice of the reflection kernel $C_{wf}$ can be more important for the result. Certainly in all cases, in which the results are sensitive to the neutral particle velocity distribution near surfaces, details of the reflection kernel matter. This is, for example, the case for computed line shapes of certain electronic transitions ($H_\alpha$ lines etc.), see e.g., reference [**kn:Reiter92d**].

## 1.5 Recycling surface sources

It has been found (mid eighties of last century: within INTOR framework) as a result of benchmark calculations with several different neutral gas transport Monte Carlo codes (EIRENE, DEGAS, NIMBUS, DDC83) on the same input data that the main divergence in the results from the different codes tested was due to different primary source terms $Q$ in the transport equation (1.13a), (1.a) and (1.b), although target fluxes and temperatures had been prescribed to be identical. Here we describe one particular EIRENE surface source option in detail. The way the source is modelled by EIRENE allows consistency with kinetic boundary conditions for plasma fluid equations at plasma facing surfaces. To describe this, we first write the volumetric source $Q$ (particles per unit time and per unit $(\mathbf{r}, \mathbf{v})$-phase space volume) in terms of a surface flux $\Gamma_w$ and a surface delta-function. The surface (wall) flux $\Gamma_w$ thus has dimension: particles per unit time, per velocity space unit, and per unit area:

$$Q_w(\mathbf{r}, \mathbf{v}, i) = \Gamma_w(\mathbf{r}, \mathbf{v}, i) \cdot \delta_w(\pi), \tag{1.63}$$

where $\pi$ is a coordinate normal to the surface $S_w$: $\pi(\mathbf{r}) = 0$, i.e.: $\mathbf{e}_\pi = \nabla_\mathbf{r}\pi(\mathbf{r})$, pointing away from the computational volume. We see that $\Gamma_w$ specifies the (ingoing) "transport" flux $\Phi$ introduced in (1.5a) at the boundary of the computational domain. We will henceforth use the notation $\Gamma$ rather than $Q$, in case of surface sources. $\Gamma_w$ is a kinetic (microscopic) surface source (flux-) density distribution, backward directed (into the computational volume, i.e., it is non-zero only for $v_\pi < 0$).

To convert from a forward directed ion flux distribution $\Gamma_{Sh,ion}(\mathbf{v})$ at the plasma-sheath interface $S_{Sh}$ to a neutral flux distribution back from the wall surface $S_w$, we fold $\Gamma_{Sh,ion}$ with a (magnetic plus electrostatic) sheath transmission kernel $T_{Sh}$ and with the surface reflection (incl. sputtering) kernel $C_w$, equation (1.55):

$$\begin{aligned}\Gamma_w(\mathbf{r}, \mathbf{v}, i) = \sum_{i'} \int\int\int d^3r' d^3v' d^3v'' \Gamma_{Sh,ion}(\mathbf{r}', \mathbf{v}'', i') \\ \cdot T_{Sh}(i'; \mathbf{r}', \mathbf{v}'' \rightarrow \mathbf{r}, \mathbf{v}') \cdot C_w(\mathbf{r}; \mathbf{v}', i' \rightarrow \mathbf{v}, i)\end{aligned} \tag{1.64}$$

The summation over $i'$ is performed over all ion species entering the sheath region.

A typical example for $\Gamma_{Sh,ion}$ is at least sonic parallel plasma flow (Mach number $M_\parallel \geq 1$ along the magnetic field lines) entering the magnetic pre-sheath. In this magnetic pre-sheath the ion trajectories are bend over (accelerated by the electric pre-sheath field) such that at the entrance of the electrostatic sheath $\mathbf{r}'_{eSh}$ (locate intermediate between magnetic pre-sheath entrance $\mathbf{r}'$ and target surface $\mathbf{r}$ at least sound speed is already reached with respect to the surface normal: $M_\pi \geq 1$.

If the magnetic field is directed normal to the target surface, then the magnetic pre-sheath is absent, $\mathbf{r}' = \mathbf{r}'_{eSh}$ and, of course $M_\parallel = M_\pi \geq 1$.

A commonly adopted simplification is made in plasma fluid descriptions by ignoring the typically small spatial volume between the plasma sheath interface and the wall surface. Hence: $T_{Sh}$ factorizes into a 3D delta function $\delta^3(\mathbf{r} - \mathbf{r}')$ and a deterministic transition kernel for the velocities $\mathbf{v}'' \rightarrow \mathbf{v}'$.

Random numbers from $\Gamma_w$ are generated conveniently by sampling, firstly, $(\mathbf{r}', \mathbf{v}'', i')$ from $\Gamma_{Sh,ion}$, next generating $(\mathbf{r}, \mathbf{v}')$ from $T_{Sh}$ and then, finally, sampling the weights, velocities and species

indices of re-emitted (neutral) particles $(\mathbf{v}, i)$ from the conditional distribution $C_w$, which is conditional on one ion (species $i'$) having hit the target at position $\mathbf{r}' = \mathbf{r}$ and with velocity $\mathbf{v}'$.
A frequently employed assumption for $\Gamma_{Sh,ion}$ is a one sided (only positive velocity components into the sheath) forward drifting Maxwellian ion flux distribution at the interface between plasma and electrostatic sheath $S_{Sh}$. In this case the bending of trajectories from parallel (to **B**) to normal (to the surface) sonic flow within the magnetic pre-sheath is already included in $\Gamma_{Sh,ion}$, e.g., via boundary conditions in a **CFD!** model for plasma ion flow, and the action of operator $T_{Sh}$ is solely the acceleration in the electrostatic sheath.
Alternatively, and in cases when the magnetic field is tilted with respect to the wall surface, one may define $\Gamma_{Sh,ion}(\mathbf{v})$ to be the kinetic transport flux distribution entering the magnetic pre-sheath, i.e. with a drift velocity of at least ion sound speed in the direction parallel to the magnetic field **B** (rather than normal to the wall). In this case the action of the magnetic pre-sheath must be accommodated in the sheath kernel $T_{Sh}$, which we might even consider to split into a magnetic pre-sheath (MP) kernel and an electrostatic sheath (Sh) kernel:

$$T_{Sh} = T_{mpSh}(\mathbf{r}_{mpSh}, \mathbf{v}_{mpSh} \rightarrow \mathbf{r}_{eSh}, \mathbf{v}_{eSh}) \cdot T_{eSh}(\mathbf{r}_{eSh}, \mathbf{v}_{eSh} \rightarrow \text{wall}) \tag{1.65}$$

with the subscripts $mpSh$ and $eSh$ labelling the plasma side magnetic pre-sheath entrance and the electrostatic sheath entrance, respectively.

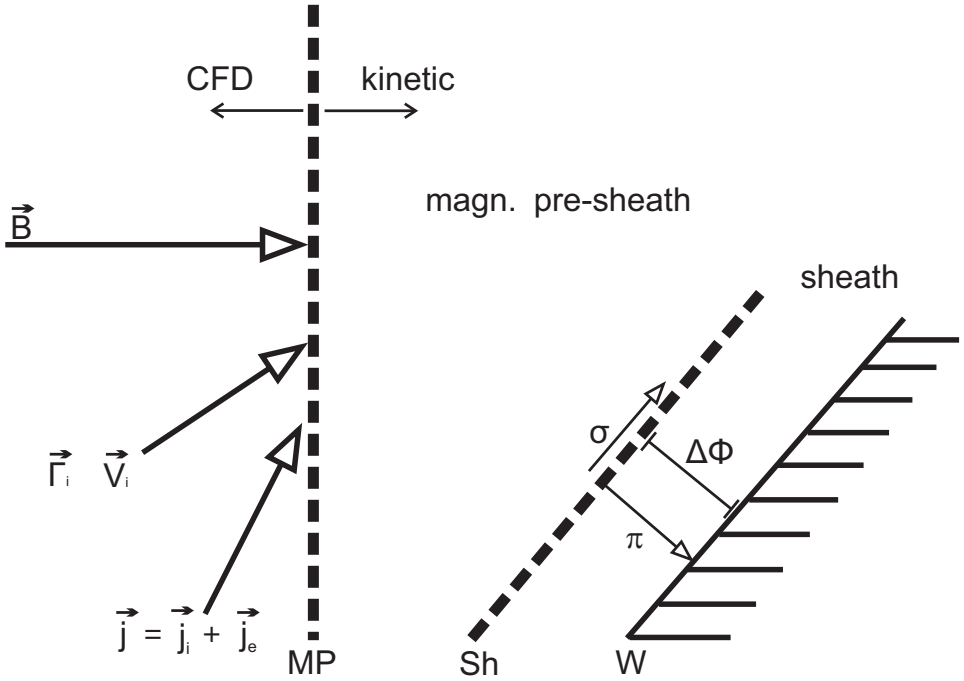



Figure 1.1: Transition region between fluid plasma (**CFD!**) and wall surface, schematic. The plasma flow is specified by **CFD!** models at the entrance surface of the magnetic pre-sheath MP (taken normal to the magnetic field $\vec{B}$). $\Delta\Phi$ is the fraction of the total sheath potential drop inside the electrostatic sheath.

### 1.5.1 truncated drifting Maxwellian flux at electrostatic sheath entrance

One of the standard options in EIRENE, in particular in applications in which EIRENE is coupled iteratively to a plasma fluid code, is a truncated forward drifting Maxwellian ion flux distribution (options: NEMODS = 6,7, section 2.7). The kinetic plasma ion flux distribution entering the electrostatic sheath is given as

$$\Gamma^0_{Sh,ion} = \frac{1}{\alpha^2} \cdot v_\pi \cdot \exp\left(-\frac{1}{v^2_{T_i}}(\mathbf{v} - \mathbf{v}_d)^2\right), \quad v_\pi > 0 \tag{1.66}$$

with $v_{T_i} = \sqrt{2kT_i/m_i}$, the thermal ion velocity, normalizing constant $\alpha^2$ and with the drift velocity $\mathbf{v}_d$. See section 2.7 for a description of related EIRENE input flags. The superscript 0 indicates normalization of the flux to one particle per unit time and unit area:

$$\int_{v_\pi>0} d^3v\, \Gamma^0_{Sh,ion}(\mathbf{v}) = 1 \tag{1.67}$$

Consider now the three orthogonal velocity components $v_1, v_2, v_3$ chosen such that $v_3 = v_\pi$, i.e. the $v_3$ velocity component is normal to the surface (target), and hence also normal to the plasma-sheath interface. $\Gamma^0_{Sh,ion}(\mathbf{v})$ factorizes into the three uni-variate distributions

$$\Gamma^0_{Sh,ion}(\mathbf{v}) = \Gamma^0_{Sh,ion,1}(v_1) \cdot \Gamma^0_{Sh,ion,2}(v_2) \cdot \Gamma^0_{Sh,ion,\pi}(v_\pi) \tag{1.68}$$

The first two factors here are simply (shifted) Gaussian distributions and sampling random velocities $v_1, v_2$ from them is trivial (e.g.: Box Muller method), whereas the third factor is a one-sided drifting Maxwellian flux distribution:

$$\Gamma^0_{Sh,ion,\pi}(v_\pi) = \frac{1}{\alpha^2_\pi} \cdot v_\pi \cdot \exp\left(-\frac{1}{v^2_{T_i}}(v_\pi - V_{d,\pi})^2\right), \quad v_\pi > 0 \tag{1.69}$$

The normalizing factor $\alpha^2_\pi$ for this (now uni-variate) distribution of the normal component $v_\pi$ reads:

$$\alpha^2_\pi = \frac{1}{2} v^2_{T_i} \cdot \exp(-\mathcal{V}^2_\pi) g(\mathcal{V}_\pi), \tag{1.70}$$

with $\mathcal{V}_\pi = V_{d,\pi}/v_{T_i}$ and $g(x) = 1 + \sqrt{\pi}\, x\, [1 + \mathrm{erf}(x)]\, \exp(x^2)$.
(Note that the dimensionless velocity $\mathcal{V}$ normalized by the thermal speed introduced here is related to the Mach number $M$ (normalized by the ion sound speed) by

$$\mathcal{V} = \frac{V}{v_{T_i}} = \frac{V}{c_s} \frac{\sqrt{Z_i\, T_e/T_i + 1}}{\sqrt{2}} = M\, \frac{\sqrt{Z_i\, T_e/T_i + 1}}{\sqrt{2}} \tag{1.71}$$

with $M = V/c_s$ and the (isothermal) ion sound speed $c_s$ defined as: $c_s = \sqrt{(Z_i\, T_e + T_i)/m_i}\, Z_i$ is the ion charge state. Hence $\mathcal{V} = M$ only if $Z_i\, T_e = T_i$.
It is clear that random sampling the $\pi$-component $v_\pi$ of $\mathbf{v}$ from $\Gamma^0_{Sh,ion,\pi}$ is not trivial even for this seemingly simplest sheath entrance ion velocity distribution, because the cumulative distribution function $F$ of a 1D forward drifting Maxwellian (with cut-off at $v_\pi = 0$):

$$F(v_\pi) = \int_0^{v_\pi} dv'_\pi\, \Gamma^0_{Sh,ion,\pi}(v'_\pi) \tag{1.72}$$

cannot be inverted explicitly.
Therefore in the EIRENE code we apply the Monte Carlo "biased source sampling" concept, (module: VELOCS.F). We sample this component $\tilde{v}_\pi$ from a "non-analog" truncated Maxwellian flux distribution $\tilde{\Gamma}^0_{Sh,ion,\pi}(\tilde{v}_\pi)$ (see section 1.3.3.1 above) with zero drift in $\pi$-direction but with different "non-analog" $\pi$- temperature $\tilde{T}_{\pi,i}$, ($\tilde{T}_{\pi,i} \neq T_i, \tilde{V}_{d,\pi} = 0$).

$$\tilde{\Gamma}^0_{Sh,ion,\pi}(\tilde{v}_\pi) = \frac{1}{\tilde{\alpha}_\pi^2} \cdot \tilde{v}_\pi \cdot \exp\left(-\frac{1}{v^2_{\tilde{T}_{\pi,i}}}(\tilde{v}_\pi)^2\right), \quad \tilde{v}_\pi > 0 \tag{1.73}$$

Note that although this non-analog distribution clearly violates any "Bohm criterion", it is, taken together with the weight correction factors described below, unbiased and a fully legitimate sampling distribution because it fulfils the "Radon–Nikodym condition" (1.32). Sampling from this non-analog distribution can be done by setting, e.g.,

$$\tilde{v}_\pi = \tilde{\alpha}_\pi \cdot \sqrt{-\ln(\xi) \cdot 2} = v_{\tilde{T}_{\pi,i}} \cdot \sqrt{-\ln(\xi)} \quad \text{since:} \quad \tilde{\alpha}_\pi = \frac{1}{\sqrt{2}} v_{\tilde{T}_{\pi,i}} \quad \text{for} \quad \tilde{V}_{d,\pi} = 0 \tag{1.74}$$

$\xi$ being a uniformly distributed random number between 0 and 1.
In order to find the weight correction factor associated with this non-analog procedure, we first note that from (1.71) we have

$$\tilde{\mathcal{V}}_\pi = 0 \rightarrow g(\tilde{\mathcal{V}}_\pi) = 1, \text{and} \quad \tilde{\alpha}_\pi = \frac{1}{\sqrt{2}} v_{\tilde{T}_i}$$

This gives the following result for the (multiplicative) weight correction factor $w(\tilde{v}_\pi)$, see equation (1.31), to be applied for each sampled non-analog velocity component $\tilde{v}_\pi$:

$$w(\tilde{v}_\pi) = \frac{\Gamma^0_{Sh,ion,\pi}(\tilde{v}_\pi)}{\tilde{\Gamma}^0_{Sh,ion,\pi}(\tilde{v}_\pi)} = \frac{\tilde{T}_{\pi,i}}{T_i} \exp\left(\frac{2\tilde{v}_\pi V_{d,\pi}}{v^2_{Ti}} - \frac{\tilde{v}^2_\pi}{v^2_{T_i}}\left(1 - \frac{T_i}{\tilde{T}_{\pi,i}}\right)\right) \cdot \frac{1}{g(\mathcal{V}_\pi)} \tag{1.75}$$

This non-analog sampling of the source term introduces additional statistical fluctuations in the results, since the initial test particle weights $w$ fluctuate with variance

$$\sigma^2(w) = \int_0^\infty d\tilde{v}_\pi \; w^2(\tilde{v}_\pi) \cdot \tilde{\Gamma}^0_{Sh,ion}(\tilde{v}_\pi) - 1 = g\left(\frac{2\mathcal{V}_\pi}{\sqrt{b}}\right) / g(\mathcal{V}_\pi)^2 \cdot \frac{1}{(2-b)b} - 1 \tag{1.76}$$

where $b = 2 - T/\tilde{T}$. We use the ansatz $b = C \cdot \mathcal{V}_\pi + 1$ and compute numerically that value of C which minimizes $\sigma^2$ in the range $0 \leq \mathcal{V}_\pi \leq 1$. We find C = 0.6026. This means that the ratio of the "real" temperature $T_i$ and the non-analog temperature $\tilde{T}_{\pi,i}$ should be chosen to be $T_i/\tilde{T}_{\pi,i} = 1 - 0.6026 \cdot \mathcal{V}_\pi$. This choice optimizes the statistical performance in the given Mach number range. In principle, of course, any other $\tilde{T}_\pi$ could have been used without introducing bias into the algorithm.
The mean energy of ions generated by this sampling procedure is (note: the distribution $\Gamma^0$ is already normalized to flux one):

$$\begin{aligned} \overline{E}(\mathcal{V}_\pi, \mathcal{V}_\sigma) &= \frac{m_i}{2} \cdot \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_0^\infty d^3v' \; v'^2 \cdot \Gamma^0_{Sh,ion}(\mathbf{v}') = \\ &= \frac{m_i}{2} \cdot \int_{-\infty}^{\infty} dv_1 \int_{-\infty}^{\infty} dv_2 \int_0^\infty dv_\pi \; (v_1^2 + v_2^2 + v_\pi^2) \cdot \Gamma^0_{Sh,ion}(\mathbf{v}') \\ &= T_i \cdot \gamma_E, \end{aligned} \tag{1.77}$$

with the half sided integration for the $\pi$-component of $\mathbf{v}$, and:

$$\gamma_E = 2 + \mathcal{V}_\sigma^2 + \left[\mathcal{V}_\pi^2 + 0.5 \cdot \frac{g(\mathcal{V}_\pi) - 1}{g(\mathcal{V}_\pi)}\right], \tag{1.78}$$

Here we have decomposed the drift velocity $\mathbf{v}_d$ into a normal (to the surface) $\pi$-component $\mathbf{v}_{d,\pi}$ and a $\sigma$-component $\mathbf{v}_{d,\sigma}$ parallel to the wall surface.

$$\mathcal{V}_\sigma = \frac{|\mathbf{v}_d - \mathbf{v}_{d,\pi}|}{v_{T_i}} = \frac{|\mathbf{v}_{d,\sigma}|}{v_{T_i}}. \tag{1.79}$$

Note that for $\mathcal{V}_\pi = 0$ (1.78) reduces to the expected formula $\gamma_E = 2 + {\mathcal{V}_\sigma}^2$ for a half sided Maxwellian flux with only drift components parallel to the surface.
Furthermore, if $\mathcal{V}_\sigma = 0$ (normal incidence of the mean flow) and $\mathcal{V}_\pi \to 1$ (i.e. for the Bohm sheath condition $M_\pi \to 1$ in case of $Z_i\, T_e = T_i$, such that $cs = v_{T_i}$), then $\gamma_E$ for the forward sided fluxes approaches the value for total net heat fluxes from (full) drifting Maxwellian distributions: $\gamma_E = 2.449 + {\mathcal{V}_\pi}^2 \simeq \frac{5}{2} + {\mathcal{V}_\pi}^2$, as it must.
We conclude that the above scheme is both an accurate and efficient random number generator for truncated shifted Maxwellian flux distributions of particles incident onto or emitted from surfaces. Of course, after having generated $\tilde{\mathbf{v}}$ with the weight $w(\tilde{v}_\pi)$ and before reflecting the particle via the kernel $C_w$, the acceleration of an ion in a sheath potential in direction $\pi$ (normal to the target surface), as formalized in the sheath transmission kernel $T_{Sh}$ mentioned above in (1.64), can easily be included.

### 1.5.2 The electrostatic sheath

The electrostatic sheath potential in front of a target surface is derived from assuming a given net electrical current $j_{pl}$ and a known ion particle current (ion species $i$) $\Gamma_i^+$ (i.e., $j_i^+ = q_i \Gamma_i^+$ electrical ion current entering a plasma-target boundary layer (the "sheath"), with $q_i = eZ_i$ being the ion charge, $e$ the elementary charge and $Z_i$ the ion charge number) flowing over the target sheath edge. Both these currents $j_{pl}, \Gamma_i^+$, and the electron current
$j^- = q_e \Gamma^- = -e\Gamma^- = j_{pl} - \sum_i j_i^+ = j_{pl} - j^+$, (with $j^+$ denoting the total, sum over all ion species, ion current) are taken to flow parallel to the $\mathbf{B}$-field at the plasma-sheath edge (or, more generally: to form (different) angles $\Psi_{\Gamma^+, j_{pl}} > 0$ degrees, respectively, with the target surface normal $\pi$).
Often, but not necessarily, it is assumed: $j_{pl} = j^+ + j^- = 0$, i.e., locally ambipolar flow conditions.
We start with a Maxwell-Boltzmann density distribution $n_e(x)$ for the electrons, with electron temperature $T_e$, in the electron retarding electric field with potential $\Phi(x)$. Here $x$ is a coordinate parallel to the current flow, e.g. parallel to the magnetic field $\mathbf{B}$ in case of absence of drift flows. The electron current density along the field is then (after reduction, possibly, due to secondary electron emission):

$$j_T^- = -e \cdot 1/4 \cdot n_{e,T} \cdot v_e^{av} \cdot (1 - \gamma_{s.e.e.}) = -e \cdot 1/4 \cdot n_{e,S} \cdot \exp[e\Delta\Phi_{S,T}/kT_e] \cdot \sqrt{\frac{8kT_e}{\pi m_e}} \cdot (1 - \gamma_{s.e.e.})$$

Here $\gamma_{s.e.e.}$ is the secondary electron emission coefficient and $v_e^{av}$ is the average electron velocity. The subscripts $S$ and $T$ stand for the plasma-sheath interface and for the target surface, respectively and $\Delta\Phi_{S,T}$ is the (typically negative) voltage drop from the plasma sheath edge to the target.

In order to determine the sheath voltage drop $\Delta\Phi_{S,T}$ we equate the component normal to the target of this current density with the corresponding component of the net ion current density:

$$\left(-j_T^-\right)_\pi = -j_T^- \cdot \cos(\Psi) = (j_T^+ - j_{pl}) \cdot \cos(\Psi) = \left(j_T^+ - j_{pl}\right)_\pi$$

Hence the cosine of the angle $\Psi$ of direction of current flow (e.g. of field line inclination against the target surface normal, if the current flows parallel to **B**) cancels out, as does the elementary charge $e$. The ion electrical current to the target $j_T^+$ (= $(j_S^+)_\pi$, no ion sources within the sheath region) is given as

$$j_T^+ = e \sum_i Z_i \cdot n_{i,S} \cdot V_{\pi,i,S} = e \sum_i Z_i \cdot n_{i,S} \cdot M_{\pi,i,S} \cdot c_{i,S}.$$

The sum is over all ion species, $Z_i$ is the charge state of species $i$, $n_{i,S}$ is the corresponding ion density and $V_{\pi,i,S}$ the ion flow velocity component normal to the target, of ion species $i$, subscript $S$ indicated that these values are taken at the sheath entrance. $M_{\pi,i}$ is the Mach number of the corresponding current component, and $c_i$ is the ion acoustic speed for species $i$, again taken at sheath entrance.
Hence, combining the three previous relations, the sheath potential difference $\Delta\Phi$ in terms of given values (at sheath entrance) for $j_{pl}, T_e, \gamma_{s.e.e.}$ and ions flows $\Gamma_i$, reads (omitting the subscript $\pi$ for the components normal to the target in these quantities):

$$e\Delta\Phi/kT_e = \ln\left[\frac{\sqrt{2\pi}}{(1-\gamma_{s.e.e.})}\left(\sum_i\left(Z_i\frac{n_{i,S}}{n_{e,S}}\cdot V_{i,S}\sqrt{\frac{m_e}{kT_e}}\right) - \frac{j_{pl}}{e\cdot n_{e,S}\cdot\sqrt{\frac{kT_e}{m_e}}}\right)\right] \tag{1.80}$$

$$= \ln\left[\frac{\sqrt{2\pi}}{(1-\gamma_{s.e.e.})}\left(\sum_i\left(Z_i\frac{n_{i,S}}{n_{e,S}}\cdot M_{i,S}\cdot c_{i,S}\sqrt{\frac{m_e}{kT_e}}\right) - \frac{j_{pl}}{e\cdot n_{e,S}\cdot\sqrt{\frac{kT_e}{m_e}}}\right)\right] \tag{1.81}$$

This expression (1.80) is evaluated by the function SHEATH of the EIRENE code for the sheath potential drop $\Delta\Phi$, taking local plasma data for $T_e$ and $n_i$ and $V_{\pi,i}$ (index $i$ runs over all ion species comprising the ionic target surface flux. The local electron density $n_e$ at sheath entrance follows then from quasi-neutrality.
All these values may have a spatial dependency and are taken at a (random sampled, according to target outflow plasma flux distributions) point at a boundary surface of a **CFD!** plasma code, e.g., at the divertor target boundary plate, which, strictly, is at the sheath entrance $S$ in front of the target. The energy gain due to sheath acceleration for a particular ion $i_0$ is then $Z_{i_0}e\Delta\Phi$.
Special forms of (1.81) are often quoted in the literature, e.g. for single ion fluid cases. Writing for the ion acoustic speed $c_i$ explicitly

$$c_i = \sqrt{\left(\frac{\gamma kT_i + Z_i kT_e}{m_i}\right)}$$

($\gamma$ denoting the adiabatic coefficient) and hence:

$$c_i \cdot \sqrt{\frac{m_e}{kT_e}} = \left[\left(\frac{\gamma kT_i}{kT_e} + Z_i\right)\cdot\frac{m_e}{m_i}\right]^{1/2} \tag{1.82}$$

we recover well known expressions for simpler cases. Consider, e.g., a single ion fluid case, $n_e = Z_i n_i = n$, and $M_i = 1$ (Bohm sheath condition for isothermal flow, $\gamma = 1$), (1.81) reduces to:

$$e\Delta\Phi/kT_e = \ln\left[\frac{\sqrt{2\pi}}{(1-\gamma_{s.e.e.})}\left(c_{i,S}\sqrt{\frac{m_e}{kT_e}}\right) - \frac{j_{pl}}{e\cdot n\cdot\sqrt{\frac{kT_e}{m_e}}}\right] \tag{1.83}$$

$$= \ln\left[\frac{\sqrt{2\pi}}{(1-\gamma_{s.e.e.})}\left[\left(\frac{kT_i}{kT_e}+Z_i\right)\cdot\frac{m_e}{m_i}\right]^{1/2}\left(1-\frac{j_{pl}}{e\cdot n\cdot c_{i,S}}\right)\right] \tag{1.84}$$

the latter equality (1.84) follows after insertion of (1.82) into the former.
If we let, furthermore, the net electrical current be zero: $j_{pl} = 0$ and take zero secondary electron emission $\gamma_{s.e.e} = 0$, we then find the classical result:

$$e\Delta\Phi/kT_e = \frac{1}{2}\ln\left[\frac{2\pi m_e}{m_i}\left(\frac{T_i}{T_e}+Z_i\right)\right]$$

Hence, e.g., for a pure hydrogen plasma ($m_e/m_i \approx 1/1836$), and for $T_e \approx T_i$, we have $e\Delta\Phi/kT_e \approx -2.5$ and $\approx -2.84$ for a pure deuteron plasma. For a pure helium plasma (fully ionized, $Z_i = 2$) this sheath factor becomes $\approx -2.98$.

### 1.5.3 The magnetic pre-sheath

In most situations when EIRENE is applied to plasma surface interaction studies the ion-plasma flow velocity $\mathbf{v}_i$ is predominantly parallel to a **B** field: $V_{\parallel,i} \approx |\mathbf{v}_i|$, and it is this parallel ion flow velocity, which is usually available from **CFD!** plasma simulations, via boundary conditions specified for the (parallel component of the) momentum balance equation at the entrance of the magnetic pre-sheath ($mpSh$) rather than at the entrance of the electrostatic sheath $eSh$.
The sheath model of EIRENE then is to be supplemented by a model **PRE-SHEATH** for the transition of known currents and $\mathbf{v}_{i,mpSh}$ to $\mathbf{v}_{i,eSh}$, i.e. in particular to derive the currents $j_{\pi,eSh}$ and flow components $V_{\pi,i,eSh}$ entering the electrostatic sheath from $\mathbf{j}_{mpSh}$ and $\mathbf{v}_{i,mpSh}$
to be written. . .

## 1.6 Combinatorial description of geometry

EIRENE is a Monte Carlo Code. Defining geometry in a Monte Carlo code is complex and laborious.
Of the three typical options for definition of the target geometry in Monte Carlo Codes:

- "**CSG!** (**CSG!**)",
- "**BREP!** (**BREP!**)"
- "Voxel Geometry"

EIRENE uses a combination of the latter two concepts.

In **CSG!** a solid object is "constructed" from the intersection, union or other Boolean operations of several "half-spaces". Each half space in specified by a relation $F(x, y, z) \leq 0$, typically $F$ is a low order polynomial. For example the neutronics **MCNP!** (**MCNP!**) code used e.g. for ITER nuclear safety studies, is based on **CSG!**.

In **BREP!** geometry, surfaces and volumes are build up from constituent parts. These parts may be related with algebraic functions (elementary **BREP!**s) or complex functions such as Bezier functions or B-splines (advanced **BREP!**s). Most **CAD!** (**CAD!**) software uses **BREP!**s in their native definition of geometry.
In the EIRENE **BREP!** options surfaces are build from (bounded) algebraic functions. These surfaces are referred to as "additional surfaces", see section 2.3.2. EIRENE uses only algebraic functions, up to second order (some extensions to raise the order to fourth order have been started recently but are not ready to use).

In "Voxel geometry", a 2D or a 3D object is constructed from identically shaped volume elements ("voxels"). In EIRENE these may be triangles or quadrangles in 2D, or tetrahedra in 3D. New options to accommodate trilinear hexaedra as 3D voxels are currently being developed in connection with 3D edge codes such as EMC3. The bounding surfaces of these voxels are referred to as "standard grid surfaces" in EIRENE, see section 2.2. These volume discretization meshes (VOXELS) can be read into EIRENE from external files. For example the mesh in B2 **CFD!** plasma fluid code is read into EIRENE directly from the **CFD!** grid generator in coupled B2-EIRENE applications. Wall meshes are build from these by assigning a surface reflection model to some of the (voxel-) surfaces (section 2.3.1), whereas internal (grid-) surfaces remain transparent for flow of test-particles.
**Note added in 2008:**
An attempt has been made in 2008 to use **CAD!** files directly for 3D wall surfaces. This was done via the already existing **CAD!**-**MCNP!** interface. We had then tried to turn the **CSG!** input for **MCNP!** (see above) into a **BREP!** form for EIRENE. In principle this works. But in practice the de-tour via the **CSG!** concept leads to prohibitively complex **BREP!** input for EIRENE.
A direct **CAD!**-**BREP!** interface seems now much more efficient to us and more natural anyway, and has meanwhile (2011) been established by an intermediate code step (the ANSYS code), which first converts **CAD!** output into a triangulation of surface elements, and then each triangle (in 3D space) is taken as "additional surface" (**BREP!** concept) into EIRENE. Very large sets of triangles required optimisation procedures, which have also been implemented (in EIRENE

module timea.f) on the basis of OCTREE concepts adopted from ray-tracing procedures used in image processing.

A complete set of surfaces in an EIRENE geometry may consist of a combination of both: standard surfaces (hence: defining standard grid cells, (voxels), and additional surfaces, defining a general **BREP!** discretization (additional cells). A more complicated situation arises when additional surfaces intersect standard grid voxels.

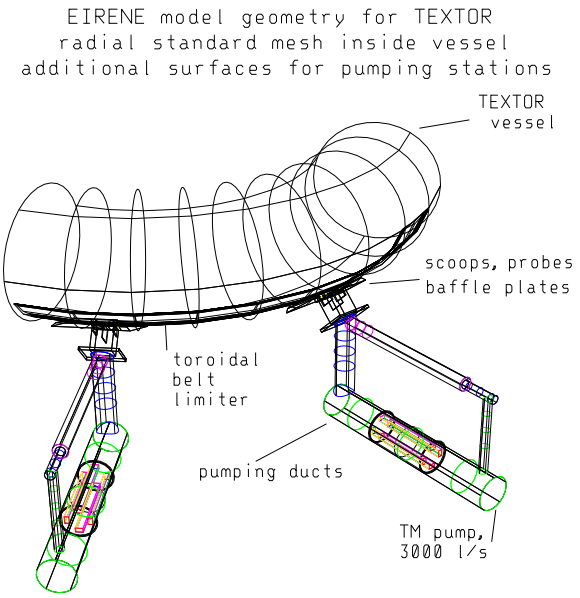



Figure 1.2 (a): Early example a combination of **BREP!** and VOXEL discretization, around 1985 for ALT-II pump limiter studies at **TEXTOR!** (**TEXTOR!**). Discretization inside the **TEXTOR!** tokamak vessel (torus): VOXEL, supplemented by about 100 (**BREP!**) first and second order surfaces for ALT-II scoops, pumping ducts and inner structures such as probe supports, getter pumps, etc..

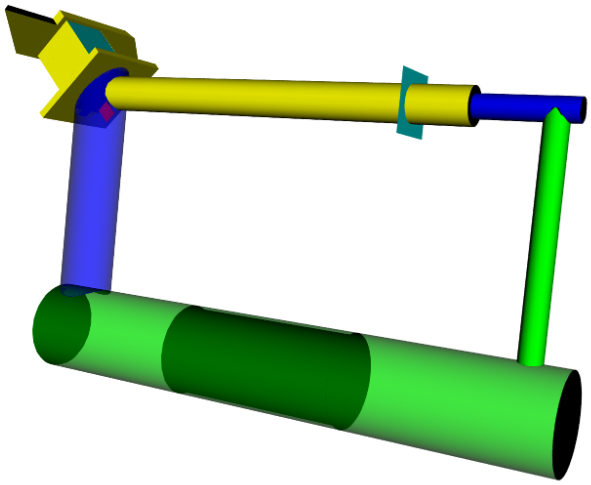


Figure 1.2 (b): Same as left Figure, but **BREP!** part only, and EIRENE re-run with more modern 3D visualisation graphics program, 2009.

If these surfaces are transparent, they can be used to conveniently "measure" fluxes at any position. But if they are reflecting, or absorbing, e.g. to define a complex shaped bounding surface in an otherwise simple grid, then some care is needed because the automatic cell volume evaluation, which is available in EIRENE for standard grid cells (voxels), may not correspond anymore to the cell volume accessible for test particles. Cell averaged quantities (divided by cell volume) may then not necessarily be scaled properly. For a possible work-around see NLERG option in input block 1, section 2.1.

From the above it follows than in an EIRENE model surface elements building up the wall can be defined as bounded algebraic surfaces, with triangles as special case, i.e. as "additional surfaces, **BREP!**", or can be bounding surfaces of the voxels, if the "grid of voxels" extends up to the real wall, or a combination of both.

As can be seen from the flowcharts, all the geometrical calculations needed for tracing test flights and for estimating volume- and surface averaged tallies are compiled in a "geometry block", which can easily be exchanged. In most cases (e.g., all installations of EIRENE outside FZ-Jülich) we load the most complete, fully 3D block "GEO3D", which allows fully 3 dimensional spatial resolution. Lower dimensional (0D, 1D or 2D) are contained as special cases, e.g. by choosing only grid surfaces with proper symmetry.
There is, in general, not a very large CPU benefit from running EIRENE with simpler specialized and geometry optimized blocks such as GEO2D or GEO1D for 2D or 1D models, respectively, because the saving in CPU costs in most cases is only around 10 - 20 %, so these options have not been supported since the late eighties of the last century anymore.

**The geometry module of EIRENE**

The following operations have to be executed by the block:
Suppose the three dimensional computational volume is discretized by a mesh of $n$ surfaces $S_i, i = 1, 2, \ldots, n$. Each of these surfaces is defined by a set of $\Lambda_i$ equations (unbounded surfaces):

$$f_\lambda^i(x, y, z) = 0, \qquad \lambda = 1, 2, \ldots, \Lambda_i, \tag{1.85}$$

and, optional, for each unbounded surface $\lambda$ a subset of $\Pi_\lambda$ inequalities (e.g. for turning the unbounded surfaces into finite segments):

$$g_{\lambda,\pi}^i(x, y, z) \leq 0, \qquad \pi = 1, 2, \ldots, \Pi_\lambda. \tag{1.86}$$

This means that for all unbounded surfaces $S_{i,\lambda}$ only the points, which fulfill all corresponding inequalities (1.86) are regarded by EIRENE as valid parts of the finite surface element.
Assume a test particle at position $\mathbf{r}_0$ = (X0, Y0, Z0) moving with speed unit vector $\mathbf{\Omega}_0$ = (VELX, VELY, VELZ). The block then has to provide the nearest intersection $\mathbf{r}_1$ of the straight line ("sample trajectory")

$$G(t) : \mathbf{r}_0 + t\mathbf{\Omega}_0$$

amongst all possible intersections with the surfaces $S_i$. Moving the "particle" from $\mathbf{r}_0$ to $\mathbf{r}_1$ and then repeating the block until the total track-length $l$ as sampled from the transport kernel $T(l)$ (1.11a) is reached, allows to generate histories with all information available along the track to evaluate the estimators mentioned in section 1.2.
In EIRENE $f$ and $g$ are general 2$^{\text{nd}}$ order (with linear surfaces as special case) algebraic equations of the form

$$\begin{aligned} h(x, y, z) = a_0 + a_1 \cdot x + a_2 \cdot y + a_3 \cdot z + a_4 \cdot x^2 + a_5 \cdot y^2 + a_6 \cdot z^2 + \\ + a_7 \cdot xy + a_8 \cdot xz + a_9 \cdot yz \end{aligned} \tag{1.87}$$

Hence in the geometrical block nothing else but second order equations

$$t^2 + p \cdot t + q = 0$$

have to be solved accurately and as efficiently as possible. We note in passing that each of this surfaces may be transparent, purely absorbing, semi transparent or (partially) reflecting. Surface sources may be defined on each surface. These surfaces may be "invisible" from one side and transparent, absorbing or reflecting from the other and each surface can "operate switches" if

intersected by a test flight, such as abandoning the evaluation of atomic or molecular reaction rates (entrance into vacuum regions) along the flight, abandoning evaluation of volume averaged responses (e.g., in some uninteresting regions) or abandoning geometrical calculations (for those surfaces, for which it can be known a priori that they will not intersect a flight path originating at a particular surface or in a particular cell).

## 1.7 time dependent mode of operation

The section is currently rewritten. Please refer to the -J. Nucl. Mater. paper: D. Reiter et al., Vol 220–222 (1995), 987–992 (ref. [**kn:Reiter94**]).
Details of options available in EIRENE for time dependent problems are also given in section 2.13.
The following issues have to be considered in case of time-dependent coupling schemes to external plasma fluid solvers, see fig. 1.5:
The coupling is explicit, hence **CFL!** (**CFL!**) type conditions for the time-step have to be obeyed. Here this means: the time-step must be shorter than the time between re-ionization and subsequent neutralization (in the volume or at the targets), because the fate of a particle during its intermediate ionized phase is not included in the explicit Monte Carlo procedure.
The noise level for collision estimators increases with smaller time-steps. Hence: avoid all collision estimators in t-dependent mode.
Please contact us for an update on this situation, if you wish to use that option.

## 1.8 non-linear effects: coupling to plasma fluid models

The section is currently rewritten. Please refer to the report Jül-2872, Feb 1994, by G. Maddison and D. Reiter for details (ref. [**kn:Maddison94**]).
The back-reaction of the host medium (plasma) on the neutral (test) particle kinetics renders the combined plasma-neutral transport problem non-linear already for the Monte Carlo solution of the "linear" Boltzmann equation. This non-linearity is not in the collision integral, but in the dependence of the collision parameters (collision rates, mean free paths, ...) on the plasma parameters. It can be dealt with by iterating a solver for the host medium (plasma) and for the kinetic equation (neutrals). Such a **CFD!** solver for 2D plasma flows is e.g. the B2-"BRAAMS" code, and its iterative coupling to the Monte Carlo kinetic code EIRENE was developed in the late eighties and early nineties of the last century, under NET-KFA contracts, involving three EFDA associates: the EIRENE group at FZ Jülich (formerly KFA), the CCFE (formerly AEA Culham) edge modelling group as well as ERM Brussels.
The historically oldest option is the "semi implicit" scheme, see fig. 1.4, because of the severe CPU time constrains in the late eighties of the last century, when B2-EIRENE was first developed, see Reiter et al., ([**kn:Net**]).
Later mostly the simpler "explicit scheme", fig. 1.3 was employed in applications, which, although probably being much slower in "code turn-around time", did require less attention by the person running the code, i.e. seemed to be numerically more robust, in particular in cold, recombining divertor plasma conditions.
Since about 2009 also available (but apparently not used in 3$^{rd}$ party applications) is a combination of both methods: The selection between explicit and semi-implicit iteration is made possible individually for each "stratum", such that now some parts of the problem can be run semi-implicitly, while others (notably the volume recombination neutral gas source) can remain explicit.
All these coupling schemes treat the kinetic part of the model in quasi-steady-state approximation with the plasma state. This is justified as long as the gas dynamics is fast compared to the plasma dynamics, the latter being roughly specified by the time step $\Delta t$ used in the **CFD!** plasma simulation.
In particular in high density, low temperature detached divertor plasmas this ordering may not necessarily be correct any longer, and a time dependent mode of operation of EIRENE for the coupling may be required due to **CFL!** constrains.
The full exploration of this option, which was implemented into EIRENE in the mid nineties of the last century for **ELM!** (**ELM!**) simulations, has not yet happened with respect to overall coupled code stability and "turn around code run time". The fig. 1.5 illustrates this mode of operation.

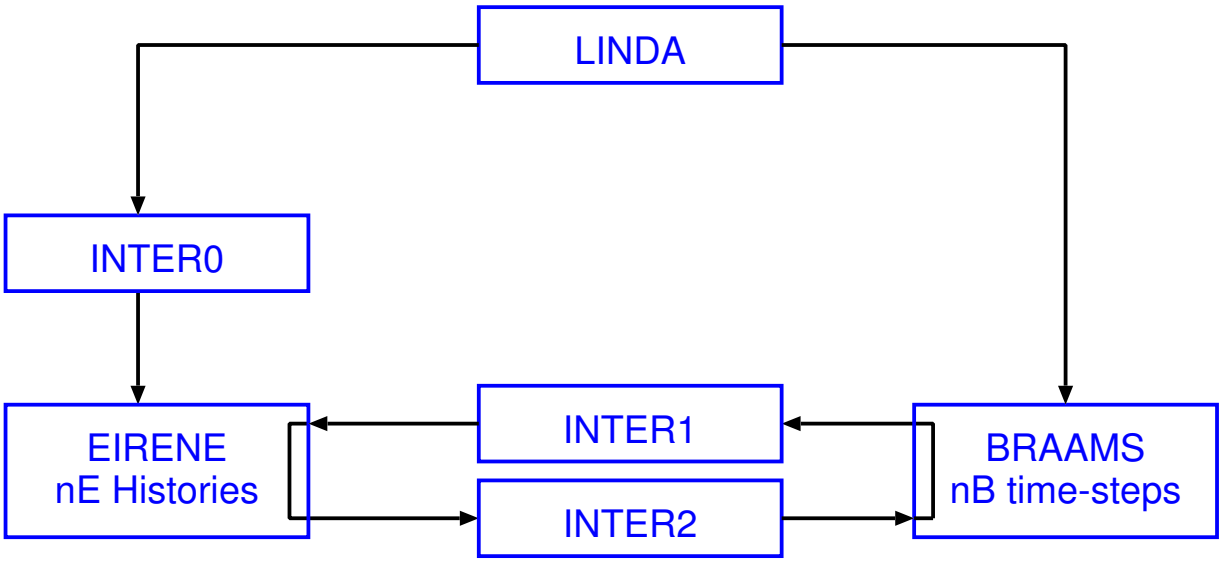



Figure 1.3: Explicit coupling scheme between EIRENE and a **CFD!** code (for plasma flow): After each plasma code cycle (or time step) a full new Monte Carlo run is carried out.

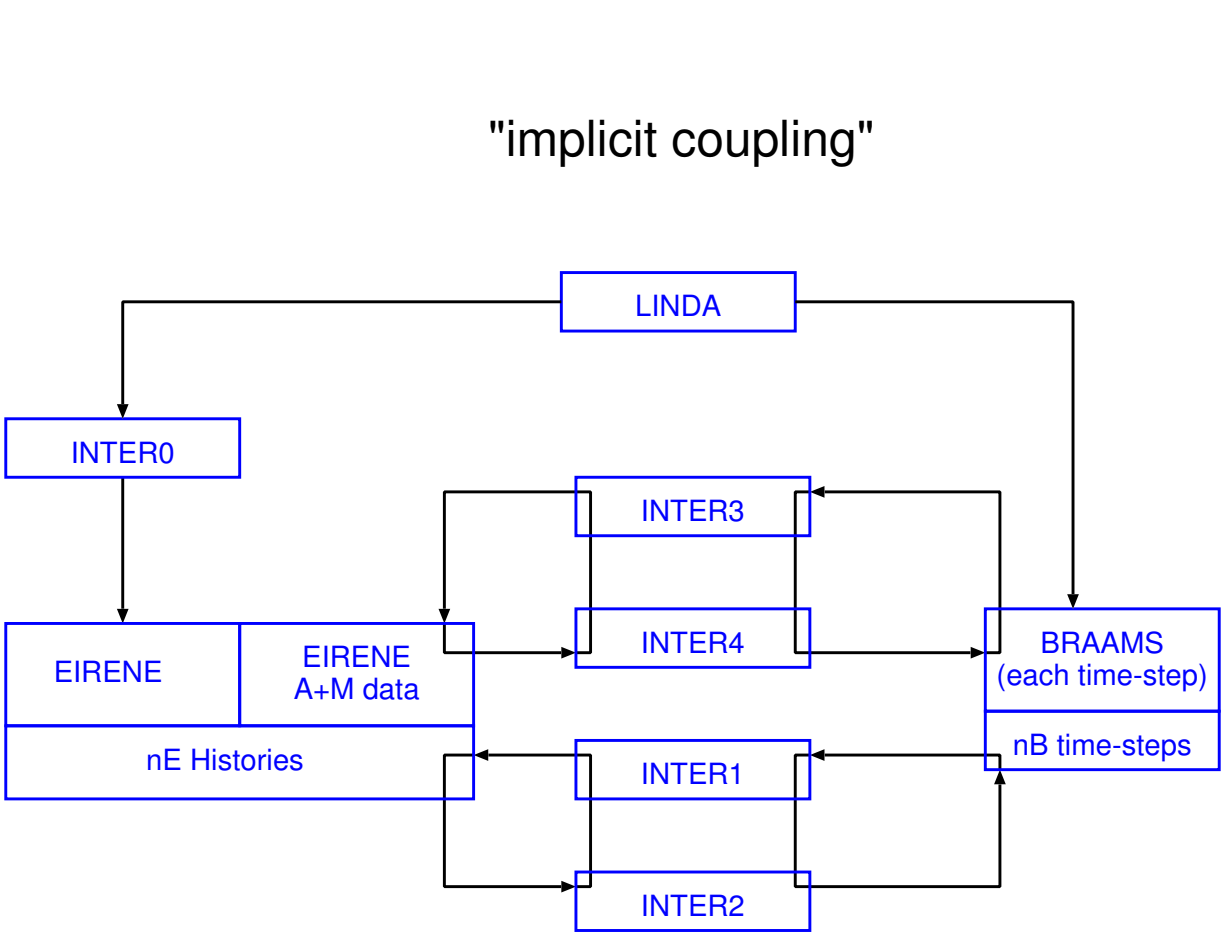



Figure 1.4: "Semi implicit" coupling scheme between EIRENE and a **CFD!** code (e.g. BRAAMS B2 for plasma flow), involving internal re-iteration on sources terms without new Monte Carlo trajectories, much less frequent Monte Carlo runs than in "explicit" mode of operation, but unresolved issues still re volume recombination contribution. LINDA is a grid generating tool providing a numerical discretization mesh which is common for B2 and EIRENE.

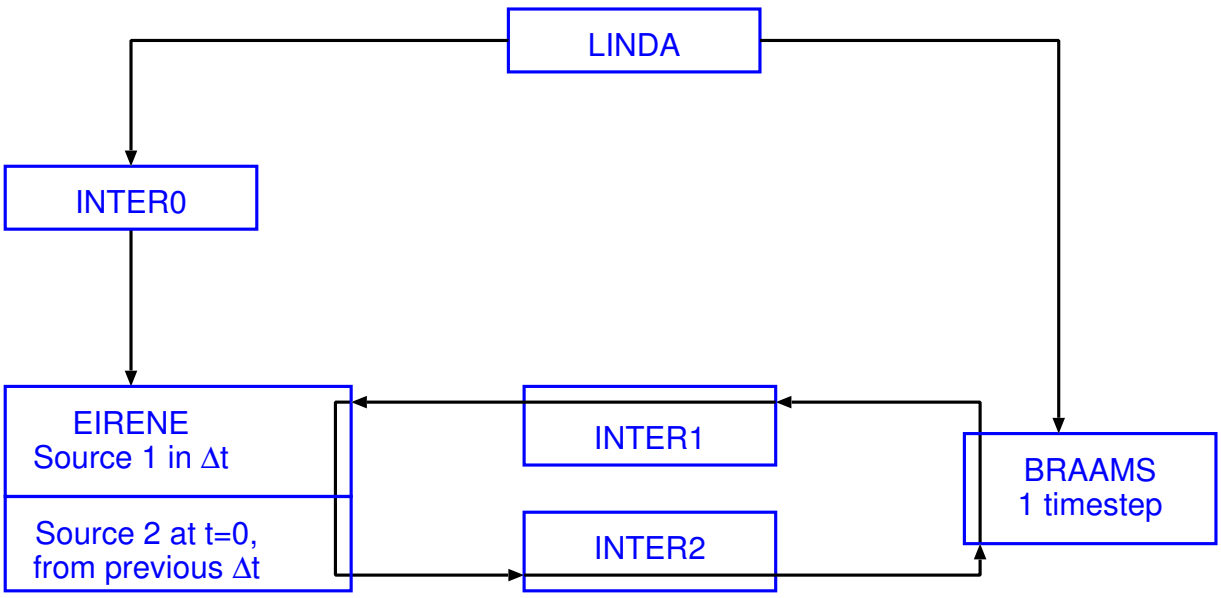



Figure 1.5: Time dependent coupling to plasma **CFD!** code.

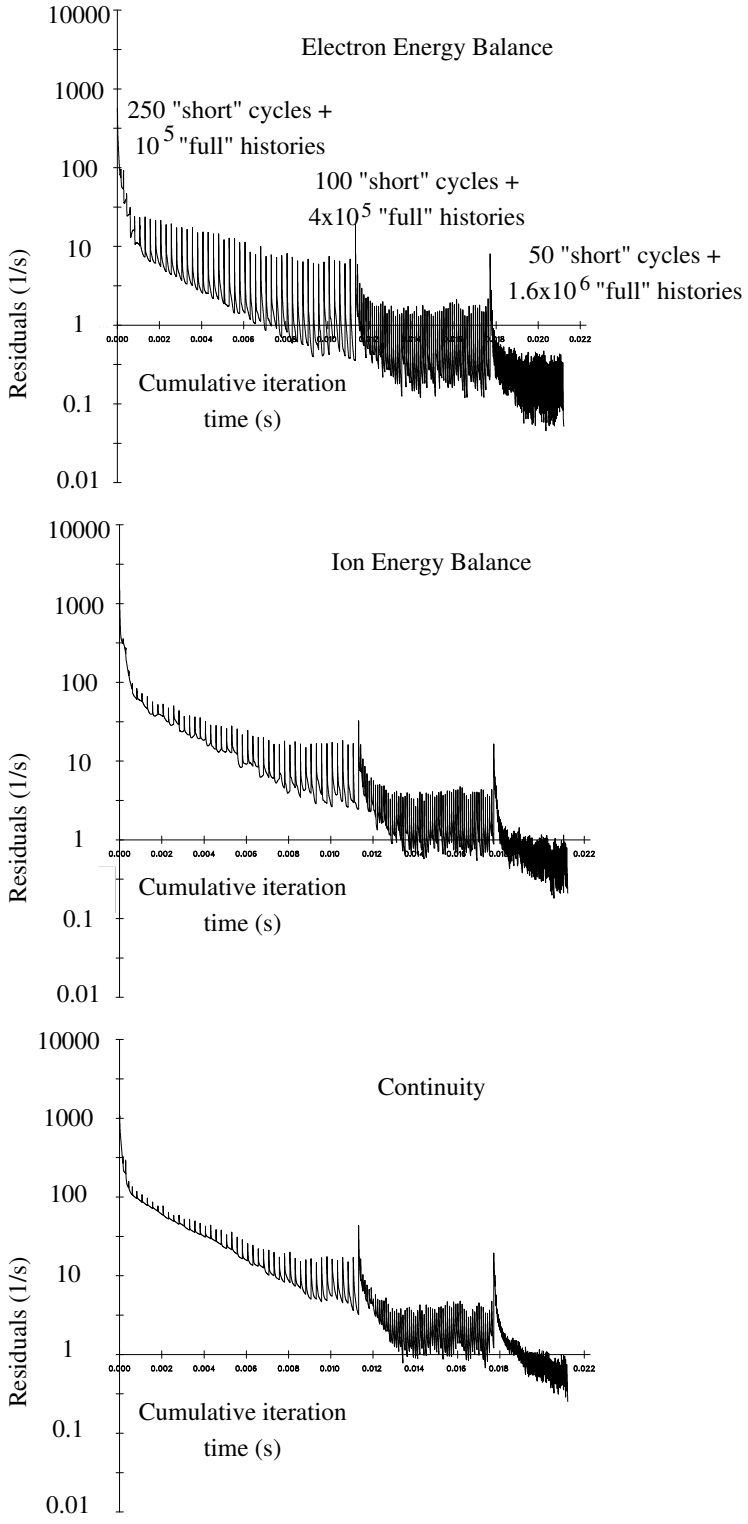
### 1.8.1 correlated sampling

A generic feature of coupled B2-EIRENE runs seems to be the loss of exponential decaying residuals (errors in B2 balances) down to machine precision, which, in stand alone B2 runs, is a signature of having achieved convergence. When coupled to EIRENE, at least with uncorrelated EIRENE trajectories between subsequent EIRENE calls (e.g.: between time steps in the explicit coupling code), residuals only saturate. The level of the saturated residuals then scales only with the square root of the number of EIRENE trajectories used per EIRENE cycle. Other convergence criteria then have to be defined, see [**kn:Maddison94**] for an early discussion of this still unsettled subject. The typical behaviour of residuals in coupled B2-EIRENE runs ("saturated residuals") is depicted in fig. 1.6.
Note: strong (strict) convergence (exponentially decaying residuals down to machine precision) can be achieved even in the presence of Monte Carlo noise, as is regularly seen in simplified and dedicated test cases, and, actually, in the very first applications of EIRENE in coupled plasma-neutral transport problems in the late eighties of the last century [**kn:Gerhauser88a**], there to **TEXTOR!** limiter configurations. This, however, seems to require the activation of the "correlated sampling" options of EIRENE in order to provide positive statistical correlation between successive time-steps (or iterations). However, except for strongly reduced (simplified) applications presently the level of positive correlation implemented via the EIRENE flag NLCRR (input block 1) seems to be too low to provide strong convergence on a regular basis and for full scale problems (e.g. the detached ITER divertor). In these latter, more complex cases the entropy producing collisions and surface interactions lead to a rapid de-correlation of trajectories, leading to a low level of correlation only, while, with the same algorithm, that level of correlation can be very high in geometrically and/or physically very simple cases.

For this consider two plasma solutions obtained with a **CFD!** plasma code coupled iteratively to EIRENE, at neighboring time-steps 1 and 2. The incremental change of the plasma solution due to the change in the Monte Carlo reaction term in the plasma flow equations is the difference between Monte Carlo source rates evaluated at time 1 and time 2.
In more general terms a Monte Carlo estimate of a difference $\Delta E$ between two different calculations $MC_1$ and $MC_2$ (due to either code version differences, due to a different physical incremental effects or due to iterative procedures with **CFD!** solvers) is:

$$\begin{aligned} \Delta E &= E(MC_1) - E(MC_2) \\ \sigma^2(\Delta E) &= \sigma^2(E(MC_1)) + \sigma^2(E(MC_2)) - 2cov(E(MC_1), E(MC_2)) \end{aligned} \tag{1.88}$$

(This holds even for any statistical estimate of an incremental effect, independent of Monte Carlo procedures). Here $E(MC)$ is the Monte Carlo estimate of the expectation value (i.e. it is the Monte Carlo solution to the kinetic transport equation), and $\sigma^2$ is the corresponding statistical variance. It is clear from (1.88) that small effects $\Delta E$ (such as those due to slightly different code versions) can only by detected when the statistical correlation quantified by the covariance of the two estimators $cov$, between the two Monte Carlo runs, is made large (EIRENE option: NLCRR).




h

Figure 1.6: Convergence of B2/EIRENE, measured by B2-residuals. Method: implicit coupling, method of “saturated residuals”, ref. [**kn:Maddison94**].

## 1.9 non-linear effects: neutral–neutral collisions

Neutral–neutral collisions make the Boltzmann collision integral non-linear (bi-linear). More generally, inclusion of any binary collisions between two test-particle species $i_0, j_0$, elastic or not, have this effect. Important examples are, of course, elastic self collisions $(i_0, i_0)_{el}$, e.g., $H_2 - H_2$ collisions leading to viscous effects in the gas, elastic cross collisions $(i_0, j_0)_{el}$, such as $H - H_2$ collisions, often leading to a heating of the molecular gas due to hot charge exchange atoms. Furthermore there may be relevant inelastic (chemical) binary particle collisions $(i_0, j_0)_{inel}$, such as $(H_2 - H_2^+ \rightarrow H + H_3^+)$ which is an important channel in hydrogen plasma chemistry. Finally, binary test particle interactions occur in case of radiation transport, for example $(h\nu + H(1s) \rightarrow H(n = 2))$, the absorption of resonance Lyman-alpha photons. In the intermediate regime between optically thin and (black body) LTE conditions the photons of a line from the discrete spectrum $h\nu$ become "test particles", which interact with other test particles, here with the ground state H atom, via line absorption. In large dense divertor plasmas the ratio $H(1s)$ to $H(n = 2)$ may have to be calculated consistently with the radiation field (a photon Boltzmann equation), competing electron impact collisions and neutral atom transport.
Two methods of solution for this non-linear problem have been tested in EIRENE, 1) a parametrization of the single particle distribution functions involved in non-linear collision terms and an iterative scheme (see next subsection: **BGK!** model collision terms), and 2) a **DMCS!** technique based upon a swarm simulation, an operator splitting ("Streaming" and "Collisions") and a random simulation of binary collisions in the swarm. Currently only the first of these two options is still being used. Despite its approximate character, it proved to be much more effective than the full **DMCS!** method, for the cases relevant to fusion research applications studied so far. A simple estimate for the relevance of neutral–neutral gas collisions is given by the so called Knudsen number $Kn$:

$$Kn = \frac{\lambda_{el}}{L} = \frac{1}{\sqrt{2}\sigma_{el} n_N L} \tag{1.89}$$

Here $\lambda_{el}$ is the mean free path for elastic neutral neutral collisions, $\sigma_{el}$ is the typical collision cross section ($cm^2$), $n_N$ the gas density ($cm^{-3}$) and $L$ is a characteristic dimension (cm). Fixing the hard sphere molecular diameter at $d = 4 \times 10^{-8}$ cm results in a typical cross section $\sigma_{el}$ of $1 \times 10^{-15}\,cm^2$ to $2 \times 10^{-15}\,cm^2$ to be used in this Knudsen number. For a gas temperature of 300 K and a pressure of 1 Pa ($n_N = 2.414 \times 10^{14}\,cm^{-3}$) we then obtain:

$$\lambda_{el}(\mathrm{cm}) = 1/(2.414 \times 10^{14} \times 2 \times 10^{-15}) \approx 2\,\mathrm{cm} \tag{1.90}$$

In the **BGK!** approximation to the Boltzmann collision integral (see below, section 1.9.1) only an "effective" mean free path appears, because the collision rate $\nu_{BGK}$ is an effective (relaxation) rate only. When the common assumption of an velocity independent collision rate is made, then this rate is chosen such that the **BGK!** model fixes just one single transport coefficient to a desired value in the continuum limit. E.g. either a certain viscosity, or binary diffusion, or heat conduction, or any one other selected transport coefficient in the continuum limit can be matched, when the Chapman–Enskog procedure is applied to derive continuum equations (Navier Stokes equations) from the Boltzmann equation with that particular **BGK!** binary collision model term. One default set of **BGK!** collision parameters used in EIRENE is based upon the concept of "diffusion volumes" to provide (weakly temperature dependent) binary diffusion coefficients $D_{12}$ (for cross collisions, e.g. for He + $H_2$) or a viscosity coefficient $\mu$ (for self collisions) in the gas

(see [**kn:Reiter94**]). This choice results in a **BGK!** collision rate coefficient $k(T) = k_0 T^{0.25}$ with the constant $k_0$ derived from such continuum limit considerations and the (experimentally determined) "diffusion volumes". Within this concept then a **BGK!** - mean free path [**kn:Kotov-Dis**] follows as:

$$\lambda_{BGK}(\mathrm{cm}) = \frac{\tilde{v}}{n\langle\sigma v\rangle} = \frac{\sqrt{8T/\pi m}}{n k_0 T^{0.25}}$$
$$= L_0 \times T[\mathrm{eV}]^{0.25}/n[1\times 10^{14}\,\mathrm{cm}^{-3}] \qquad (1.91)$$

where the proportionality factor $L_0$ (in cm) is typically in the range between 7 and 20 ($L_0 = 20$ cm for D–D self collisions, $L_0 = 9.75$ cm for $D_2$–$D_2$ self collisions and $L_0 = 8.7$ cm for $D_2$–He cross collisions). $L_0$ is the **BGK!** mean free path (in cm) in a gas at $T = 1$ eV and density $n = 1\times 10^{14}\,\mathrm{cm}^{-3}$.
For the same conditions as above: $T = 300$ K ($T = 0.026$ eV) and a pressure of 1 Pa ($n_N = 2.414\times 10^{14}\,\mathrm{cm}^{-3}$), one then finds $\lambda_{BGK} \approx 1.6$ cm in a $D_2$ gas.
For $Kn \ll 1$ a neutral fluid model (diffusion approximation, Euler, or Navier Stokes, etc.) is more appropriate, while $Kn \gg 1$ leads to the (linear) free molecular flow regime, in which only the linear collisions terms (collisions of neutrals with surfaces and a given "plasma" host medium) are retained.

### 1.9.1 BGK! approximation

currently under development. Please contact us for details and status.
See fig. 1.12.
Further details can be found in Ref. [**kn:Reiter96**].

The **BGK!** approximation used in the EIRENE code to model neutral–neutral collisions, of both monatomic and multiple species gases, and its implementation via successive approximation of the non-linear Boltzmann equation, is a special case of the classical "test particle method" for solving the Boltzmann equation, as opposed to the "direct Monte Carlo Simulation" method. The former was introduced in rarefied gas dynamics by J.K. Haviland, in 1961, see: J.K. Haviland, "The solution of Two Molecular Flow Problems by the Monte Carlo Method", in: Methods in Computational Physics (Ed. Alder), Vol. 4, 1965, Academic Press. An early review and comparison of different Monte Carlo methods in kinetic theory, also comparing this particular "test particle method" with the **DMCS!** scheme, is given by the report: N.A. Derzko, "Review of Monte Carlo Methods in kinetic Theory", UTIAS Review No. 35 (April 1972) University of Toronto, Institute of aerospace studies, April 1972. (This review also includes a discussion of a third Monte Carlo scheme, the so called "Integral evaluation method" of A. Nordsieck and B.L. Hicks.)
In rarefied gas dynamics problems for fusion reactor design, in particular for the ITER divertor design, the Haviland method, but applied to a Boltzmann equation with a (non-linear) **BGK!** collision integral, was first developed for the EIRENE code around 1995 (see Reiter et al. [**kn:Reiter96**]), and it is routinely applied in computational divertor design studies for ITER since about 2004 by V. Kotov, A.K. Kukushkin, et al.
The method of "successive approximations" employed in EIRENE is illustrated best when using the integral (Fredholm) form of the Boltzmann equation ... to be continued ...

#### 1.9.1.1 A: Self collisions

Consider first the elastic self collisions amongst particle species $A$. The **BGK!** approximation to the Boltzmann collision integral reads:

$$\left.\frac{\partial f_A(\mathbf{v})}{\partial t}\right|_{BGK} = -\nu(f_A(\mathbf{v}) - f^0(\mathbf{v})) \tag{1.92}$$

where

$$f^0(\mathbf{v}) = \frac{n}{(2\pi mT)^{3/2}} \exp\left[-\frac{m(\mathbf{v}-\mathbf{v})^2}{2T}\right] \tag{1.93}$$

is a drifting Maxwellian distribution.
The parameters $n, \mathbf{v}, T$ in $f^0$ are uniquely determined by conservation of particles, momentum and energy in these elastic collisions.
When the collision rate $\nu$ is independent of velocity $v$, as it is the case in the default model described above, then the parameters are simply given by the first velocity moments of $f_A$.
A key feature for these velocity independent collision frequencies is the simple closed form solution for $f_A(\mathbf{v}, t)$ in a homogeneous medium, which translates into a particular simple realisation of this type of collisions in a Monte Carlo procedure. Let $g(\mathbf{v}) = f_A(\mathbf{v}, t = 0)$, then

$$f_A(\mathbf{v}, t) = g(\mathbf{v})e^{-\nu t} + \left(1 - e^{-\nu t}\right) f^0(\mathbf{v}) \tag{1.94}$$

where the parameters $(n, \mathbf{v}, T)$ in $f^0$ can be calculated directly as moments of $g(\mathbf{v})$ (i.e. from the EIRENE default tallies PDEN, VDEN, EDEN.)
When a velocity dependent collision frequency is used, e.g. to capture more than just one continuum limit transport coefficient correctly (or adjust to another Prantl number), then the solution of the initial value problem for the distribution function $f_A$ is not as simple any more, and the parameters $(n, \mathbf{v}, T)$ in $f^0$ become fictitious local parameters related to the five moments of $f_A(\mathbf{v})$ weighted with $\nu(\mathbf{v})$:

$$\int \nu(\mathbf{v}) M_i(\mathbf{v}) f^0(\mathbf{v}) d^3(v) = \int \nu(\mathbf{v}) M_i(\mathbf{v}) f_A(\mathbf{v}) d^3(v) \quad i = 1, \ldots, 5 \tag{1.95}$$

Here $M_i(\mathbf{v})$ are the five collisional invariants: $M_1 = 1$ (particle conservation), $M_{2,3,4} = (v_x, v_y, v_z)$ (momentum conservation), and $M_5 = m/2v^2$ (energy conservation). Storage for the rates on the right hand side of these relations is foreseen on the EIRENE "BGK" default tallies, and the Maxwellian integrals on the left hand side have to evaluated on a case by case basis, depending on the choice of the velocity dependence in $\nu(\mathbf{v})$ made.

#### 1.9.1.2 B: Cross collisions

to be written. . .

#### 1.9.1.3 B1: Cross collisions: electron or photon impact

In case of photon absorption processes, or other processes with no momentum (velocity) transfer, one virtual background species suffices:
For example, in case of resonance (Lyman) line absorption with excitation $H(1s) \rightarrow H(2)$, the velocity distribution of H atoms does not change, and only a photon absorption rate coefficient

(additional to the electron impact ionisation rate coefficient) needs to be provided, rather than a spectral parametrization of the Lyman radiation field, since only the density but not the distribution function of the H atoms is affected by the photons.
From the photon point of view, a virtual parameterized H atom species (e.g. with flow velocity, temperature) is needed, because via Doppler effects the absorption line shape in the photon transport equation is very sensitive to the local atomic gas parameters

#### 1.9.1.4 Implementation, "virtual particle" concept, iteration

Self collisions between those particle species, which are followed by EIRENE (i.e., between any two "test-particle"-species) can be included by using the iteration option NITER > 1 (input block 1), and by adding a proper collision process to the collision kernel (and its related total cross section $\Sigma_{nonlin}$ to the total cross section $\Sigma$):

$$C = C_{lin} + C_{nonlin} \tag{1.96}$$

$$\Sigma = \Sigma_{lin} + \Sigma_{nonlin} \tag{1.97}$$

Whereas $C_{lin}$ and $\Sigma_{lin}$ depend only upon parameters of the background medium ("bulk particles", usually: electrons and ions), the kernels $C_{nonlin}$ and the rate coefficients $\Sigma_{nonlin}$ may also depend upon parameters of the test particles (atoms, molecules, test ions) themselves.
If other non-linearities are already present in the model, such as those due to self-consistent feedback between the neutrals and the plasma state (see above: section 1.8), code-iterations between the test particle and bulk plasma parameters can also be used to simultaneously iterate on the neutral–neutral collision effects. Then NITER=1 suffices, i.e. only one single EIRENE cycle per plasma code step, to relax simultaneously both the neutral–neutral interaction and the neutral-plasma interaction non-linearities.
In order to include such non-linear interactions in a case, a copy of the test-particle species cards (input block 4a, 4b or 4c) is put into the background species specification (input block 5), as a "virtual" species. E.g., if one wishes to include H - H elastic neutral–neutral collisions in a simulation, then a (virtual) species of H particles with a drifting Maxwellian distribution has to be specified as one additional background particle species ("bulk ions with charge state zero", by abuse of language). During the iterative procedure, the parameters of the Maxwellian of this artificial background species (e.g. density, temperature, drift velocities in case of velocity independent collision rates) are those that have been obtained from the corresponding test particle tallies in the previous iteration. One iteration step ends by overwriting these parameters at the end of a cycle with the new values, in the post-processing phase of each iteration cycle (module MODBGK.f) and by preparing the new collision rate coefficients for the next iteration cycle.

### 1.9.2 Direct Simulation (DMCS!) of self-collisions

The **DMCS!** method was implemented in EIRENE in the late eighties of the last century, but then abandoned due to its limitations to physically rather simple cases (e.g. no feedback with the consistent bulk plasma state, the often dominant non-linearity in fusion applications). Currently it is not developed nor supported any more. Contact us for details and status. Further details can be found in Ref. [**kn:Behringer**]

## 1.10 Radiation transport, photon gas simulations

The extensions of EIRENE towards radiative transfer problems, and also the material in this present short description, is largely based on the monograph by Oxenius: [**kn:Oxenius**].
The theory of light and its interaction with matter is one of the best that physics can offer, and can predict virtually every observed phenomenon with great accuracy [Feynman, 1985, QED: The strange Theory of Light and Matter, Princeton University Press, Princeton]. In our simulations of radiation transport with the EIRENE code we assume a geometric optics model (i.e., essentially, "particle theory of light"), which is sufficient for all our purposes. Light is emitted, scattered, reflected and absorbed, but moves on straight lines between these events. I.e., a continuously varying index of refraction is not allowed, and we ignore most properties of light that depend on a wave or quantum model for their explanation (e.g. diffraction, or interference). However, the linear wave-optics phenomenon of polarization and the linear quantum optics phenomenon of fluorescence [absorption of a certain wavelength (line), and re-emission at another wavelength (line)] can conveniently be added to our linear geometric optics model.
We are concerned with the kinetic theory of interacting particles and photons, where the behaviour of each species is governed by a kinetic equation: the kinetic equation for photons being nothing else but the well known equation of radiation transfer, i.e., the "strangely normalized photon-Boltzmann equation". In contrast to plasma radiation (e.g., Bremsstrahlung) the radiation processes studied here are due to single, uncorrelated particles such that emission and absorption of the plasma is obtained by simple summation of all contributions of all individual particles. We therefore consider all particles as uncorrelated and in well defined one-particle states.
For continuum radiation (bound-free or free-free) the frequency of the photon in the atoms rest frame may be replaced by the frequency in the lab. frame, because all cross sections are slowly varying functions of frequency. By contrast, for line-radiation due to bound-bound transitions, the Doppler effect must be taken into account in all quantities that vary rapidly as function of frequency, such as cross sections, line-profile, or redistribution functions
The key atomic parameters for radiation transport are the Einstein (or Einstein-Milne) coefficients $A_{mn}$, $B_{mn}$ and $B_{nm}$ of spontaneous emission, induced (stimulated) emission and absorption, respectively. We will discuss them here first for a gas of stationary atoms and include Doppler- and other line broadening effects later.
For a thermal radiation field (temperature T) in a gas of atoms with density $n_1$ (lower state) and $n_2$ (upper state) the rate equation expressing balance between emission and absorption reads:

$$n_1 B_{12} R_\nu(T) = n_2 [A_{21} + B_{21} R_\nu(T)]$$

.
$R_\nu(T)$ is a specific function characterizing the wavelength and temperature dependence of the thermal radiation field (i.e., the Planck-function). For example the specific intensity $I_\nu$ (energy-/area/time/solid-angle/frequency-interval) is often used, or the (angle)-integrated intensity $J_\nu = \int I_\nu d\Omega$, or the specific energy density $u_\nu = 1/c J_\nu$ (energy/volume/frequency-interval), etc. Also the wavelength dependence may be given either using frequency $\nu$ (Hz), $\omega$ (rad s$^{-1}$), wavelength $\lambda$ or energy $E = h\nu = \hbar\omega$.
Depending upon all these choices the numerical factors in the definitions of the $B_{nm}$ coefficients (sometimes even in the $A_{nm}$) differ in the literature. For our purpose (Monte Carlo solution of the full radiation transfer equation) we need to define monochromatic mean free paths for photons, rates and rate coefficients and cross sections for the individual processes.

In order to take advantage of the analogy between radiation and particle transport equations as much as possible, we will choose the numerical constants in the Einstein coefficients such that these parameters have the same meanings and even the same units in both cases of radiation and particle transport. I.e., we will use the "absorption coefficient" (better: "extinction coefficient", if scattering is allowed) $\alpha_\nu = n\sigma \equiv \Sigma_t$, (inverse monochromatic mean free path), which directly corresponds to the "total macroscopic cross section" (dimension: 1/length) defined earlier for particle transport problems.

### 1.10.1 Line shape options

Emission of photons from a particular line is simulated by sampling firstly the location of emission, and secondly the frequency and direction of emission. Spatial distribution of photon emission is treated in EIRENE in exactly the same way as for any other particle type (neutrals, ions), and actually using the same modules of code for that. Emission from the transition $i \rightarrow k$ from an upper state $i = nl$ of an atom is treated, by abuse of language, as "volume recombination source": The position is sampled from the spatial distribution $A_{ik} P^{nl}(\vec{r})$, which, because the Einstein emission coefficient $A_{ik}$ is constant, is the same as the spatial distribution $P^{nl}(\vec{r})$ of upper state particle density. ... Given the birth point $\vec{r}$ of the photon, and hence all local parameters of the host medium at this point, the frequency $\nu$ (in fact: the energy $E = h\nu$) is sampled from the normalized line shape function $\Phi(\nu)$, ... The following line shape options are currently available:

ISHAPE = 0 $\delta$-profile: mono-energetic emission (not yet available for absorption)

ISHAPE = 1 pure Doppler profile

ISHAPE = 2 Lorentz profile, truncated at zero

ISHAPE = 3 Doppler convoluted Lorentz profile, i.e. Voight profile

ISHAPE = 4 LORVDW: convoluted Lorentz and van der Waals pressure broadened profile

ISHAPE = 5 Doppler broadened LORVDW profile

ISHAPE = 6 normal Zeeman triplet, $\delta$-profile in each component

ISHAPE = 7 Doppler broadened Zeeman-$\delta$-profile in each component

ISHAPE = 8 Doppler broadened Zeeman–Stark-profile, 10 Lorentzian model, Rosato 2006

## 1.11 Charged Particle Transport

EIRENE also follows charged particles since EIRENE$_{1987}$. The particle tracing is carried out in Subroutine FOLION. We refer to this "default" ion transport model as the *"minimal trace ion model"*, as opposed to refinements added only later, starting with Seebacher et al., (2012) [**kn:Seebacher**]. The differences of the module FOLION to the module FOLNEUT for neutral particles (and photons) are:

- Motion is not along straight lines between events, but instead determined by drifts and forces due to the electro-magnetic fields.
- Time stepping is not of "Poisson type" (exponentially distributed time steps), but with constant increments $\Delta t$, because the equation of motion has to be solved numerically, and this introduces already a discrete time-stepping anyway.
- after each time-step a Coulomb collision kernel is modelled, i.e., a diffusive step in velocity space.
- after each time-step a diffusion-step is modelled, i.e., a diffusive step in physical space (not yet fully written).
- ionized test particles which hit a surface see an electrostatic sheath potential, by which they are either reflected, slowed down or accelerated, depending on the sign of their charge relative to the sheath potential.

### 1.11.1 Orbit integration

Orbit integration in a magnetized plasma is carried out to various orders of the small parameter $\delta = \rho/L$, ($\rho$: thermal gyro-radius, $L$ spatial length scale, e.g. gradient length of B-field, etc.)

#### 1.11.1.1 Particles following the B-field

Simple, analytical, orbit integration, zeroth order in $\delta$.

**A: no electric field, no drifts**

To zeroth order in the smallness parameter $\delta = \rho/L$, i.e. for $\rho \simeq 0$, charged particle orbits are the magnetic field lines themselves. The magnetic field unit vector $\mathbf{b} = \mathbf{B}/|\mathbf{B}|$ is provided as input vector in each cell of the computational grid, see input block 5 (section 2.5). By default the magnetic field vector is taken to be constant within a grid cell, as are also the plasma (background) input tallies.

With the "FEM" interpolation option (EIRENE$_{2006}$ and younger) iso-parametric elements can be used for interpolating input tallies in various grid options, and hence providing a magnetic field $\mathbf{B}$ which continuously varies inside the grid cell (and then also providing non-zero derivatives such as grad—$\mathbf{B}$— or B field curvature).

In 2D cases the interpolation scheme currently ignores the third coordinate, even if it is curvilinear (e.g., NLTRA option, toroidal effects in a 2D setting). As a consequence toroidal curvature effects are not yet properly seen by charged particles in that case, and the corresponding toroidal curvature drift had been added separately in [**kn:Seebacher**].

Default is $\mathbf{b} = (0., 0., 1.)$ in each cell. The velocity vector of each test-ion followed by EIRENE is projected onto $\mathbf{b}$ in each grid cell along the trajectory. The velocity used for pushing the particle along its trajectory is only the parallel component $\mathbf{v}_{\|} = (\mathbf{v} \cdot \mathbf{b})\mathbf{b}$.

**B: E × B drift only**
Only the $\mathbf{E} \times \mathbf{B}$ drift is included. This is still zeroth order in $\delta$. By modifying the electric field in each cell into one effective $\mathbf{E}^*$ field, some features of guiding centre drifts (namely: the grad B drift) can be already captured at this level.

#### 1.11.1.2 Guiding centre approximation

To first order in $\delta$ a number of so called guiding centre drifts arise. Their inclusion into EIRENE (module FOLION) is a later code extension, and not fully operational yet. Once this option is fully implemented, guiding centre currents will be derivable from EIRENE runs as volume tallies. These guiding centre currents, supplemented by the (divergence free) magnetisation currents, are one important component of the plasma currents.

### 1.11.2 Coulomb Collision Models

In this section conventional notation for masses of colliding charged test particles labeled $a$ and field (background) particles $b$ is used: $m_a, m_b$ for the masses, for atomic mass numbers $\mu_a, \mu_b$ and for reduced masses $\mu_{a,b}$ are used. Charge states are $Z_a, Z_b$. Particles of type $b$ are background particles, with a given velocity distribution $f(\vec{v}_b)$, whereas particles of type $a$ are individual test particles with a specific velocity $\vec{v}_a$.
Four (velocity dependent) collision relaxation rates (inverse time) arise: The slowing down rate $\nu_s^{a,b}(v_a)$, the deflection (or transverse diffusion) rate $\nu_\perp^{a,b}(v_a)$, dispersion (or parallel diffusion) $\nu_\|^{a,b}(v_a)$ and the energy loss rate $\nu_\varepsilon^{a,b}(v_a)$
Due to energy conservation in the binary Coulomb collisions one has [**kn:NRL**]:

$$\nu_\varepsilon^{a,b}(v_a) = 2\nu_s^{a,b}(v_a) - \nu_\perp^{a,b}(v_a) - \nu_\|^{a,b}(v_a). \tag{1.98}$$

I.e.: the energy transfer rate $\nu_\varepsilon^{a,b}$ is given via the "slowing down rate" $\nu_s^{a,b}$, the transverse diffusion rate $\nu_\perp^b$ and the parallel diffusion rate $\nu_\|^b$, for any velocity $\vec{v}_a$ and any distribution $f_b(\vec{v}_b)$.
Explicit results when $f(\vec{v}_b)$ is a Maxwellian distribution are given in most textbooks or lecture notes on plasma physics since many decades. In general: unless otherwise stated, notations here are as in [**kn:NRL**]. The resulting rates (or: rate coefficients, if divided by target density $n_b$) have the same functional dependencies on test particle velocity (or energy) and background temperature as the "Beam-Maxwellian rates" (or rate coefficients) of format H.3, H.4, H.5 for in-elastic collisions described in section 4 (atomic and molecular data input, section 2.4)

#### 1.11.2.1 Simple Coulomb Collision Models

Strongly simplified Coulomb collision models are usually sufficient for molecular ions, because of the molecular (radical) ion's typically very short lifetime until further fragmentation (dissociative excitation, dissociative recombination, etc.) to atomic products. In order to roughly account for thermalization effects and background plasma cooling or momentum transfer, the energy, parallel

and the perpendicular velocity of such ions can be modified "continuously" during particle orbit integration, hence approximating the velocity space diffusion (Fokker Planck equation) by mean values of energy and momentum of the test particle. There are two classes of simple collision models: deterministic and stochastic.

### A: Deterministic relaxation in velocity space

The energy and/or velocity of charged test particles are modified in a fully deterministic way along the particle trajectory. I.e.: along with orbit integration also energy, or certain velocity components are evolved according to ordinary (non-stochastic) differential equations, using appropriate relaxation times.

#### A1: Only energy relaxation

The simplest Coulomb Collision model is an averaged **BGK!**-type relaxation of energy of test particles towards the mean energy of the Maxwellian background ions. In the earliest versions of the EIRENE code (mid eighties) it was originally based upon the hydrocarbon ion equilibration model given by Langer, [**kn:Langer**] and it is a slight generalization thereof. The equation for the energy $\varepsilon_a$ of a test-ion $a$ in a field of background ions $b$ (field particles) with temperature $T_b$ is given as:

$$\frac{d\varepsilon_a}{dt} = -\nu_\varepsilon^{a,b} \cdot \varepsilon_a \quad , \quad \varepsilon_a(t=0) = \varepsilon_a^0 \tag{1.99}$$

In general this is a non-linear equation because the energy exchange collision rate (energy transfer rate) $\nu_\varepsilon^{a,b}$ depends on $\varepsilon_a$ itself. The full expression for $\nu_\varepsilon^{a,b}(v_a, T_b)$ is given below, it involves the Maxwell integral $\Psi$ and its derivative $\Psi'$, see below.

The *limit of small test particle velocities* (as compared to thermal velocities of the field particles $b$), i.e.:

$$v_a \ll \sqrt{(2T_b/m_b)}, \quad \text{i.e.} \quad x^{a/b} = \mu_b v_a^2/2kT_b \ll 1, \tag{1.100}$$

as often particularly appropriate for molecular ions $a$, In this limit one has for the basic "Spitzer rates" (loc. cit.):

$$\nu_s^{a,b} = n_b Z_a^2 Z_b^2 \lambda_{a,b} \cdot 6.8 \times 10^{-8} \frac{\mu_b^{1/2}}{\mu_a} \left(1 + \frac{\mu_b}{\mu_a}\right) T_b^{-3/2} \tag{1.101}$$

$$\nu_\perp^{a,b} = n_b Z_a^2 Z_b^2 \lambda_{a,b} \cdot 1.4 \times 10^{-7} \frac{\mu_b^{1/2}}{\mu_a} T_b^{-1/2} \varepsilon^{-1} \tag{1.102}$$

$$\nu_\parallel^{a,b} = n_b Z_a^2 Z_b^2 \lambda_{a,b} \cdot 6.8 \times 10^{-8} \frac{\mu_b^{1/2}}{\mu_a} T_b^{-1/2} \varepsilon^{-1} \tag{1.103}$$

and hence, using Equation (1.98):

$$\nu_\varepsilon^{a,b} = n_b Z_a^2 Z_b^2 \lambda_{a,b} \cdot 6.8 \times 10^{-8} \frac{\mu_b^{1/2}}{\mu_a} \frac{1}{T_b^{1/2}} \left[2\frac{1}{T_b}\left(1 + \frac{\mu_b}{\mu_a}\right) - 2\frac{1}{\varepsilon_a} - \frac{1}{\varepsilon_a}\right] \tag{1.104}$$

$$= \nu_\varepsilon^c + \nu_\varepsilon^v \tag{1.105}$$

with the "constant" and "variable" contribution $\nu_\varepsilon^c$ and $\nu_\varepsilon^v$, respectively, defined as

$$\nu_\varepsilon^c = 2n_b Z_a^2 Z_b^2 \lambda_{a,b} \cdot 6.8 \times 10^{-8} \frac{\mu_b^{1/2}}{\mu_a} \frac{1}{T_b^{3/2}} \left[1 + \frac{\mu_b}{\mu_a}\right] = \tilde{\nu}_\varepsilon \left[1 + \frac{\mu_b}{\mu_a}\right] \tag{1.106}$$

$$\nu_\varepsilon^v = 2n_b Z_a^2 Z_b^2 \lambda_{a,b} \cdot 6.8 \times 10^{-8} \frac{\mu_b^{1/2}}{\mu_a} \frac{1}{T_b^{3/2}} \left[-\frac{3T_b/2}{\varepsilon_a}\right] = \tilde{\nu}_\varepsilon \left[-\frac{3T_b/2}{\varepsilon_a}\right]. \tag{1.107}$$

Using this decomposition of $\nu_\varepsilon^{a,b}$ in equation (1.99) we find:

$$\frac{d\varepsilon_a}{dt} = -\tilde{\nu}_\varepsilon \left[1 + \frac{\mu_b}{\mu_a}\right] \cdot \varepsilon_a + \tilde{\nu}_\varepsilon \cdot \frac{3T_b}{2} \tag{1.108}$$

Hence, in this small test particle velocity limit of $\varepsilon_a \ll 3/2T_b$, and introducing the mass correction factor $\delta_{a,b} = 1 + \mu_b/\mu_a$ the solution of this (now: linear) ordinary differential equation exists in closed form:

$$\varepsilon_a(t) = \varepsilon_a^0 \cdot \exp(-\tilde{\nu}_\varepsilon \delta_{a,b} t) + \frac{3T_b}{2\delta_{a,b}} \left[1 - \exp(-\tilde{\nu}_\varepsilon \delta_{a,b} t)\right] \tag{1.109}$$

**Example: hydrocarbon ions [kn:Langer]**
With $\mu_b = 1$ for protons, $\mu_a = 16$ for $CH_4^+$, the Coulomb logarithm $\lambda_{a,b} \approx 10$, hence $2(\lambda_{a,b}/\mu_a)\delta_{a,b} \approx 1.3$ and

$$\nu_\varepsilon = \tilde{\nu}_\varepsilon \delta_{a,b} \approx 8.8 \times 10^{-8}\, n_b\, T_b^{-3/2}$$

($n_b, T_b$ in $\mathrm{cm}^{-3}$ and eV, resp.) which is the solution (11) given in [**kn:Langer**] for thermalization of hydrocarbon ions in a Maxwellian hydrogen plasma. The relaxation is towards an energy $\varepsilon_p = 3/2T_b/(1 + 1/16)$, which is (very slightly) lower only than the mean energy of the target Maxwellian, due to the particular velocity dependence in the energy relaxation rate in this small test particle velocity limit.
And this is also exactly the approximation which was also used in EIRENE since about 1987 for all test ions traveling in a bath of background ions.
To be done: in case of drifting Maxwellian background: first transform into plasma frame, such that stationary Maxw. background, then relax test particle energy in this frame, then transform back... To be done: isotropization: currently the ratio $v_{\perp B}$ to $v_{\parallel B}$ relative to the B-field is preserved.

The *full expression for the energy relaxation frequency* $\nu_\varepsilon^{a,b}$, without the approximation of slow particles $a$, i.e. for all values of $x^{a/b}$ is:

$$\nu_\varepsilon^{a,b}(x^{a/b}) = 2\left[\mu_a/\mu_b \Psi(x^{a/b}) - \Psi'(x^{a/b})\right] \nu_0^{a/b} \tag{1.110}$$

where $x^{a/b}$ has been defined above, relation (1.100)

$$\nu_0^{a,b} = n_b Z_a^2 Z_b^2 \lambda_{a,b} \cdot 4\pi/(\mu_a^2\, v_a^3) \tag{1.111}$$

and the Maxwell integral $\Psi(x)$:

$$\Psi(x) = \frac{2}{\sqrt{\pi}} \int_0^x dy\, e^{-y} \sqrt{y} \qquad = \mathrm{erf}(\sqrt{x}) - \frac{2}{\sqrt{\pi}} e^{-x} \sqrt{x} \tag{1.112}$$

$$\Psi'(x) = d\Psi/dx \qquad = \frac{2}{\sqrt{\pi}} e^{-x} \sqrt{x} \tag{1.113}$$

With this more general expression for the collision frequency Equation (1.99) can then only be integrated numerically along the particle trajectory. However, other features of the collision process (velocity spreading, isotropization) then should be taken into account as well in order to ensure proper equilibration of energy $\varepsilon_a$ of test particles to $3/2T_b$ of the fixed (stationary Maxwellian) background, as well as momentum conservation. Without such further velocity space effects a test-particle with energy $\varepsilon_a$ corresponds to the non-equilibrium situation of a plane parallel (mono-energetic) flux, and energy exchange of this non-equilibrium flux with a Maxwellian background vanishes at a critical energy $\varepsilon'$ distinct from from $3/2T_b$. [**kn:Trubnikov**].

**A2: BGK! Coulomb-Collision operator**

(Diploma Thesis Felix Reimold. . . )

#### 1.11.2.2 The Fokker-Planck Collision Model

**A:**

A gyro averaged Fokker-Planck Coulomb Collision Model, which was originally developed for a stand alone trace impurity ion transport code DORIS [**kn:Doris**, **kn:Reiser98**]. It was later accommodated as collision kernel in EIRENE. Distinct from various other drift-kinetic collision operators published already since the early nineties of the last century its range of applicability is rather limited. In its present version this collision operator ignores the difference between "real particle position" and the guiding center, hence, e.g. (classical) diffusion terms due to finite Larmor radius effects are not taken into account. The averaging over gyro angle limits this operator to distributions of relative velocities which are symmetric around the B-field line, i.e. essentially a 1D, parallel to B, situation, (but it allowed for distributions which may deviate from Maxwellians in this B-field direction, e.g. to accommodate parallel to B thermal force effects). See also: Thesis J. Seebacher, Univ. Innsbruck, 2009, and [**kn:Seebacher**].

**B: binary Monte Carlo collision procedure**

A procedure similar to the "Abe, Takizuka Coulomb collision algorithm" frequently used in PIC codes [**kn:Takizuka**] is employed. This procedure consists in random sampling a bulk (background) collision partner, and then to carry out an effective binary collision between these two particles. It hence can be viewed as Monte Carlo "randomization" of the algorithm A, in which the integration over background distribution is carried out deterministically.

## 1.12 Parallelization of EIRENE Code

EIRENE is often coupled iteratively to finite volume or finite element **CFD!** codes, which provide the host medium (plasma background). Then EIRENE is used as kinetic neutral particle transport module, providing, often in iterative mode, the reaction part in the overall "diffusion-advection-reaction problems. Non-linearly coupled fusion edge plasma flow simulations fall into that category.
In contrast to the **CFD!** codes, which can be parallelized using domain decomposition, EIRENE uses a particle partitioning technique, which is usually most suitable for Monte Carlo methods. Here the stochastic Monte Carlo trajectories are distributed on different processes, whereas the domain is the same on all processes. If there would be no stratified source sampling also typically involved, then this particle partitioning would provide a good load balancing automatically. Most 3D edge plasma applications using the EMC3-EIRENE hybrid Lagrangian **CFD!**/Lagrangian kinetic Monte Carlo code are run in this ("embarrassingly parallel") single stratum mode of operation.
However, the 2D hybrid finite-volume/Lagrangian Monte Carlo code B2-EIRENE is heavily resorting on "stratified sampling". This means the primary sources of neutral particles are split by their physical or computational origin into "sub-strata", and results are then linearly superimposed finally. This is, amongst others, a variance reducing technique, but it also leads to load balancing issues in the Monte Carlo part, even with particle partitioning.
Different strategies have been implemented in the past for assigning strata to processes:
problem specific matrix: PROCFORSTRA(NSTRA,0:NPRES-1), to be filled in startup routine pedist.f
. . . to be written . . .

## 1.13 EIRENE flow charts

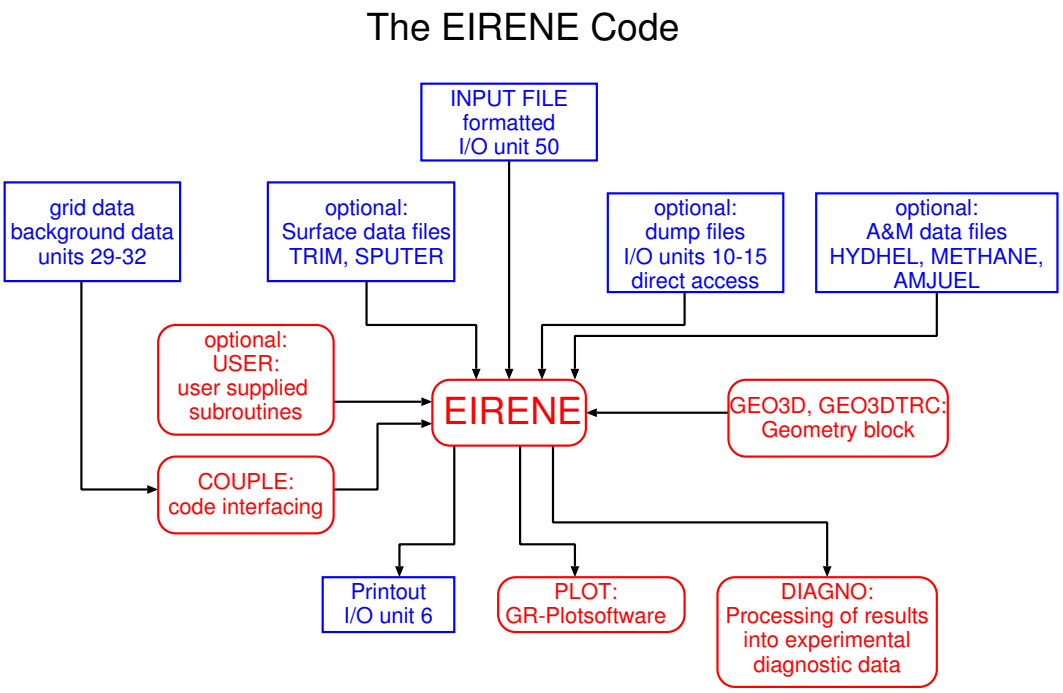



Figure 1.7: Structure: EIRENE code.

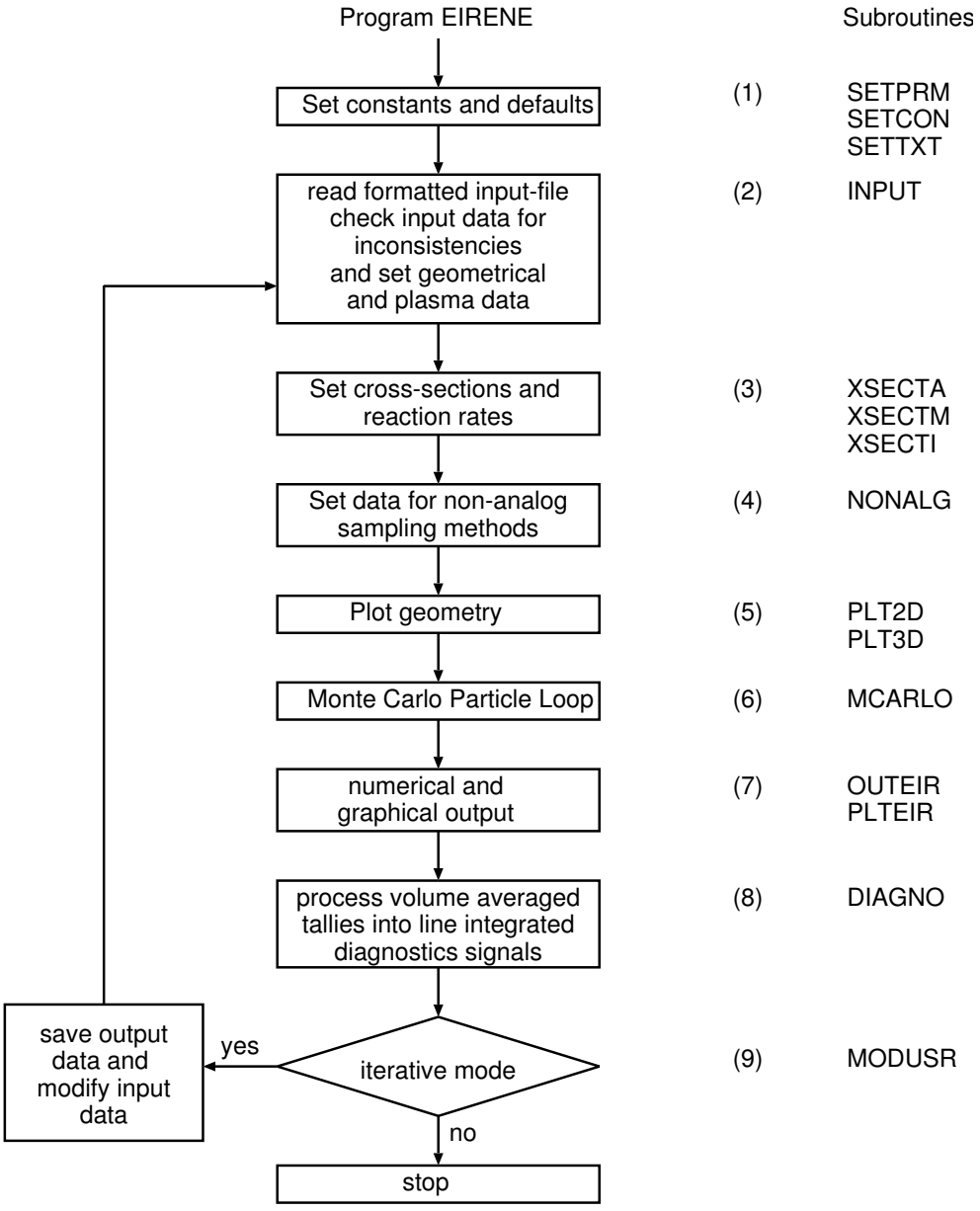



Figure 1.8: Program EIRENE.

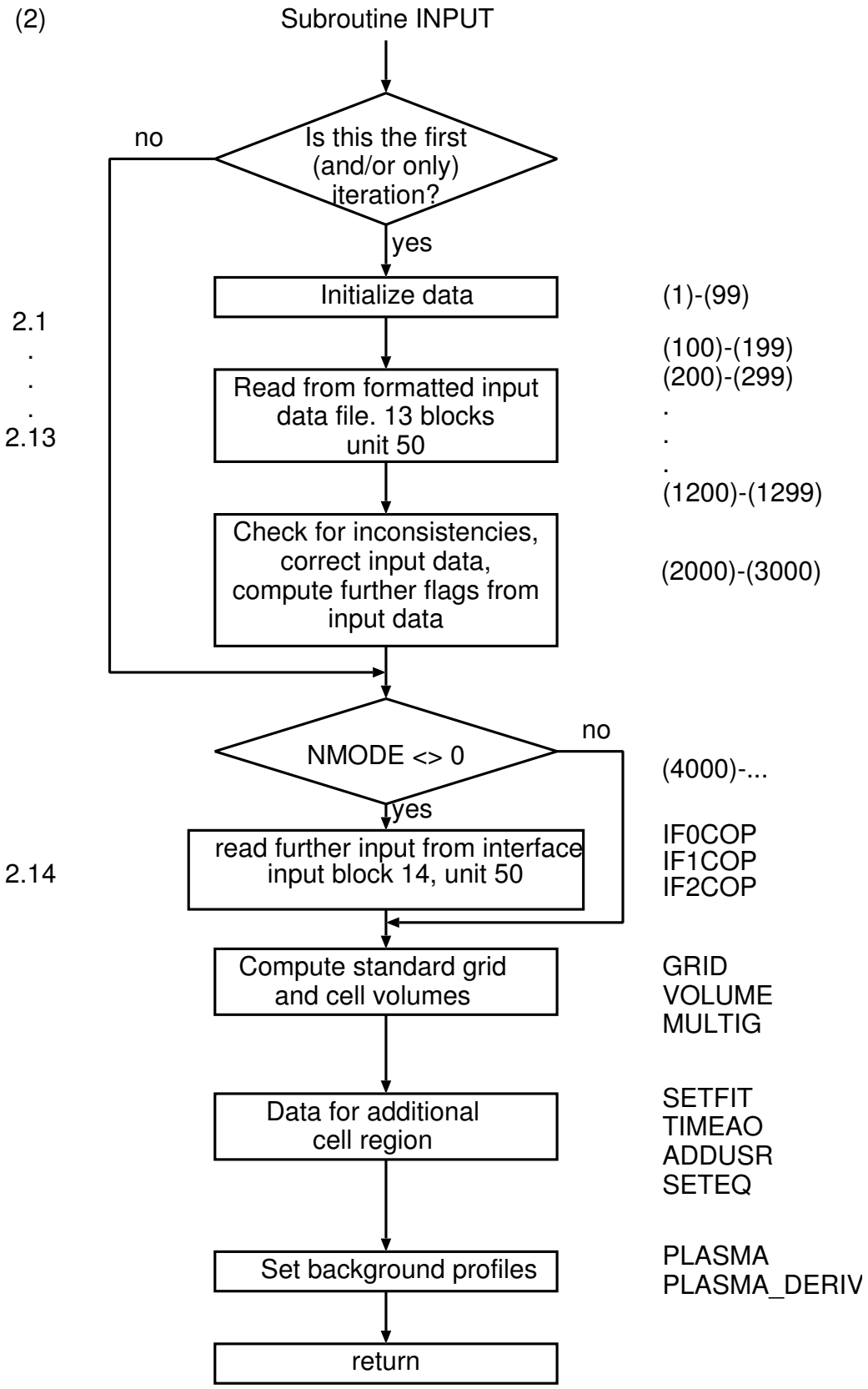



Figure 1.9: Subroutine INPUT.

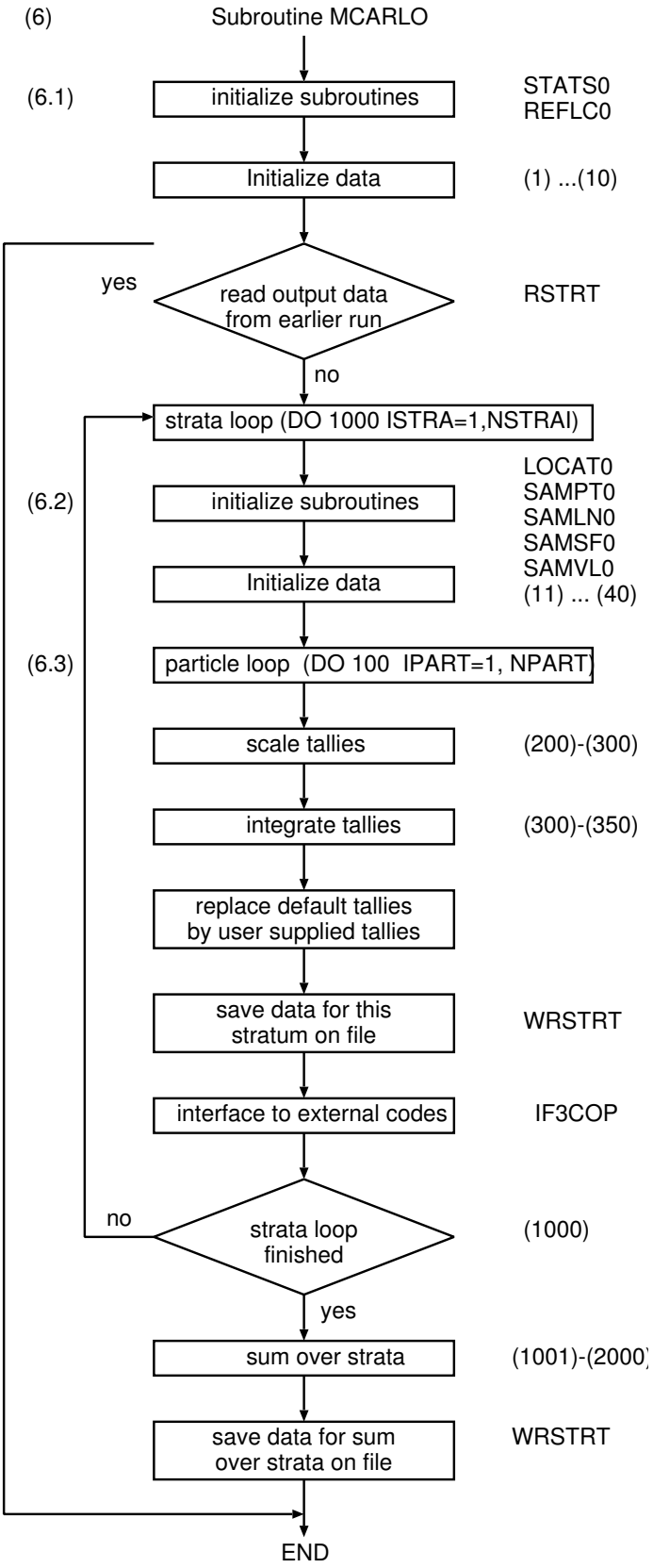



Figure 1.10: Subroutine MCARLO.

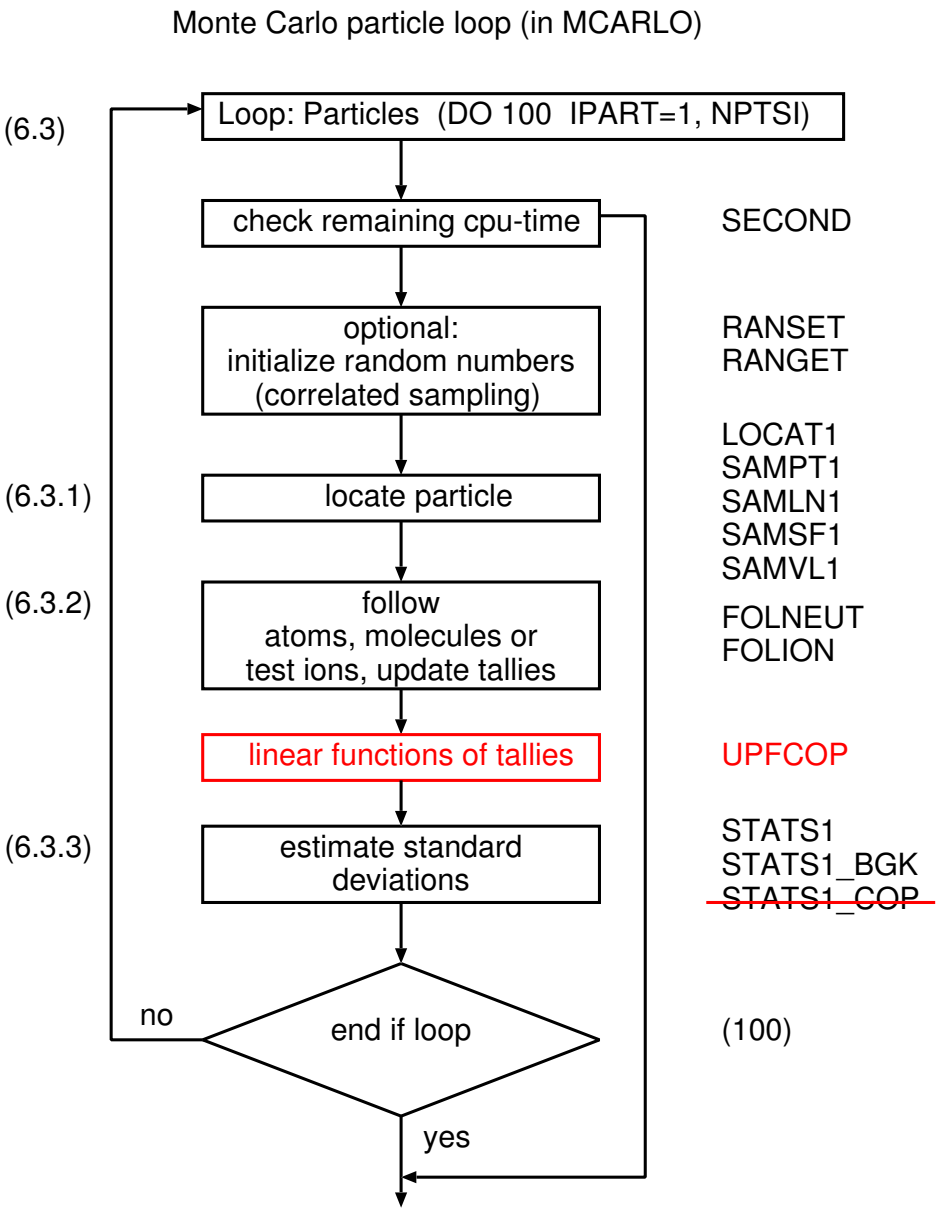



Figure 1.11: Particle loop in subr. MCARLO. Default volume averaged tallies are updated, "on the fly" *(per event)* in routines UPDATE, problem specific tallies for code-code interfacing (COPV(...), e.g. B2-EIRENE) in UPDCOP, specific tallies for non-linear **BGK!** operators (BGKV(...)) in UPDBGK. Corresponding variances are updated *(per history)* after completion of each trajectory in STATS1, STATS1_COP, STATS1_BGK. Addition/revision in 2012: Updating of linear functions of default tallies, *per history*, in UPFCOP, e.g. for code coupling tally COPV. This makes redundant the specific statistical variance evaluation in STATS1_COP.

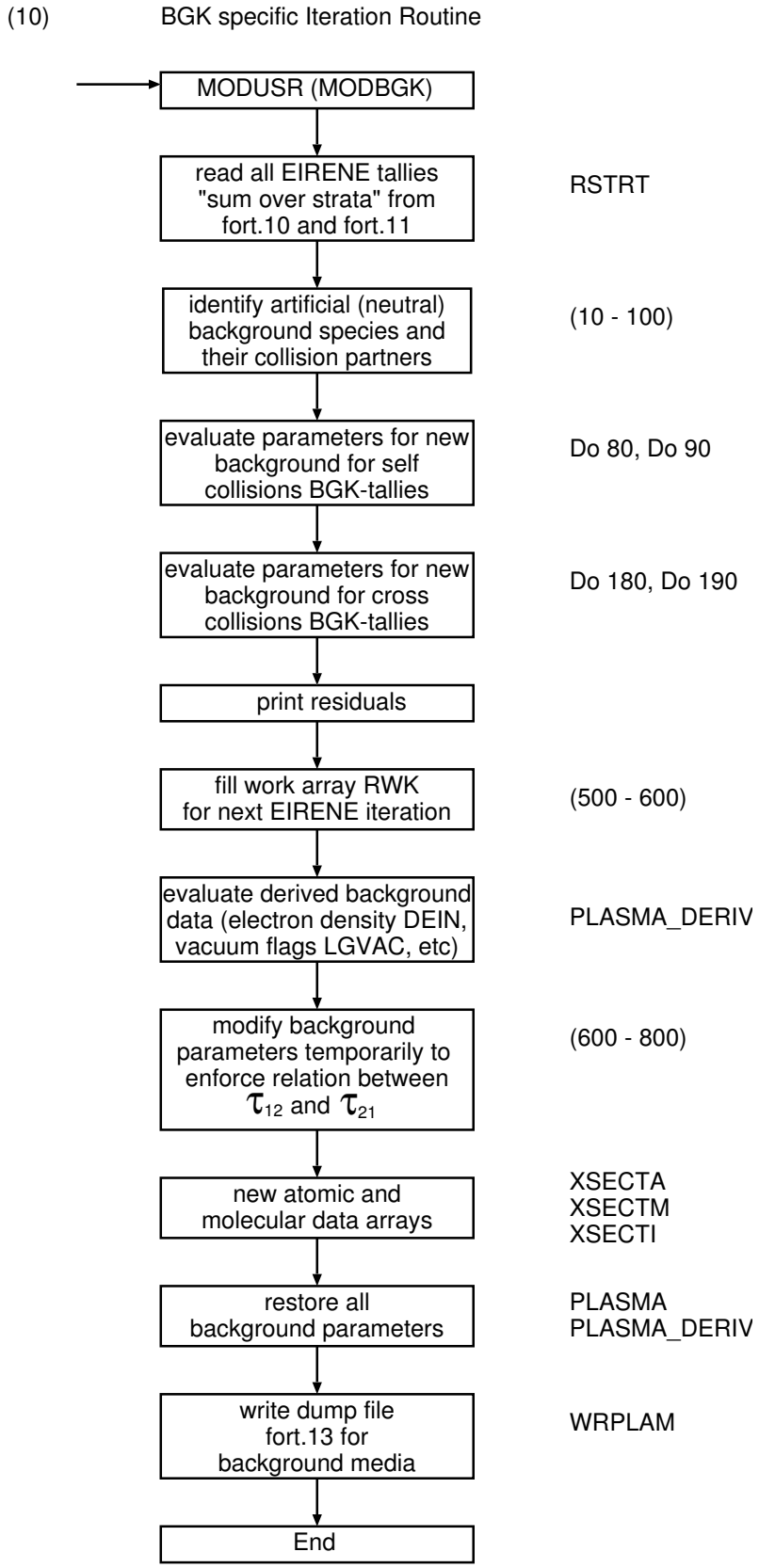



Figure 1.12: Iterative mode for neutral–neutral interactions.

# Chapter 2

# Description of EIRENE input file

**General Remarks**

Reading of all input data is initialized in subroutine INPUT. Starting with EIRENE version EIRENE_JSON_input two I/O options are available: the traditional formatted input or the more modern JSON input. In order to distinguish which type of input is available the code reads the first line of input from I/O channel IUNIN. Depending on the first character found the appropriate read procedure is carried out.
First character

= $*$ means traditional formatted input is used.

= { means JSON input is expected.

Independent of the input format used the code will produce a corresponding JSON file. Thus the code works as an input converter as it is planned to completely move to JSON format over time.

**Formatted Input**

The formatted input is read from unit IUNIN via subroutine READ_FIXFORM
There are 14 "blocks", each one starting with a card

$***$ *n . TEXT*

where n is the number of the input block, n = 1, 2, . . . , 14. Some blocks are subdivided into so called "sub-blocks", each one starting with a card

$**$ *TEXT* .

Each input block is described in one of the following 14 sections 2.1 to 2.14:

2.1 *Input data for operating mode*
2.2 *Input data for standard mesh*
2.3*A Input data for "Non-default Standard Surfaces"*
2.3*B Input data for "Additional Surfaces"*
2.4 *Input data for species specification and atomic physics module*
2.5 *Input data for plasma background*
2.6 *Input data for surface interaction models*
2.7 *Input data for initial distribution of test particles*
2.8 *Additional data for some specific mesh zones*
2.9 *Data for statistics and non-analog methods*
2.10 *Data for additional volumetric and surface averaged tallies*
2.11 *Data for numerical and graphical output*

2.12 *Data for plasma diagnostic module DIAGNO*
2.13 *Data for nonlinear and time-dependent mode*
2.14 *Data for interfacing with external codes*

This last block 14 may be read in from the user supplied part (with any format specified there) from this same unit IUNIN, (but equally well from any other unit specified there), e.g., from the subroutine INFCOP for interfacing with other codes.

**Input format**

The following conventions are used:

- Standard Fortran naming conventions, i.e., variable names starting with the letters I, J, . . . , N are Integer, all other variables are Real.

In addition to these standard convention the following rules are employed:

- Character strings start with the letter C. . . or with T. . .
- Logical variables start with NL. . . , LG. . . or with TRC. . .

The format for a card containing only real variables: (6E12.4)
The format for a card containing only integer variables: (12I6)
The format for a card containing only logical variables: (12(5L1,1X))
The format for a card containing text: (A72)
The format for a card containing K (K < 6) integer variables first and then up to 6-K real variables next: (K(I6,6x),(6-K)E12.4)
An input card starting with '*' is recognized as a comment line, i.e., it does not contain input data. These comment lines should not start with '**' or '***', because in this latter case EIRENE would assume that the input for the current block is completed. It would expect the first card of the following block (or '* comment. . . ' lines belonging to this next block) as next card. Any data not read from the input file, before the following block is read, are set to default values.
An arbitrary number of such comment lines may be included in the input file, but each block of comment lines must always be put at the beginning of a block or sub-block. I.e., comment lines are only identified by EIRENE if preceded by a card starting with '**' or '***' indicating a new sub-block or block respectively.

Example (see section 2.7 below):

```
*** 7. Data for initial distribution of test particles
* this is a comment
* there are NSTRAI=2 strata in this run
* this is a comment
 2
** Data for first stratum
* this is a surface recycling source of helium ions, IPLS=2
* this is a comment
FFFF
   .
   .
```

```
   .
   .
** Data for second stratum
* this is a hydrogen ion volume re-combination source, IPLS=1
FFFF
   .
   .
   .
*** 8. Data for some specific zones
   .
   .
   .
   .
```

and so on.

In very few cases formats other than these are used. They are then explicitly mentioned in this manual and in case of doubt the user should check in subroutine READ_FIXFORM.

**JSON input**

JSON is an acronym for JavaScript Object Notation [**kn:Json**] which is an open standard file format. It can represent four primitive types and two structured types. The primitive types are strings, numbers, boolean and null. The structured types are objects and arrays.
An object is a

name : value

pair where "name" is a string and "value" can be any of the primitive or structured types. A JSON input is easily readable and can also be viewed within browsers or online editors.
Libraries for JSON have been implemented for many programming languages including FORTRAN. An implementation for an object-oriented FORTRAN library can be found on github [**kn:Williams**].
Unlike the formatted I/O reading the JSON input file does not use the I/O unit IUNIN but expects the input file to have a specific filename. The name **eirene.input.json** has been chosen. Therefore this file should be linked to unit IUNIN initially.
Similar to the formatted input the JSON input consists of 15 blocks. A zeroth block was introduced to hold the input header and comments on the specified case. Each block is its own JSON object. Usually all blocks are placed into the main input file **eirene.input.json** but can be put into separat files as well. Then the main input file only gets a skeleton object of the block with an indicator where to look for the corresponding input. The names of the blocks follow the naming in the formatted input and describe the intended subject of the data in the block. An entry "MANUAL" in each block refers to the corresponding section in this manual. Sub-blocks are grouped into their own objects with appropriate names. The blocks are named:

*HEADER*

*GENERAL_DATA*

*STANDARD_MESH*
- *RADIAL_GRID*
- *POLOIDAL_GRID*
- *TOROIDAL_GRID*
- *MESH_MULTIPLICATION*
- *ADD_CELLS*

*NONDEF_STD_SURFACES*

*ADDITIONAL_SURFACES*

*PHYSICS_MODEL*
- *SPECIES_SPEC*
  - *REACTIONS*
  - *ATOMS*
  - *MOLECULES*
  - *TEST_IONS*
  - *PHOTONS*
- *BACKGROUND*
  - *BULK_IONS*
  - *PLASMA*

*REFLECTION_MODELS*

*SOURCES*

*SPECIFIC_ZONES*

*STATISTICS*

*ADDITIONAL_TALLIES*

*OUTPUT*

*DIAGNOSTICS*

*TIMEDEPENDENT_MODE*

*INTERFACING*

The input of interfacing blocks has been converted to JSON format as well. Any other data read by user routines is placed into a file **user_data.input** by the converter and is read in the same way as with the formatted input.
The names for the variables are almost everywhere identical to those given in this manual. Deviations are indicated in the corresponding block.
Variables and blocks can be omitted from the input file if they are not required by the simulation. Omitted variables are set to 0 or .false. in case of boolean. The converter however will always produce the complete JSON files. The two blocks *ADDITIONAL_SURFACES* and *PHYSICS_MODEL* are always placed into and read from separate files named **eirene_add_surfaces.input.json** and **eirene_physics_model.input.json**.

**Units**
cgs units are used, with two exceptions:

All **temperatures and energies** are in eV.
All **particle fluxes** are in Ampere (i.e. $1.6022 \times 10^{-19}$ #/s), even neutral particle fluxes. Since energies are given in eV units, a particle flux × energy is already in watt.
E.g., to convert a **gas flow rate** $Q$ (= "mass-flow rate" at temperature $T_0$), which is a quantity given with dimension pressure times volume divided by time, to a particle flux (source strength, given in 1/ time), one utilizes the universal gas law $P \times V = N \times k_B T$ to convert from pressure times volume (typically at standard temperature $T_0 = 273.15$ K) to the number $N$ of particles. For example $x$ **SLPM!** (**SLPM!**) first divide $x$ by 22.4 (then: mol min$^{-1}$, at "standard conditions" = normal pressure 1013.25 mbar and normal temperature $T_0$). Next multiply by Avogadro's no. $N_A = 6.0221 \times 10^{23}$, (i.e., then particles min$^{-1}$), then divide by 60 (then: particles s$^{-1}$), and finally multiply by the elementary charge $1.6022 \times 10^{-19}$ (then amp, as used for particle fluxes in EIRENE). Gas (mass-) flow rates given in other units, e.g. Torr m$^3$ s$^{-1}$ are converted correspondingly.
Another frequently used dimensional quantity is the (effective) **pumping speed** $S$, and also **conductance** $L$ which both have dimension volume / time (e.g. litres/sec), i.e. these are "volumetric flow rates" rather than mass flow rates. Multiplying these volume flow rates by the entrance gas pressure (or by the pressure difference in case of conductance) produces the (pump-) throughput, a mass flow rate again, i.e., at given temperature a "pumped particle flux" (s$^{-1}$).
Pumping speeds and conductances typically depend on pressure themselves, and on temperature $T$ and the type of gas (mass $m$). In the molecular flow regime the pressure dependence in $S, L$ diminishes, and $S$ ($L$) can be written as product $S, L = A_{S,L} \cdot v$, of an area and velocity, with $v \propto \sqrt{T/m}$. Implementation of a specified volumetric flow (e.g. a given effective pumping speed at pumps) in EIRENE is via effective surface area and sticking probabilities, see section 2.6.1, input flag **RECYCT**
All **densities** are in cm$^{-3}$.
All **velocities** are in either cm s$^{-1}$ or in (isothermal) Mach-number units.
All geometrical data are in cm.
Note: consequently, e.g., energy densities are in eV cm$^{-3}$, and energy fluxes are given in eV amp, i.e., in watt.
Hence: In order to convert from EIRENE **energy density** units (eV cm$^{-3}$) into **pressure** units in N m$^{-2}$ (i.e., Pascal: Pa) one must multiply by $1.6022 \times 10^{-19}$ (eV to N m) and multiply by $1 \times 10^6$ (cm$^{-3}$ → m$^{-3}$). Check: 1 Pa gas at 300 K corresponds to about $2.7 \times 10^{14} \times 0.026$ eV cm$^{-3}$ ≈ $7 \times 10^{12}$ eV cm$^{-3}$, or: a gas pressure P in Pa, at gas temperature T (K) corresponds to a gas density n with $n(\mathrm{m}^{-3}) = P/k_B T$, with $k_B = 1.38 \times 10^{-23}$ the Boltzmann constant in J K$^{-1}$ units. At 300 K and 1 Pa, the corresponding gas density is $n = 2.415 \ldots 10^{20}$ m$^{-3}$ = $2.415 \ldots 10^{14}$ cm$^{-3}$.
For example, energy densities as computed by EIRENE defaults tallies 5, 6, 7 and 8 (see table 6.3) must be converted into "temperature densities" (a factor 2/3, if one neglects the energy of the directed motion) and then be re-scaled to pressure units as described above.
For further scaling to other pressure units note, e.g.:
101.3 Pa = 760 mTorr = 1 mbar
Further: momentum is in g cm s$^{-1}$, and a **momentum flux** or momentum rate of change is in g cm s$^{-1}$ amp (per cell) and the momentum source density in the EIRENE tallies for interfacing with plasma codes is then in g cm s$^{-1}$ amp cm$^{-3}$.
Hence, conversion from g cm s$^{-1}$ amp into Pa m$^2$ (momentum source per cell in plasma momentum balance equations in SI units) is done by firstly multiplying with $1 \times 10^{-5}$ (then: kg m s$^{-1}$ amp)

and then dividing by $1.6022 \times 10^{-19}$ (hence: $\mathrm{kg\,m\,s^{-2}}$). After dividing by the cell volume ($\mathrm{m^3}$) the momentum source is in $\mathrm{Pa\,m^{-1}}$, or $\mathrm{N\,m^{-3}}$ (force density).

## 2.1 Input data for operating mode

**General Remarks**

The variables in this block control some of the more general options in EIRENE such as overall running time, usage of dump files but also a few parameters depending on the simulation model (drifts included or not, etc.).

**The Input Block**

```
        TXTRUN
*** 1. Data for operating mode
```

first card: integer flags controlling global options for the entire run.

```
        NPRLL  NMODE  NTCPU  NFILE  NITER0  NITER  NTIME0  NTIME
```

next input line is only for EIRENE$_{2002}$ and younger, and optional in these versions. Defaults, if this card is not included, are given below.

```
        NOPTIM NOPTM1 NGEOM_USR NCOUP_INPUT NSMSTRA NSTORAM
      .   NGSTAL NRTAL NREAC_ADD
```

next line (second mandatory line): logical flags for global options [format: 12(5L1,1X)].

```
        NLSCL NLTEST NLANA NLDRFT NLCRR
      .   NLERG NLIDENT NLONE NLMOVIE NLDFST
      .   NLOLDRAN NLCASCAD NLOCTREE NLWRMSH NEXVS
      .   NLTRIMESH NLSPCSCL NLSPCSCL_ON NLSOLEDGE
```

This line was changed in EIRENE tag MsV-2023Mar-PBoernerUnified-beta to

```
        NLSCL NLTEST NLANA NLDRFT NLCRR
      .   NLERG NLIDENT NLONE NLMOVIE NLDFST
      .   NLRANMAR NLCASCAD NLOCTREE NLWRMSH NEXVS
      .   NLTRIMESH NLSPCSCL NLSPCSCL_ON NLSOLEDGE
```

Next input lines are only for EIRENE$_{2005}$ and younger, and optional in these versions. There can be any number of these cards starting with character string *CFILE* in the input file.

```
        "CFILE" DBHANDLE DBFNAME
```

**Meaning of the input flags for operating mode**

**TXTRUN** At least one line of text to identify the run on printout and plot files.

Each text card must start with *. The first of these text lines will appear on all plots, the full text will be echoed at the beginning of the printout file

**NPRLL** (Formerly at this place: NMACH, flag for machine precision (round off errors), now: out.)

The new (2018) flag NPRLL controls the various options for (MPI) parallelization modes, i.e. the strategies to assign processes to strata. This is defined in PEDIST.f (initialization phase) by filling the matrix PROCFORSTRA(strata,processors).

= 1 old default: attempting proportional allocation and, simultaneously, load balancing (see under: stratified sampling, the remarks towards the end of section 1.3.3.2).

= 2 not in use

= 3 not in use

= 0 new default: "embarrassingly parallel mode": all processes do the full MC job (all strata).

Most transparent, and simplest method, load balancing guaranteed. But perhaps not optimal wrt. communication

= −1 user defined distribution of strata to processes (subr. PEDIST, calls: PEDIST_USR.f)

**NMODE** Operating mode

= 0 EIRENE run as stand-alone-code. Code coupling segment *couple_Dummy.f* may be used. Input block 14 has a fixed format, see section 2.14.

≠ 0 EIRENE calls the code coupling subroutine INFCOP in the code coupling segment *couple_Name.f* where 'Name' is a character string identifying the particular external code, to which EIRENE is to be coupled (e.g.: Name = B2, Name = DIVIMP, Name = U-file, Name = FIDAP, Name = EMC3, Name = OSM, etc.).

Hence: the routine INFCOP is called by EIRENE for communication with external data sources, e.g., external data structures for iterative mode by coupling to other codes. The calling program for the first 3 entries of INFCOP is subroutine INPUT, and the call to subroutine INFCOP is after reading 13 blocks from the formatted input file (READ (IUNIN, ...)). A $14^{th}$ block of the input file is read from subroutine INFCOP with the format (if any) specified there. At entry IF0COP geometrical data are provided (overruling corresponding data in input block 2).

At entry IF1COP plasma profiles (more generally: background medium data) are defined (overruling the background data specified in block 5).

Any other data (e.g., surface- or volume source distributions overruling the data in input block 7) are expected from entry IF2COP.

> 0 If NMODE > 0, subroutine INFCOP is called once again after each completed stratum (at the entry IF3COP(ISTRAA,ISTRAE)) to return data to another code (post-processing). The call to IF3COP is from subroutine MCARLO, at the end of the DO 1000 -loop over the strata. One final call to the interfacing routine (entry IF4COP) is implemented after the calls to the EIRENE printout- and plot routines, e.g. to perform global balances etc. after summation of the contributions from the individual strata.

**NTCPU** Maximum number of CPU seconds allowed for this run. NTCPU must be less than or equal to the time parameter in the job-card (if any).

If more than one iteration is carried out (NITER, see below, this section) or more than one time cycle (NTIME, see below, this section), then each iteration or cycle can take up to NTCPU seconds.

In 2008 the definition of NTCPU was slightly changed: the initialisation overhead is not any longer included in NTCPU. I.e., this variable NTCPU now is close to the true Monte Carlo sampling time. The total CPU-time, including initialisation overhead as well as post-processing is printed at the end of an EIRENE run (see: "total CPU-time of this run" in printout file).

**NFILE** Flag for the use of dump files FT10, FT11, FT12, FT13, FT14 and FT15.

= 0: Neither reading nor saving of data is done.

= JKLMN (NFILE a 5 digit integer

**N = 1** EIRENE writes output data from this run into files FT10 and FT11 to save them for plot or printout options in this or in later 'read only' runs with the same input file.

**N = 6** same as NFILE-N=1, but only the data for the sum over all strata are written (sufficient, e.g., for **BGK!**-iterations, or for post-processing, (subroutine DIAGNO, see input block 12) as long as this post-processing only is to be done on total results (not on contributions from individual strata).

**N = 2** EIRENE reads output data from an earlier run from files FT10 and FT11. No new particle histories are computed, only the requested output is printed and plots are produced. Iteration cycles or time cycles (if any) are abandoned.

**N = 7** same as NFILE-N=2, but only the data for the sum over strata are read. See N=6 option described above.

**M = 1** EIRENE writes geometry data into file FT12. These data are the output from subroutines GRID and VOLUME in the initialization phase.

**M = 2** EIRENE reads geometry data from file FT12. The geometry subroutines GRID and VOLUME are not called. This option should be used if the geometry has not changed as compared to an earlier run. It reduces the CPU costs for the overhead.

**L = 1** EIRENE writes plasma data and atomic and molecular data ("A&M data") into file FT13. These data are output from subroutines PLASMA, XSECTA, XSECTM, XSECTI, XSECTP in the initialization phase. At the end of a run, some background medium data may have been modified in subroutine MODUSR, (iterative mode) see NITER flag below. In this case FT13 may be re-written to prepare for next iterations.

Also the primary source parameters (input block 7) are saved, for later iterations or time-steps.

**L = 2** EIRENE reads plasma data, A&M data from file FT13. It also reads the entire input block 7 from FT13, and overwrites the input read from this block 7 by those data. Routines PLASMA, XSECTA, XSECTM, XSECTI, XSECTP are not called. This option should be used if neither the plasma background nor the selection of atomic processes to be used, nor the primary source model (block 7) has changed as compared to an earlier run. It then reduces the CPU costs for the overhead.

**L = 3** Acts as if both NFILE-L=1 and NFILE-L=2. I.e. reading background data from file, and writing new background data onto file at the end of the run, the time-step or the iteration. For continuation of iterative calculations, for example.

**L = 4** Same as NFILE-L=3, except: primary source data (input block 7) are not read from fort.13, but are taken as specified in input block 7. This allows to modify these primary sources parameters during iterations or time-steps not only via module MODUSR, but directly via input file.

**L = 6,7,8,9** Same as L=1,2,3,4, respectively, but XDR file format is used.

**K = 1** EIRENE saves some data for optimizing non-analog sampling and stratified source sampling on file FT14.

**K = 2** EIRENE reads from file FT14 and tries to optimize operation for the next run, time-step or iteration. Currently only allocation of CPU time in stratified source sampling is optimized. More details: see section 2.1.1 below.

**K = 3** Acts as if both NFILE-K=1 and NFILE-K=2.

**J = 1** Only for time-dependent option (see block 13). EIRENE writes "census data" at the end of last time-step onto file FT15.

**J = 2** EIRENE reads "census data" from file FT15, and uses it as initial distribution for the coming next time-step.

**J = 3** Acts as if both NFILE-J=1 and NFILE-J=2.

**NITER0** Initial iteration number: (Default: NITER0=1) Irrelevant parameter. Only needed for book keeping and printout. EIRENE labels the iterations from NITER0 to NITER.

**NITER** Number of iterations, if EIRENE runs in "iterative mode".

$> 0$ EIRENE calls user supplied subroutine MODUSR after completing the run; some model parameters may be modified here for the next iteration step, and some results from the previous step may be saved on a file.

$> 1$ EIRENE recalls itself but does not read from the formatted input file again. This recalling is repeated NITER times (including the first iteration). The CPU time NTCPU is used for each iteration. Hence the true CPU time then is $NTCPU \cdot NITER$.

**NTIME0** Initial time-cycle number: (default: NTIME0=1). Irrelevant. Only needed for book keeping and printout. EIRENE labels the time steps from NTIME0 to NTIME.

**NTIME** Total number of iterations ("cycles") in time carried out in one single run. The total time per cycle is defined as NTMSTP · DTIMV (see below, input block 13).

After each time-cycle the subroutine TMSTEP is called. In this routine the "census arrays", which store the test particle population at time $t_i$:

$t_i = t_{i-1} + \Delta t = t_0 + i\Delta t = TIME0 + ITIME \cdot [NTMSTP \cdot DTIMV]$

are filled and prepared for the next time-cycle. The census arrays from the previous time cycle (if any), i.e., at $t = t_{i-1}$, are overwritten here.

After the last time-cycle, the census arrays are written on file fort.15, in order to permit continuation in time in a next run.

The background conditions (and source distributions or any other input parameters) for the next time cycle can be modified in subroutine TMSUSR($t_i$), which is called from subroutine TMSTEP.

If the population on the census array is not empty or known from a previous run (NFILE-J flag, see above), then this census population defines one additional stratum for the current cycle. I.e., the census population then determines the initial condition for the distribution function $f(\underline{r}, \underline{v}, i, t = t_0)$. The source strength FLUX is computed from that initial condition.

**NOPTIM** Default: NRAD, = total number of grid cells

(verify this default in case of older versions: there it may have been NOPTIM=1)

NOPTIM is the first dimension of the arrays IGJUM3(ICELL,ISURF), which may be used for optimizing code performance by reducing unnecessary geometrical calculations. See 'CH3-cards' in input block 8 (section 2.8).

**NOPTM1** Default: 1

**NGEOM_USR** for value =1: user defined (external) geometry package (LEVGEO=10), then no grid storage is provided in EIRENE. Default: 0

**NCOUP_INPUT** =1: Storage for data transfer via coupling routines, =0: no such storage. Default: 1

**NSMSTRA** "Sum over strata" disabled for value 0, enabled for value 1. Default: 1

**NSTORAM** Storage vs. speed in atomic data evaluation. Maximum storage, fastest computation: =9. Minimum storage, maximum work (slowest option) =0. Default: 9

**NGSTAL** Storage for spatial distribution of surface tallies on non-default standard surfaces for value =1. For value =0: only total (spatially integrated) surfaces fluxes. Default: 0

**NRTAL** Condensing mesh cells into fewer larger cells, for output volume tallies. The underlying fine mesh has NRAD cells (see input block 2). The coarser mesh, obtained from condensing cells into one larger cell, has NRTAL cells. Default: 0: Then internally: NRTAL=NRAD, and no condensation is carried out.

NCLTAL(IRAD) = IRTAL: grid cell IRAD is condensed into the larger cell IRTAL. I.e.: output is average over a larger cell IRTAL, which is comprised of all cells IRAD such that NCLTAL(IRAD) = IRTAL, IRAD=1, NRAD

The input data in input block 5 (background medium) are always given on the fine mesh (size: NRAD)

For output tallies (cell averaging) several cells IRAD1, IRAD2, ... can be condensed into one larger cell IRTAL. Cell volumes, scoring, statistics are automatically done on the coarser grid (size: NRTAL).

The index array NCLTAL(NRAD) can be defined in the problem specific "user" routines (see chapter 3), e.g.: INIUSR, GEOUSR, etc..

Default: NCLTAL(IRAD)=IRAD for IRAD=1,NRAD

**NREAC_ADD** Storage for additional reaction decks read onto EIRENE arrays in USR-routines, post-processing, etc. Default: 0

Next input card, global logical switches. The switches 10–15 in this card may have version specific meanings and are for internal code options only.

**NLSCL** Some volume averaged tallies are re-scaled in order to exactly preserve the total number of particles, which otherwise would be the case only up to statistical precision (due to the use of track-length estimators). EIRENE computes three factors FATM, FMOL and FION such that particle balances for atoms, molecules and test ions, respectively, are accurately observed, if NLSCL = TRUE.

**NLTEST** Tests for consistency between cell numbers and geometrical data along the particle tracks are carried out at each point of collision. If inconsistencies are detected, the history is stopped and an error message is printed. The contribution of these particles to the particle- and energy balances is stored in the bins "PTRASH" and "ETRASH" respectively.

**NLANA** De-activates (NLANA=.TRUE.) all non-analog sampling distributions, such as biased source sampling, splitting, etc. Select NLANA=.TRUE., if particle trajectory plots are used to get an intuitive picture of what is going on physically.

**NLDRFT** Drift component in the bulk ions velocity distribution is included, i.e. the assumed underlying distribution in velocity space is a drifting Maxwellian for volumetric background tallies of bulk particles. (see input block 5, input tallies VXIN, VYIN, VZIN)). Otherwise (if NLDRFT = FALSE) an isotropic Maxwellian distribution is assumed for the background particles and the input for VXIN, . . . , VZIN is ignored.

This flag also affects the output tallies for energy exchange, momentum exchange between test particles and background particles, as well as sampling from linear collision kernels.

**NLCRR** Correlated sampling is used. See discussion at end of section 1.8.

**NLERG** The case is automatically reduced to a case for estimating cell volumes by utilizing an "ergodic property". More details: see section 2.1.2 below.

**NLIDENT** In multi-processor calculation mode NLIDENT forces all processors working on the same stratum to use the same sequence of random numbers and hence to carry out exactly identical work. This flag can be used to test the parallelized code version.

**NLONE** The case is automatically reduced to a single species and "one speed transport" problem (to simplify setting up cases for comparison with analytic results). (not ready, don't use)

**NLMOVIE** reset a number of model parameters to enable a series of geometry plots for making particle trajectory movies. This option only works in connection with time dependent mode, see NTIME flag described above, and input block 13 (section 2.13). More details: see section 2.1.3 below.

The next 5 logical flags are for internal use only, in proprietary parts of the code. They should not be activated without checking the underlying code. Note in particular: NLOLDRAN, see below. These variables may disappear and re-appear with other names/features, no code backward compatibility. They should all be set to FALSE, except probably NLOLDRAN, with may be code version dependent.

**NLDFST** for internal use. Testing time step options in Fokker Planck solver (folion.f)

**NLOLDRAN** Switch between random number generators. Opposite meaning in different code versions. One should set this switch such that generator "H1RN" is used, and NOT the congruential generator. See function "ranf.f". The "junk" congruential generator has been kept, so far, to permit full backward compatibility. But the code now (from revisions April 17 on) heavily complains if it is used.

**NLRANMAR** has replaced NLOLDRAN since EIRENE tag MsV-2023Mar-PBoerner-beta. It indicates that the "junk" congruential generator has been completely removed. NLRANMAR switches on testing of generator "H1RN".

**NLCASCAD** for internal use. Unfinished option for enforcing fully analog collision cascades, rather than weighting (collide.f)

**NLOCTREE** for internal use. Testing CPU optimization for timea.F routines ("additional surfaces")

**NLWRMSH** write out additional surface information for internal (FZJ) triangular grid generators (input.f)

**NEXVS** not in use?

**NLTRIMESH** stop run after preparation for triangulation code (NLWRMSH)

**NLSPCSCL** extend volume averaged and surface averaged tallies for species specific rescaling, execute species specific scoring for the respective tallies and calculate refined scaling factors

**NLSPCSCL_ON** apply the refined scaling factors

**NLSOLEDGE** switch on use of FNUEQI_SOL in FOLION and add contributions tally MIPL in FPKCOL

Optional input cards, for pathways to external databases. Their presence is identified by the starting character string CFILE of these cards.

**DBHANDLE** Name of the database which is specified in this "CFILE" card.
DBHANDLE is called "FILE" in JSON input.
The format of a data file is identified in EIRENE by its name. Possible database names currently recognized are:

AMJUEL, HYDHEL, METHAN, H2VIBR, HYDRTC, ADAS, SPECTR, PHOTON, POLARI,

SPUTER, TRIM, TRMSGL,

with volumetric collision databases in the first block, and surface interaction databases in the second.

**DBFNAME** Absolute or relative path and name of file to be used for this database.
DBFNAME is specified as "PATH" in JSON input

Example for formatted input:
*CFILE AMJUEL /home/path/AMdata/amjuel.tex*
*CFILE TRIM /home/path/Surfacedata/TRIM/trim.dat*

Example for JSON input:

```
"CFILE": [
  {
    "FILE": "AMJUEL",
    "PATH": "/home/path/AMdata/amjuel.tex"
  },
  {
    "FILE": "TRIM",
    "PATH": "/home/path/Surfacedata/TRIM/trim.dat"
  }
]
```

Default: If the specification of an external database name and path, which is later (e.g. in block 4 (section 2.4) or block 6 (section 2.6)) referred to in a particular run, is not given here, then these data files are expected, usually with the same name, in the local directory from which the calculation takes place. Due to historical reasons and for backward compatibility for some data files also different default names are recognized.
Default file names to be used in this local directory currently are
AMJUEL,
HYDHEL,
METHAN or "METHANE", for database name METHAN
H2VIBR,
HYDRTC,
SPECTR,
PHOTON,
POLARI,
SPUTER,
TRIM or "fort.21", for database name TRIM
For ADAS (AMdata) and TRMSGL (Surfacedata) there is an exception from this scheme. In these cases no default file names are available. The notation "ADAS" and "TRMSGL" here are only synonymous for any atomic or surface data file which is stored in the same format as the so called **ADAS!** (**ADAS!**) adf11 files and the TRIM conditional quantile data tables, independent on whether these tables originate from **ADAS!** or TRIM.
And input variable DBFNAME specifies only the path to the directory which is the root of the data tabulated in **ADAS!** (tabular) format and the TRIM format data tree, respectively.
Example:
*CFILE ADAS /home/path/AMdata/Adas_Eirene_2010/adf11/*
*CFILE TRMSGL /home/path/Surfacedata/TRIM/*
The particular data file from these trees (e.g.: SCD96 in case of **ADAS!**, or A_on_B in case of TRMSGL), the chemical element and the charge stage of a particular ion $Z = 1, \ldots, Z + 1$ (e.g.: C, 4), are detailed later in input block 4 (for volumetric collision processes based on **ADAS!** adf11 format), and input block 6 (TRMSGL, for the surface reflection database based on TRIM) see below, sections 2.4 and 2.6, respectively.

### 2.1.1 Automated stratification optimization: proportional allocation

The stratified sampling in EIRENE can improve the performance (measured in terms of statistical error per CPU-time), if proportional allocation of sample size $N_k$ to strata source strength $S_k$ can be achieved. See section 1.3.3.2 for more information on stratified source sampling. The input flag ALLOC in input block 7 (section 2.7) controls this allocation.
Monte Carlo codes such as EIRENE are, in principle, trivially parallelizable by just distributing the Monte Carlo trajectory calculation over the available compute nodes. If parallelization is activated in combination with stratified source sampling, however, and strata are assigned to compute nodes, then load balancing may be hard to achieve together with proportional allocation of weights to strata. This is because trajectories from some strata may be very short (e.g. transport into purely absorbing media, for example of impurity particles until ionisation) and very fast to compute, while from other strata these trajectories may be very long and time consuming to compute, e.g. in near diffusive regimes with many collisions. Often volume recombination strata in cold plasma regions fall into this category.
Load balancing and proportional (or otherwise optimal) allocation to strata then requires estimates of run-time per history for each stratum prior to the Monte Carlo simulation itself.
This information is stored in output stream FORT 14, and can be read back in into the next EIRENE run and used then for assignment of compute nodes to strata and vice versa. This is controlled by the NFILE-K flag described above.

### 2.1.2 The NLERG option for cell volumes

All non-transparent surfaces are made perfectly reflecting (mono-energetic, cosine distribution). All volumetric collision processes are de-activated. Except, possibly, for the very first flight, there is only either one atomic species (IATM=1) or one molecule species (IMOL=1). Unless otherwise specified in input block 13, a single time-step, with time limit DTIMV=1.0 (s) is set and up to NPRNLI=10 test flights are launched. Under such conditions, the particle density should be constant throughout the grid. Statistically significant deviations from that constancy indicate wrong cell volumes as compared to the volumes seen by the test particles (e.g. due to additional surfaces intersecting the grids, or complicated shapes of some additional cells (see input blocks 2 and 3). In a second run the volumes of these cells can then be set explicitly, e.g. in input block 8.

### 2.1.3 The NLMOVIE option for making movies of trajectories

documentation to be written

## 2.2 Input for Standard Mesh

**General Remarks**

EIRENE follows atoms, molecules or test ions in a 3-dimensional computational box. This box is discretized by zones (mesh cells), the boundaries of which are defined either by regular "standard mesh surfaces", i.e., co-ordinate surfaces, (**BREP!** representation of geometry, section 1.6) which are described here, and/or by zones whose boundaries are defined more generally by "additional surfaces" (see below block 3B). Alternative to this **BREP!** option there is also the option of unstructured meshes (Voxel representation, loc. cit.) of triangles ( in 2D) or tetrahedra (in 3D).
In 2-dimensional applications the co-ordinate surfaces automatically collapse to coordinate lines, and in 1-dimensional applications they collapse to coordinate (grid) points.
As long as test particles travel inside the "standard mesh", all cell indexing computations are done automatically. At transition into the more general "additional mesh" and inside these additional cells, this indexing has to be specified by the flags ILSWCH, ILCELL etc. by the user, in input block 3B.
There are up to 3 sets of standard co-ordinate surfaces, each set being parameterised by the arrays RSURF(i1), PSURF(i2) and TSURF(i3).
Here RSURF(1), RSURF(2), ..., RSURF(NR1ST) are grid-points (parameters) in the 1st co-ordinate direction, mostly the radial or x direction (depending on the geometry level, internal parameter LEVGEO, see below) defined by the input data in sub-block 2A. I.e., there are NR1STM= NR1ST-1 cells in the 1st co-ordinate direction (e.g. radial or x direction). With regard to the terminology in section 1.6, (1.85), a corresponding co-ordinate surface is labelled with a double index 1,IR and is defined by the equation
$f_1^{1,IR}(x, y, z)$ =constant=RSURF(IR).
PSURF(1), PSURF(2), ..., PSURF(NP2ND) are grid-points in the 2nd co-ordinate direction, typically the poloidal or y direction (sub-block 2B), and there are NP2NDM=NP2ND-1 cells in this direction. A corresponding co-ordinate surface is labelled $2, IP$ and is defined by the equation
$f_1^{2,IP}(x, y, z)$=constant=PSURF(IP).
In case NP2ND = 1 this 2rd grid is absent and the calculation automatically collapses to a 1D case, i.e. a case with only 1 set of co-ordinate points. This does not mean that the 2rd, ignorable coordinate must be straight Cartesian (y-direction). E.g. a spherically symmetric volume, but with only radial resolution in space and polar and azimuthal symmetry, then is still referred to as 1-dimensional calculation.
In case of the NLPLG option (mesh in x-y plane generated by a set of polygons) for simplicity and by an abuse of language we sometimes refer to the polygons themselves as radial surfaces or "radial polygons" RSURF and refer to the polygons obtained by connecting all points with a particular index along the various radial polygons as "poloidal polygons" PSURF. No use is made in EIRENE of the arrays RSURF and PSURF in connection with the polygon mesh option. The NLPLG option permits to run EIRENE on 2D computer generated meshes with grid cuts in one direction.
Furthermore, TSURF(1), TSURF(2), ..., TSURF(NT3RD) are grid-points in the third co-ordinate direction, mostly the toroidal or z direction (sub-block 2C). A co-ordinate surface is labeled $3, IT$ and is defined by the equation $f_1^{3,IT}(x, y, z)$ =constant=TSURF(IT).
In case NT3RD = 1 this 3rd grid is absent and the calculation automatically collapses to a 2D

case, i.e. a case with only 2 sets of co-ordinate lines. This does not mean that the 3rd, ignorable coordinate must be straight Cartesian (z-direction). E.g. a full torus, but with only radial and poloidal resolution and toroidal symmetry then is referred to as 2-dimensional calculation.
In addition to these grids and, depending on the geometry options chosen, EIRENE then computes further "derived" grid arrays, for example a "(radial) surface labeling grid" RHOSRF, which may or may not be identical to the RSURF grid, and a "zone centered (radial) surface labeling grid" RHOZNE(IR) defined by:
RHOZNE(IR) = 0.5 ∗ (RHOSRF(IR)+RHOSRF(IR+1))
These grids may be used for the input of closed form radial- or x direction plasma profiles or as abscissae in plots of e.g. poloidally (or y) and toroidally (or z) averaged tallies.
We have found it convenient to allow for several (NBMLT) identical copies of such "standard meshes", separated and bounded by arbitrary "additional surfaces". Input flags for this "mesh multiplication" option are comprised in sub-block 2D.
As mentioned above, in addition to these regular grids one can define cells of almost arbitrary complexity by using the "additional surfaces" defined in sub-block 3B. Such surfaces are labeled $0, IA$ in EIRENE. Appropriate cell number switching flags must be set on transparent additional surfaces separating these general "additional cells". Data relevant for the additional cells (e.g. the volume) are read in sub-block 2E. The number of additional cells is NRADD.
Therefore, the total number of cells in an EIRENE run is NSBOX, with
NSBOX = (NR1ST · NP2ND · NT3RD) · NBMLT + NRADD .
Cell volumes in the standard mesh region, as well as the center of mass vector in each of these cells are computed automatically. The corresponding data in the additional cell region must be specified by the user, e.g. in the input block 2E or 8, or in the user supplied routine GEOUSR (section 3.6), which is called by EIRENE in the initialization phase.

**The Input Block**

```
*** 2. Data for Standard Mesh
      (INDGRD(J), J=1,3)
** 2A. x- or radial co-ordinate surfaces
      NLRAD
      IF (NLRAD) then
        NLSLB  NLCRC  NLELL  NLTRI  NLPLG  NLFEM  NLTET NLGEN
        NR1ST  NRSEP  NRPLG  NPPLG  NRKNOT NCOOR
        IF (INDGRD(1).LE.5) THEN
          IF (NLSLB.OR.NLCRC.OR.NLELL.OR.NLTRI) THEN
            RIA  RGA  RAA  RRA
            IF (NLELL.OR.NLTRI) THEN
              EPIN  EPOT  EPCH  EXEP
              ELIN  ELOT  ELCH  EXEL
              IF (NLTRI) THEN
                TRIN  TROT  TRCH  EXTR
              ENDIF
            ENDIF
          ELSEIF (NLPLG) THEN
            XPCOR  YPCOR  ZPCOR  PLREFL
           (NPOINT(1,K)  NPOINT(2,K), K=1,NPPLG)
```

```
            DO 21 I=1,NR1ST
              (XPOL(I,J) YPOL(I,J),   J=1,NRPLG)
21          CONTINUE
          ELSEIF (NLFEM) THEN
            XPCOR YPCOR ZPCOR
            NRKNOT
            (XTRIAN(I),I=1,NRKNOT)
            (YTRIAN(I),I=1,NRKNOT)
            DO ITRI=1,NR1ST
              I,NVERT(1,ITRI),NVERT(2,ITRI),NVERT(3,ITRI),
                NEIGH(1,ITRI),NSIDE(1,ITRI),IPROP(1,ITRI),
                NEIGH(2,ITRI),NSIDE(2,ITRI),IPROP(2,ITRI),
                NEIGH(3,ITRI),NSIDE(3,ITRI),IPROP(3,ITRI)
            ENDDO
          ENDIF
        ELSEIF (INDGRD(1).EQ.6) THEN
          IF (NLSLB.OR.NLCRC.OR.NLELL.OR.NLTRI) THEN
            RIA RGA RAA
          ELSEIF (NLPLG.OR.NLFEM.OR.NLTET) THEN
            XPCOR YPCOR ZPCOR
          ENDIF
        ENDIF
      ENDIF
** 2B. y- or poloidal grid surfaces
      NLPOL
      NLPLY NLPLA NLPLP
      NP2ND NPSEP NPPLA NPPER
      IF (INDGRD(2).LE.5) THEN
        YIA YGA YAA YYA
      ENDIF
** 2C. z- or toroidal grid surfaces
      NLTOR
      NLTRZ NLTRA NLTRT
      NT3RD NTSEP NTTRA NTPER
      IF (INDGRD(3).LE.5) THEN
        ZIA ZGA ZAA ZZA ROA
      ENDIF
** 2D. mesh multiplication
      NLMLT
      NBMLT
      IF (NBMLT GT 1) (VOLCOR(NM), NM = 1,NBMLT)
** 2E. additional cells outside standard mesh
      NLADD
      NRADD
      IF (NRADD.GT.0) (VOLADD(NM), NM = 1,NRADD)
```

**Meaning of the Input Variables**

**INDGRD** This index controls the meaning of input variables of different standard grid options.

INDGRD(1) for the radial (or x-direction) grid

INDGRD(2) for the poloidal (or y-direction) grid

INDGRD(3) for the toroidal (or z-direction) grid

*Data for first standard mesh*: *radial or x grid RSURF*

**NLRAD = .TRUE.**

A radial or x grid is defined. Otherwise the complete sub-block 2A may be omitted. Depending upon the logical parameters in the next input card the "geometry - level" variable LEVGEO is set internally.

**LEVGEO**

= 1 Cartesian coordinates x (and y)

= 2 polar coordinates r (and $\theta$)

= 3 general curvilinear coordinates: a full 2D mesh (polygonal coordinate lines) is used in the x - y plane. Grid cuts are permitted in the y-direction.

= 4 a 2D "finite element" mesh of triangles is used in the x - y plane

= 5 a 3D "finite volume" mesh of tetrahedrons used

= 10 a general, user defined geometry block is used. All geometrical calculations are performed in problem specific routine VOLUSR, TIMUSR, etc.

If NLRAD=.FALSE., then no spatial grid is defined and the default geometry level LEVGEO = 1 is used. Volume discretization may still be achieved "by hand" by defining "additional surfaces" (input block 3b) and appropriate cell number switching.

**INDGRD(1)**

= 1 standard grid option; the input parameters are used as described below.

= 2 As for INDGRD(1)=1, but in case of LEVGEO = 2 by this option a radial grid is defined such that the spacing of radial surfaces is not equidistant at some poloidal position, but instead such that the area enclosed between two neighboring surfaces is kept constant. The input variable RGA is irrelevant in this case.

= 3, 4, 5 not in use

= 6 Data for NR1ST radial surfaces are read from the work array RWK. These data must have been written onto RWK in the user supplied subroutine INFCOP. By this option grid parameters may directly be transferred into EIRENE from other files, e.g. from plasma transport codes (see below: section 2.14. Data for interfacing routine INFCOP).

One and only one of the next 8 logical variables NLSLB ... NLTET, NLGEN must be .TRUE. This variable defines the so called "geometry level" of a run, i.e. the internal EIRENE variable LEVGEO

**NLSLB = .TRUE.**

Geometry level: LEVGEO = 1

Cartesian geometry, the x co-ordinate is discretized by setting

$x_I$ = RSURF(I) I = 1, NR1ST.

Furthermore, the flux-surface labeling grid RHOSRF(I) is identical with the grid RSURF(I).

**NLCRC = .TRUE.**

Geometry level: LEVGEO = 2

Cylindrical or toroidal geometry, 1D ("radial") mesh of concentric, circular surfaces. Polar coordinates are used in the x-y plane. The third coordinate (either $z$ or toroidal angle $\phi$) is either ignorable or discretized according to options described below (block 2c).

The radial surfaces are given by $r^2 = x^2 + y^2$ = const. and radial coordinate $r$ is discretized by setting $r_I$=RSURF(I) I=1,NR1ST. Furthermore RHOSRF(I) = $\sqrt{\frac{AREA}{\pi}}$ where AREA is the area inside surface number $I$. Thus for this option one has again: RHOSRF(I) = RSURF(I), I=1,NR1ST.

**NLELL = .TRUE.**

Geometry level: LEVGEO = 2

Mesh of nested, but not necessarily concentric or confocal elliptical flux surfaces. The equation for the "radial" surface is $(x - EP)^2 + (y/EL)^2 = r^2$. The radial coordinate $r$ is discretized by setting $r_I$=RSURF(I) I=1,NR1ST.

EP and EL may vary with coordinate $r$. These parameters are stored in the arrays EP(I), EL(I), I = 1, NR1ST which now are used in addition to RSURF to define one coordinate surface in the first (radial or $x$- grid).

RHOSRF: as in NLCRC option.

Note: RHOSRF and RSURF may differ in this case.

**NLTRI = .TRUE.**

Geometry level: LEVGEO = 2

to be written: triangularity in mesh of nested closed algebraic surfaces

**NLPLG = .TRUE.**

Geometry level: LEVGEO = 3

The mesh in the x-y plane is described by NR1ST polygonal arcs of length NRPLG each. A polygon may consist of several "valid" and "invalid" parts (to account for "grid cuts" in **CFD!** meshes). The "invalid" parts of a polygon are not seen by test particles and are allowed for in EIRENE only in order to facilitate index mapping in case of runs coupled to plasma transport models, which resort to computer generated meshes including grid cuts.

The polygons must not intersect each other.

In this case RHOSRF(1)=0., and RHOSRF(I) is the area enclosed by polygon number 1 and polygon number I.

**NLFEM = .TRUE.**

Geometry level: LEVGEO = 4

The mesh in the x-y plane consists of NR1ST triangles, composed from NRKNOT knots.

In this case a flux surface labeling grid RHOSRF is not defined.

**NLTET = .TRUE.**

Geometry level: LEVGEO = 5

3D discretization of volume by tetrahedrons. For this grid option please make contact to the authors.

**NLGEN = .TRUE.**

Geometry level: LEVGEO = 10

Arbitrary geometrical configuration. Mesh consists of NR1ST arbitrarily shapes cells (in any dimension). Particle tracing routines must be provided by user (VOLUSR, SAMUSR, TIMUSR, LEAUSR)

**NR1ST** Number of grid-points in the radial (or x-direction) standard mesh

if NR1ST $\leq$ 1, no radial (or x-direction) standard mesh is defined.

**NRSEP** This flag is active for LEVGEO = 1 or LEVGEO = 2. Otherwise it is irrelevant.

The first (x- or radial) standard mesh is composed by two equidistant x- or radial grids of co-ordinate surfaces with different grid density. There are NR1ST-NRSEP+1 grid-points in the first, and NRSEP grid-points in the second part. The grid-point RSURF(NR1ST-NRSEP+1) belongs to both parts.

**RIA** left endpoint of standard grid (internally set $\geq$ 0 if LEVGEO =2); RSURF(1)=RIA

**RGA** boundary separating first and second part of standard grid with different grid-point densities; RSURF(NR1ST-NRSEP+1)=RGA

**RAA** right endpoint of standard grid: RSURF(NR1ST)=RAA

**RRA** if RRA > RAA, one additional, outer void zone is defined

RSURF(NR1ST) = RRA, and the parameter RAA now determines the surface no. NR1ST - 1.

(irrelevant, if RRA $\leq$ RAA)

if NLELL = .TRUE.:

**EPIN** Value of EP(r) for cylindrical co-ordinate surface number 1 with $r_1$=RIA (see: NLCRC = .TRUE. option above)

**EPOT** Value of EP(r) for cylindrical surface number NR1ST with

$r_{NR1ST}$=RAA

**EPCH** Value of EP(r) for cylindrical surface number NR1ST+1 with

$r_{NR1ST+1}$=RRA

(irrelevant if RRA ≤ RAA)

**EXEP** The variation of the "shift function" EP(r) with r is given by

$EP(r) = EPIN + (\frac{r-RIA}{RAA-RIA})^{EXEP} \cdot (EPOT - EPIN)$

**ELIN** Value of EL(r) for cylindrical surface number 1 with

$r_1$=RIA (see: NLELL = .TRUE. option)

**ELOT** Value of EL(r) for cylindrical surface number NR1ST

with $r_{NR1ST}$=RAA

**ELCH** Value of EL(r) for cylindrical surface number NR1ST+1 with

$r_{NR1ST+1}$=RRA

(irrelevant for RRA ≤ RAA)

**EXEL** The variation of the "ellipticity function" EL(r) with r is given by

$EL(r) = ELIN + (\frac{r-RIA}{RAA-RIA})^{EXEL} \cdot (ELOT - ELIN)$

if NLTRI = .TRUE.:

**TRIN, TROT, TRCH, EXTR** to be written: the option for algebraically given triangular grids is currently no available.

if NLPLG = .TRUE.:

**NR1ST** number of polygons for discretization in "radial" or x direction

**NRPLG** Number of points per polygon

**NPPLG** Number of valid parts on each polygon. Each polygon is described by the x and y co-ordinates of NRPLG points. It is not necessary that all this points are used for the polygon. One can cut the polygon into several valid parts interrupted by parts which are not seen by the test particles. ( Default : NPPLG = 1 ). This option facilitates the use of 2-d computer generated meshes which contain topological grid cuts.

**XPCOR,YPCOR**

shift whole mesh by that vector in x,y-plane

**RFPOL** if RFPOL > 0., one additional polygon zone is defined, at a distance RFPOL outside the polygon NR1ST, and then NR1ST is increased by one.

(irrelevant, if RFPOL ≤ 0.)

**NPOINT(1,J)**

Index of the first point of the valid part number J (same for each radial polygon)

( Default : NPOINT(1,1) = 1 )

**NPOINT(2,J)**

Index of the last point of the valid part number J (same for each radial polygon)

( Default : NPOINT(2,1) = NRPLG )

**XPOL(K,I)** x-co-ordinate of the polygon point number K on polygon number I

**YPOL(K,I)** y-co-ordinate of the polygon point number K on polygon number I

if NLFEM = .TRUE.:

**NR1ST** number of triangles for discretization in x-y plane

**XPCOR,YPCOR**

shift whole mesh by that vector in x,y-plane

**NRKNOT** There are NRKNOT knots, by which the triangles are defined

**XTRIAN,YTRIAN**

x and y co-ordinates of the knots, respectively.

**NVERT(I,ITRI)**

Each triangle ITRI is defined by 3 points $P_1$, $P_2$ and $P_3$ from the set of NRKNOT knots.

NVERT(I,ITRI) is the number of point $P_I$ (I=1,2,3) in the set of knots for triangle ITRI.

**NEIGH(I,ITRI)**

The three sides of each triangle $S_1$, $S_2$, $S_3$ are defined such that $S_1$ connects $P_1$ and $P_2$, $S_2$ connects $P_2$ and $P_3$ and $S_3$ connects $P_1$ and $P_3$.

NEIGH(I,ITRI) is the number of the neighboring triangle at side $S_I$ for I=1,2,3.

**NSIDE(I,ITRI)**

NSIDE(I,ITRI) is the number (1,2 or 3) of the side of the neighboring triangle, which corresponds to side $S_I$ of the triangle ITRI to be written

**IPROP(I,ITRI)**

ISTS = ABS(IPROP) is the integer, by which a particular surface property is assigned to side I of the triangle ITRI. ISTS=0 stands for default grid option (transparent surface, cell indexing is done automatically). Otherwise ISTS is the number of an additional (ISTS < NLIMI) or non-default standard (NLIM < ISTS <NLIM+NSTSI) surface option as read in sub-blocks 3a or 3b, respectively. By default the normal vector of each side of a triangle points out of the triangle. In case IPROP < 0, this vector points into the triangle. This is relevant at surface options with ILSIDE $\neq$ 0.

if NLTET = .TRUE.:
Documentation not available here. Make contact with the authors.
if NLGEN = .TRUE.:

**NR1ST** number of cells in otherwise arbitrary mesh option NLGEN. Use problem specific routines (see chapter 3) to set up mesh, cell volumes, and flight intersection times for test particles.

*Data for second standard mesh*: *poloidal or y grid PSURF*

**NLPOL** A poloidal or y grid is defined. Otherwise the complete block 2B may be omitted and the volume averaged tallies are then automatically integrated over this co-ordinate.

**INDGRD(2)**

= 1 standard grid option

= 2, 3, 4, 5, 6 not in use (default: INDGRD(2)=1)

**NP2ND** Number of grid-points in y- or poloidal direction

**YIA,YGA,YAA,YYA**

**for the LEVGEO=1 option:**

the y grid PSURF is defined in the same way as the x grid, using the parameters YIA, YGA, . . . (cm) instead of RIA, RGA, . . .

**for the LEVGEO=2 option:**

the poloidal angle $\theta$ grid PSURF is defined in the same way as the radial $r$ grid was, using the parameters YIA, YGA, . . . (poloidal angle in degree) instead of RIA, RGA, . . .

for all options LEVGEO > 2: this input card is irrelevant.

Defaults: (needed for scaling, i.e., cell volumes):

YIA=0., YGA=0., YAA=1., YYA=1. in case of LEVGEO=1,

and YIA=0., YGA=0., YAA=360., YYA=360. in case of LEVGEO=2.

*Data for third standard mesh*: *toroidal or z grid TSURF*

**NLTOR** A toroidal or z grid is defined. Otherwise the complete block 2C may be omitted, defaults described below are used and the volume averaged tallies are then automatically integrated over this co-ordinate.

In case NLTOR = .TRUE., sub-block 2C must be read.

**INDGRD(3)**

= 1 standard grid option

= 2, 3, 4, 5, 6 not in use (default: INDGRD(3)=1)

One and only one of the next four Boolean variables must be TRUE

**NLTRZ = .TRUE.**

cylindrical approximation is used, i.e., TSURF is a grid in z direction. The co-ordinate surfaces are given by z=TSURF(L)

(Default: NLTRZ = .TRUE.)

**NLTRA = .TRUE.**

toroidal approximation is used. The coordinate line is a polygonal approximation of a circle, or of an angular section thereof.

In case NLTOR = .TRUE., the 3rd grid TSURF is a grid of toroidal angles. The toroidal segment (or the full torus) is approximated by NT3RD-1 straight cylindrical segments.

In case NLTOR = FALSE, there are NTTRA toroidal periodicity boundaries, such that the toroidal segment is approximated, again, by NTTRAM=NTTRA-1 straight cylinders, but without toroidal resolution in the results. (Note: If NLTOR, then: NTTRA = NT3RD, internally.)

The torus axis of the entire mesh can be shifted in radial direction by adding a radial offset ROA (see below) to the x coordinates of the poloidal mesh (RSURF,PSURF).

The radial shift of the poloidal mesh RMTOR of the approximated torus is computed from ROA such that the volume inside radial surface NR1ST is exactly equal to the volume of an exact torus with poloidal cross section defined by the shifted radial grid (first standard mesh: RSURF).

Due to the approximations made by defining a torus by NTTRAM = NTTRA - 1 straight cylinders, this condition is fulfilled only approximately for the other radial surfaces. RMTOR converges to ROA with increasing NTTRA. NTTRA $\simeq$ 30 is already a very good approximation.

(Default: NLTRA = FALSE).

**NLTRT = .TRUE.**

torus co-ordinates R,PHI,THETA. Presently being developed for NLSLB,NLPLG and NLTRI options. Not ready for use.

**NT3RD** Number of grid-points in z- or toroidal direction

(default: NT3RD = 1, i.e. no grid is defined)

**NTTRA** only needed in case NLTRA and .NOT.NLTOR. See above.

**ZIA,ZGA,ZAA,ZZA,ROA**

The 3rd grid ZSURF is defined in the same way as the x grid, using the parameters ZIA, ZGA, . . . (cm) instead of RIA, RGA, . . .

In case of NLTRZ = .TRUE., a z-grid is defined. ROA is irrelevant.

In case of NLTRA = .TRUE., ZIA and ZAA are toroidal angles (in degrees). A grid of toroidal angles is defined. For example use ZIA=0.0 and ZAA=360.0 for a full torus. In case NLTOR this grid also defines the toroidal resolution. If .NOT.NLTOR, then periodicity at the endpoints ZIA and ZAA is automatically enforced.

ROA is the radial shift of the poloidal cross section defined by the x-y grids. E.g.: if the x-y- grids are given with magnetic axis as origin, then ROA is the major radius. If the x-y-grids are already given with respect to the torus axis at their origin, then ROA =0 (or better e.g.: $1.0E-4 = \text{ROA} \ll 1$)

Note: ROA affects evaluation of cell volumes.

Note also: in case of geometry and trajectory plots (input block 11), this major radius offset ROA of poloidal cross sections has to be taken into account when defining plot-frames.

(Defaults: NLTRA = FALSE, ZIA = 0, ZAA = 1)

*Data for mesh multiplication*

**NLMLT** the complete "Standard Mesh" data are copied NBMLT times

**NBMLT** Number of identical copies of the standard mesh. Transition from one such mesh (called "block" in EIRENE) into another one has to be defined by transparent additional surfaces (see block 3B)

**VOLCOR** The volumes of all cells of the standard mesh as computed by EIRENE (in subroutine. VOLUME) are multiplied by one common factor VOLCOR for each "block".

*Data for additional zones outside the Standard mesh*

**NLADD** There are cells in the computational volume, which are defined through "additional surfaces" as cell boundaries. E.g. a standard mesh (if there is one) is augmented by "additional cells" in this case. If NLADD = FALSE, the complete block 2D may be omitted and the defaults are used.

**NRADD** Number of additional zones. The cell indexing along test flights in these zones has to be specified explicitly by making use of the ILSWCH, ILCELL parameters (block 3B)

(Default: NRADD = 0 ).

**VOLADD** Volume ($cm^{-3}$) of each additional zone as seen by the test-particles.

### 2.2.1 Mesh Parameters

As stated above, the computational volume in an EIRENE run is subdivided into NSBOX cells,
NSBOX = (NR1ST · NP2ND · NT3RD) · NBMLT + NRADD .
The can mesh consist two different parts.
The first is the so called "standard mesh" part ("VOXEL-geometry"), defined by the grids RSURF, PSURF and TSURF. There may then be NBMLT copies of that mesh, separated by additional surfaces (see below, section 2.3.2 (default: NBMLT=1).
The second part is the so called "additional cell region". Up to NRADD additional cells, of arbitrary shape, are defined by their boundaries ("**BREP!**"-geometry): the "additional surfaces" defined in section 2.3.2.

NR1ST is the number of surfaces in the $1^{st}$ (radial -or x-) grid. There are NR1STM = NR1ST - 1 cells.
The cell index of a particular cell is called NRCELL.
$1 \leq$ NRCELL $\leq$ NR1STM
NP2ND is the number of surfaces in the $2^{nd}$ (poloidal -or y-) grid. There are NP2NDM = NP2ND - 1 cells.
The cell index of a particular cell is called NPCELL.

1≤ NPCELL ≤ NP2NDM
NT3RD is the number of surfaces in the 2RD (toroidal -or z-) grid. There are NT3RDM = NT3RD - 1 cells.
The cell index of a particular cell is called NTCELL.
1≤ NTCELL ≤ NT3RDM
For the cells of the standard mesh, NACELL = 0
For the cells in the "additional cell region" :
1≤ NACELL ≤ NRADD
and
NRCELL = 0, NPCELL = 1, NTCELL = 1

The last radial (or x-) cell (number NR1ST), for any NPCELL, NTCELL inside the standard mesh, is not a real geometrical cell. It is used to store the radial (or x-) average of the volume averaged tallies.
Likewise, the last poloidal (or y-) cell (number NP2ND) is not a real geometrical cell, but used to store the poloidal (or y-) average of the volume averaged tallies (for any NRCELL, NTCELL inside the standard mesh)
NTCELL = NT3RD: analogically.

A particular cell can be specified in one of two possible ways:
Either by giving the 5 cell numbers:
NRCELL, NPCELL, NTCELL, NBLOCK, NACELL
or, alternatively, by giving the position in the 1-dimensional cell array
NCELL

The relation between these two options is:
NCELL = NRCELL + ((NPECLL-1)*NR1P2) + (NTCELL-1))*NP2T3 + NBLCKA
with
NBLCKA = (NBLOCK -1) * NSTRDT +NRADD

### 2.2.2 geometry level LEVGEO: symmetry and dimensionality of a run

tbd: add a few figures here. . .
The discretization of a domain is achieved by defining surfaces. These can be sets of coordinate surfaces $u^i(x, y, z) = const_i$ (then producing a discretization of the coordinate $\nabla u$).

**LEVGEO**

= 1 Simple Cartesian grids (1D, 2D or 3D) are used for code development, comparison with simple (semi-analytical) transport models, general 1D "one speed transport theory", validation of diffusion approximations, etc. The co-ordinates in the 3 directions are referred to as x,y and either z or phi, respectively, the latter depending on whether the third co-ordinate is linear (Cartesian) or circular.

= 2 Typically used in connection with 1D radial core transport models (e.g., "Duechs code", 1982 TRANSP 1987, RITM, (1996), . . . ), circular limiter tokamaks, or for simple parametric studies in toroidal or cylindrical coordinates (this is the oldest EIRENE geometry option, available since 1980). Shifted elliptical grid surfaces (1D), but also in combination with a polar grid (then: 2D) and/or a cylindrical or toroidal resolution in the third dimension.

= 3 This 2D discretization of a domain, based upon polygonal lines, in the x-y plane was originally developed for the B2-EIRENE interfacing. More generally, this option allows to directly use many 2D **CFD!** meshes directly in EIRENE for kinetic transport.

= 4 This example shows the extension of the 2D B2-grid, polynomial option LEVGEO = 3, into the outer "vacuum"-region by utilizing the LEVGEO = 4 option (plenum, vacuum chamber, grid refinement)

= 5 general 3D grid based upon tetrahedrons

= 10 user supplied geometry option. All grid relevant computations are done in user specified (problem specific) routines: EIRENE module: USER.f

## 2.3 The Input Block for Surfaces

Grid surfaces may be assigned special properties (e.g.: reflecting, absorbing, periodicity, modified cell index switching, etc.). Their definition is described in section 2.3.1.
In addition to these grid surfaces also additional surfaces can be seen by the histories (vacuum boundaries, special surfaces for scoring fluxes, diagnostic surfaces, . . . ). Such surfaces can be general linear or second order surfaces in 3D space. Their definition and properties are described in section 2.3.2.

### 2.3.1 The Input Block for "Non-default Standard Surfaces"

**General remarks**

Some of the "standard grid surfaces" RSURF, PSURF and TSURF may be defined as reflecting or absorbing (rather than the default transparent character of co-ordinate surfaces), or as transparent but with other switching features (cell indexing, transition into additional cell region etc.) than the default settings. E.g., surface averaged tallies (see block 11 (section 2.11), end of section 3.2 and table 6.4) are only evaluated on either "Additional Surfaces" (section 2.3.2) or on these "Non default Standard Surfaces", whereas they are not computed for those standard grid surfaces which are not explicitly identified in this block.
The geometrical properties of these "Non default Standard Surfaces" are specified here. This is described below.
**Particle-surface interaction models**
Although the (local) data for particle-surface interaction models (PSI-models) for each specific surface can be read in this input block, their meaning is described in block 6 (section 2.6) together with the globally valid input data for particle-surface interactions.
In any particular run there are NLIM "additional surfaces", see section 2.3.2. For many arrays containing surface properties the "non-default standard surfaces" are stored after the additional surfaces. E.g. FLAG(NLIM+4) would store the data FLAG for non-default standard surface no. 4.

```
*** 3A. Data for "Non-default Standard Surfaces"
      NSTSI
      DO 31 ISTSI=1,NSTSI
```

unless otherwise stated, N = NLIM+ISTSI is the index of the following surface data for surface ISTSI:

```
        TXTSFL
        ISTS IDIMP INUMP(ISTSI,IDIMP)
     .    IRPTA1 IRPTE1 IRPTA2 IRPTE2 IRPTA3 IRPTE3
        ILIIN ILSIDE ILSWCH ILEQUI ILTOR ILCOL
     .    ILFIT ILCELL ILBOX ILPLG
```

optional for non transparent surfaces (ILIIN > 0):
The next 3 (or 4) lines comprise the block for local particle-surface interaction data (in analytic terminology: the boundary condition at this surface element). If they are omitted, the default particle-surface model is activated for this particular surface element, see section 2.6.

```
      ILREF ILSPT ISRS  ISRC
      ZNML EWALL EWBIN TRANSP(1,N) TRANSP(2,N) FSHEAT
      RECYCF RECYCT RECPRM EXPPL EXPEL EXPIL
C (this line may be omitted, then: default sputter model,
C  see \cref{sec2.6})
      RECYCS RECYCC SPTPRM
   31 CONTINUE
```

also, optional for non-transparent surfaces (ILIIN > 0), and alternative to the local surface interaction data block mentioned above:
read a surface interaction model identifier to link one of the surface "local reflection models" defined in block 6 (section 2.6) below to this current surface. The presence of such a link is identified by the string SURFMOD_, followed by a name modname. Later, in input block 6 (section 2.6), a "local reflection model" block with that name modname must be included. This option allows quick changes of particle-surface interaction parameters, affecting many surfaces at once, by changes in just one location of the input file.

SURFMOD_modname

**Meaning of the Input Variables for "non-default standard surfaces"**

**NSTSI** Total number of non-default standard surfaces that do not act as prescribed by the default transparent standard co-ordinate surface model.

**TXTSFL** Text to characterize a surface ("name of the surface") on the printout file.

**ISTS** irrelevant; labelling index

**IDIMP** flag to identify mesh from which this particular surface is chosen.

= 1 surface from the x- (radial) standard mesh (RSURF) Note that for the unstructured grid options NLTRI and NLTET (i.e. for the 2D triangular grid option or for the general 3D grids of tetrahedra) all surfaces are referred to as 1st grid (x or radial) surfaces, by abuse of language.

= 2 surface from the y- (poloidal) standard mesh (PSURF)

= 3 surface from the z- (toroidal) standard mesh (TSURF)

**INUMP** Number of the surface in mesh RSURF, PSURF or TSURF respectively

**IRPTA,IRPTE**

Only a subregion of the surface acts by the "non-default options" specified for this particular surface. This subregion is defined by these flags.

If JMP is a surface from the first mesh, then IRPTA2→IRPTE2 and IRPTA3→IRPTE3 are the surface index ranges of the 2nd and 3rd mesh, respectively, for which this surface acts as non-default surface. IRPTA1 and IRPTE1 are irrelevant.

If JMP is a surface from the 2nd mesh, then IRPTA1→IRPTE1 and IRPTA3→IRPTE3 are the surface index ranges of the 1st and 3rd mesh, respectively, for which this surface acts as non-default surface. IRPTA2 and IRPTE2 are irrelevant.

If JMP is a surface from the 3rd mesh, then IRPTA1→ IRPTE1 and IRPTA2→ IRPTE2 are the surface index ranges of the 1st and 2nd mesh, respectively, for which this surface acts as non-default surface. IRPTA3 and IRPTE3 are irrelevant.

The third card ILIIN, . . . , ILPLG is identical to the corresponding card for "additional surfaces", see block 3b (section 2.3.2) below, "Input Data for Additional Surfaces". One exception is the flag ILSIDE, which controls the sign of the surface normal vector (hence the orientation of a surface). In case of unstructured "standard grids" NLTRI or NLTET (triangles in 2D and tetrahedrons in 3D) there is no well defined default surface orientation and the flag ILSIDE is irrelevant in such cases.

The input data in the block for the local reflection model are described below, see section 2.6 : "Input Data for Particle-Surface Interaction Models".

### 2.3.2 Input Data for "Additional Surfaces"

**General remarks**

Internally each additional surface is defined by an algebraic equation and some algebraic inequalities specifying the boundary of that surface, i.e., that part of the surface which is seen by the test particles. By changing the sign of all coefficients in the algebraic equation the orientation of the surface normal vectors are reversed.
The intersection with the nearest surface along a trajectory is found by checking surfaces in the sequence of their input (in subroutine TIMEA of the geometry block GEO3D). This must be taken into consideration, if there are two parts of one surface specified by different boundaries and with non-empty intersection. In this case always the surface later in the input sequence is seen by the test particles.
One can define two distinct plane surfaces as one surface of second order, provided that this is compatible with the options available for surface boundaries.
We will refer to positive and negative directions of flights for particles intersecting a surface. By "positive" it is meant that the angle between the trajectory of the particle and the surface normal at the point of intersection is less than 90 degrees, "negative" has the opposite meaning.
**speeding up geometry calculations:**
For optimization of CPU performance the checking of intersections with surfaces along a test flight can be abandoned depending upon the current position of a test particle.
For particles in cell no. NCELL all surfaces MSURF with IGJUM3(NCELL, MSURF) = 1 are not checked. For particles on surface MS all other surfaces MSURF with IGJUM1(MS, MSURF) = 1 are not checked.
The integer matrix IGJUM1 can be set either in this block, see CH1 cards described below, or in some user-specified problem specific routines (e.g. GEOUSR). The integer matrix IGJUM3 can be set via the CH3 cards in input block 8, or, again, e.g. in problem specific routines such as GEOUSR.
**ignorable coordinates:**
Surfaces with ignorable coordinates are specified either by setting the corresponding coefficients in the surface equation to zero, or, in case of 2-point, 3-point, 4-point of 5-point options, by setting the corresponding coordinates of the points to -1.D20 and +1.D20, respectively.
**coordinate systems, transformations:**
All surfaces are specified in Cartesian coordinates.
In case of NLTRA (toroidal geometry approximation), surfaces must be defined in the local coordinate system of toroidal cells centered at the "magnetic axis", (xa,ya) = (RMTOR,0.), i.e., excluding the major radius of the torus.
To facilitate input of the geometrical data, each single (or a set of) surface(s) can be transformed by a Cartesian mapping after specification of the surface coefficients and boundaries in some "convenient" coordinate frame, by including "TRANSFORM cards" at the end of an input block for a particular surface. One such "TRANSFORM-deck" can act on an entire range of surfaces, but only on those which have been read previous to the "TRANSFORM-deck". Hence, in order to transform the entire set of additional surfaces by one deck, this deck must come after the last surface (no. NLIMI, see below).
**Particle-surface interaction models**
Although the (local) data for particle surface interaction models for each specific additional

surface can be read in this input block, their meaning is described in block 6 (section 2.6) together with the globally valid input data for particle surface interactions.

**The Input Block**

```
*** 3B. Data for Additional Surfaces
      NLIMI
C     ''CH0-cards" (may be omitted)
C     (format: 3A,69A)
```

arbitrary number of strings $\pm n/m$, separated by blanks. n and m must be integer variables $1 \le n, m \le NLIMI$.

```
      DO  ILIMI=1,NLIMI
        TXTSFL
C       ''CH1-cards" (may be omitted)
C       (format: 3A,69A)
```

arbitrary number of strings $\pm n/m$, separated by blanks. n and m must be integer variables $1 \le n, m \le NLIMI$.

```
C       ''CH2-cards" (may be omitted)
C       (format: 3A,69A)
```

arbitrary number of strings ± n/m, separated by blanks. n and m must be integer variables $1 \le n, m \le NLIMI$.

```
        RLBND  RLARE  RLWMN  RLWMX
        ILIIN ILSIDE ILSWCH ILEQUI ILTOR ILCOL ILFIT ILCELL
     .    ILBOX ILPLG
        IF (RLBND.LT.0.) THEN
          A0LM  A1LM  A2LM  A3LM  A4LM  A5LM
          A6LM  A7LM  A8LM  A9LM
```

let RLBND = -KL, then first read L cards:

```
          ALIMS  XLIMS  YLIMS  ZLIMS
```

and then read K blocks next:

```
          ALIMS0  XLIMS1  YLIMS1  ZLIMS1  XLIMS2  YLIMS2
          ZLIMS2  XLIMS3  YLIMS3  ZLIMS3
        ENDIF
        IF (RLBND.EQ.0.) THEN
          A0LM  A1LM  A2LM  A3LM  A4LM  A5LM
          A6LM  A7LM  A8LM  A9LM
        ENDIF
        IF (RLBND.GT.0..AND.RLBND.LT.2.) THEN
          A0LM  A1LM  A2LM  A3LM  A4LM  A5LM
          A6LM  A7LM  A8LM  A9LM
          XLIMS1  YLIMS1  ZLIMS1  XLIMS2  YLIMS2  ZLIMS2
        ENDIF
        IF (RLBND.GE.2.) THEN
```

```
        P1(1,..)  P1(2,..)  P1(3,..)  P2(1,..)  P2(2,..)
     .    P2(3,..)
        IF (K .GT. 2) THEN
          P3(1,..)  P3(2,..)  P3(3,..)  P4(1,..)  P4(2,..)
     .      P4(3,..)
          IF (K .GT. 4) THEN
            P5(1,..)  P5(2,..)  P5(3,..)
          ENDIF
        ENDIF
      ENDIF
```

optional for non transparent surfaces (ILIIN > 0):
The next 3 (or 4) lines comprise the block for local particle-surface interaction data (in analytic terminology: the boundary condition at this surface element). If they are omitted, the default particle-surface model is activated for this particular surface element, see section 2.6.

```
      ILREF  ILSPT ISRS  ISRC
      ZNML EWALL EWBIN TRANSP(1,N) TRANSP(2,N) FSHEAT
      RECYCF  RECYCT  RECPRM  EXPPL  EXPEL  EXPIL
C (following line may be omitted, then: default sputter
C  model, see \cref{sec2.6})
      RECYCS  RECYCC  SPTPRM
```

also optional for non transparent surfaces (ILIIN > 0):
read a surface interaction model identifier to link one of the surface "local reflection models" defined in block 6 (section 2.6) below to this current surface. The presence of such a link is identified by the string SURFMOD_, followed by a name modname. Later, in input block 6 (section 2.6), a "local reflection model" block with that name modname must be included. This option allows quick changes of particle-surface interaction parameters, affecting many surfaces at once, by changes in just one location of the input file.

```
      SURFMOD_modname
```

optional: read one or several blocks of five cards each for orthogonal mapping. The presence of each such block is identified by the first card of that block containing the string *'TRANSFORM'* followed by the other four cards:

```
      TRANSFORM
      ITINI,ITEND
      XLCOR,YLCOR,ZLCOR
      XLREF,YLREF,ZLREF
      XLROT,YLROT,ZLROT,ALROT
```

```
C (end of ILIMI loop)
    ENDDO
```

**Meaning of the Input Variables for additional surfaces**

**CH0 n1/m1 n2/m2 ...** surfaces from the range n1 to m1, n2 to m2, . . . , are ignored by EIRENE. CH0 is indicated as "CH0-LINES" in JSON input.

Specifying a surface in such a CH0-card is identical to taking it out from the input file. It may be more convenient in some cases, however, to use the CH0 option, because the labelling index of the remaining valid surfaces is not altered then. No input is read for these surfaces, and the input segment for the next valid surface (identified by the string '*text') is read next.

**NLIMI** number of surfaces in the input block

**TXTSUR** Text to identify a surface ("name of the surface") on the printout file

**CH1(ILIMI) n1/m1 n2/m2 ...** surfaces from the range n1 to m1, n2 to m2, ..., are considered invisible for a particle located on this current surface ILIMI. Intersection of trajectories starting from surface ILIMI with those "invisible" surfaces is not checked.
CH1 is indicated as "CH1-LINES" in JSON input

**CH2(ILIMI) n1/m1 n2/m2 ...** Only for second order surfaces ILIMI, n1-m1, ... The first of the two possible intersections is ignored for particles located on surface no. ILIMI.
CH2 is indicated as "CH2-LINES" in JSON input

*General Data for Surfaces*

**RLBND** flag for different options to define the boundary of the surface

**RLARE** Area (in $cm^2$) of the surface element which is seen by the test particles. (Default: 666.0) (needed only for scaling of non default surface averaged tallies) If RLARE is not specified here, (i.e., if a value less than or equal to zero is read) then EIRENE tries to evaluate this area itself. For some surfaces this is still not possible automatically.

**RLWMN** lower weight limit for space weight window for particles crossing the surface in positive direction. (not in use)

**RLWMX** upper weight limit for space weight window for particles crossing the surface in positive direction. (not in use)

**ILIIN** defines the type of surface

$> 0$ non-transparent surface

$= 1$ reflecting, partly or purely absorbing surface.

local reflection model has to be specified unless default model is to be used; all surface tallies (see table 6.4) are updated and a switch can be operated.

$= 2$ purely absorbing surface ;. surface tallies for incident fluxes (i.e., POT... and EOT... tallies in table 6.4) are updated and the particle history is stopped then.

$= 3$ mirror for incident test particles. I.e., specular reflection for neutral test particles, and for charged test particles the sign of the velocity component parallel to the B-field is reversed.

= 4 periodicity surface, with regard to x, y, or z coordinate, depending upon whether this surface is a standard x, y, or z grid surface, respectively. Move particle to x / radial grid surface no. m, m integer (or to y / poloidal or to z / toroidal surface no. m, respectively) and continue track from there with otherwise identical particle parameters.

This option is currently implemented only for Cartesian grids (NLSLB and NLTRZ) and for non-default standard grid surfaces only.

The periodicity options are not fully implemented yet. Please contact the author for the current status of your particular version.

$\leq 0$ transparent surface (for example: hole in one of the other "additional surfaces"). Particle and energy fluxes onto and from these surfaces do not contribute to global balances.

= 0 Particle history is not interrupted

No surface tallies are updated, no switches can be operated. Fastest option.

$= -1$ Particle history will be stopped and restarted. A switch can be operated. I.e., this surface is used only for switching (see below: ILSWCH) or re-initializing the particle's track at the point of intersection. No surface tallies are updated.

$= -2$ as -1, and, additionally:

if a particle is crossing the surface in the positive direction, (one sided-) surface tallies are updated, e.g., by default: partial particle and energy currents $J^+$ (amp) and $K^+$ (watt). These are stored in the POT... and EOT... tallies of table 6.4. If the particle crosses the surface in the direction opposite to the surface normal, then negative partial particle and energy currents $J^-$ (amp) and $K^-$ (watt) are updated. These are stored in the PRF... and ERF... tallies of table 6.4.

$= -3$ Net currents (e.g. $J^+ - J^-$), are evaluated, and stored on the POT... and EOT... tallies (see table 6.4). The PRF... and ERF... tallies are empty for these surfaces.

$\leq -4$ Not in use. Currently: same as ILIIN=-2 option.

**ILSIDE**

= 0 both sides of the surface act as described by ILIIN option (default).

= 1 particles incident on the surface in the negative direction will be absorbed (i.e., ILIIN = 2 option from that side).

= 2 particles incident on the surface in the negative direction will be killed and the message

"ERROR IN ADDCOL"

or

"ERROR IN STDCOL"

will be printed. The contribution of these particles to the particle- and energy flux balances will be called PTRASH and ETRASH respectively. This option should be used for geometry testing whenever the user expects particles incident only from one side.

= 3 particles incident on the surface in the negative direction will not see the surface, i.e., this surface acts like a (semi) transparent surface (ILIIN = 0 option) from that side.

= −1 as 1, but with the opposite direction of the surface normal

= −2 as 2, but with the opposite direction of the surface normal

= −3 as 3, but with the opposite direction of the surface normal

**ILSWCH** = IJKLMN, i.e. six digits I, J, K, L, M and N

= 0 no switch is operated

**N** EIRENE flag ITIME

= 1 The calculation of the step sizes in the standard mesh is abandoned for a particle which crosses the surface in the positive direction, and is reactivated, if the particle strikes in the negative direction

= 2 as 1, but with the direction of the surface normal reversed for this option.

**M** EIRENE flag IFPATH

= 1 Abandon the calculation of the collision rates (entry into the vacuum) for a particle which is striking the surface in the direction of the surface normal. For particles incident from the other direction, evaluation of collision rates is reactivated.

= 2 as 1, but with the direction of the surface normal reversed.

**L** EIRENE flag IUPDTE

= 1 Abandon the updating of volume-averaged tallies for a particle which is striking the surface in the direction of the surface normal. For particles incident from the other direction, updating of volume averaged tallies is reactivated.

= 2 as 1, but with the direction of the surface normal reversed for this option.

I,J,K flags for switching cell numbers at transition into a different mesh cell.

**K** for particles in an additional cell, i.e., not in one of the "standard mesh" blocks:

= 1 Increase the actual additional cell number NACELL for a particle striking the surface in the direction of the surface normal by ILACLL. Decrease NACELL by ILACLL if the particle is striking in the negative direction. Specification of ILACLL is via the input variable ILCELL, see below.

= 2 as K = 1, but with the direction of the surface normal reversed for this option.

for particles inside the "standard mesh ", i.e., not in the "additional cell region"

= 1 Increase the standard mesh block number NBLOCK for a particle striking the surface in the direction of the surface normal by ILBLCK. Decrease NBLOCK by ILBLCK if the particle is striking in the negative direction. Specification of ILBLCK is via the input variable ILCELL, see below.

= 2 as K = 1, but with the direction of the surface normal reversed.

**J** for particles at the boundary between "additional" and "standard" mesh regions.

= 1 entrance into standard mesh, block no.

NBLOCK = ILBLCK

or exit from standard mesh into additional cell

NACELL = ILACLL.

Specification of ILACLL and ILBLCK is via the input variable ILCELL, see below. If ILACLL = 0, then no switch to additional cell is operated. (E.g.: for surfaces which are reflecting from this side).

= 2 as J = 1. The direction of the surface normal does not matter here.

**I** similar to J-flag, i.e., for transitions between standard and additional meshes, but different cell number switching.

= 1 Entrance into standard mesh, block no.

NBLOCK = NACELL+ILBLCK,

if the particle is striking in the positive direction, or

NBLOCK = NACELL-ILBLCK,

if the particle is striking in the negative direction. Exit from standard mesh into additional cell

NACELL = NBLOCK+ILACLL,

for a particle striking the surface in the positive direction, or

NACELL = NBLOCK-ILACLL,

if the particle is striking in the negative direction.

Specification of ILACLL and ILBLCK is via the input variable ILCELL, see below.

= 2 as I = 1, but with the direction of the surface normal reversed.

If a test particle history starts from a surface (NLSRF option), then ILSWCH acts as if this particle had struck the surface prior to the birth process in the positive direction. This default setting is only available for ILSIDE $\neq 0$ and can (must) be overruled by the SORIFL flag (see section 2.7), e.g. if a surface source needs to be defined on a surface with ILSIDE = 0.

**ILEQUI** The algebraic equations for the surfaces J and IABS(ILEQUI(J)) will be described by exactly the same coefficients (up to a common sign, if ILEQUI(J) .lt. 0). For example a triangle can be specified by the three corners and another part of the same plane surface can be specified directly by its algebraic coefficients. To avoid round-off errors one should use the ILEQUI option in such cases, in particular if surface J is a transparent "hole" in surface ILEQUI(J), or vice versa.

Default: ILEQUI = 0

**ILTOR** For NLTRA option only (see block 2c):

if ILTOR>0 :

the surface is defined with respect to the local coordinate system of the toroidal segment with "cell-number" ILTOR, hence: 1 ≤ILTOR ≤NTTRAM.

if ILTOR=0 :

the surface is defined with respect to any local coordinate system. I.e., the surface equations are taken to be the same in each local system.

If the surface equations are z-independent, then this surface is toroidally symmetric (within the NLTRA-approximation).

Otherwise the surface has NTTRAM-fold periodicity.

This flag is irrelevant for NLTRZ (i.e., if cylindrical coordinates are used) or for NLTRT.

Default: ILTOR = 0

**ILCOL** Flag for the color that is used for plotting this surface on 2d or 3d geometry plots. If ILCOL ≤ 0 than -ILCOL is used and the surface area is filled in by that color on the 3d geometry plots.

Default: ILCOL=1

**ILFIT** = MN

This option is relevant only for surfaces with one ignorable coordinate, i.e. it only works for the $2. \leq RLBND \leq 3.$ surface boundary options.

It is a tool to facilitate a neat fitting of surfaces, in particular for connecting curved and plane surfaces (i.e. to avoid particle leakage due to numerical round-off errors in the algebraic surface coefficients.

M and N (3 digits each, M may be omitted if not needed) are the numbers of the surfaces (which must have the same ignorable coordinate) the boundaries of which should match to those of the actual surface J. The boundaries of these neighboring surfaces M and N must be specified by the RLBND = 1 or RLBND = 1.5 option.

The fitting is achieved by a small automatic internal modification of the surface data P1 and/or P2 of surface number J in subroutine SETFIT. Printout of the modifications made there is activated with the TRCSUR flag (input block 11).

**ILCELL** Parameter ILBLCK and ILACLL for the ILSWCH flags described above. Let ILCELL = NM, with N and M being integers with 3 digits each. Then N = ILBLCK and M = ILACLL.

**ILBOX** to be written

**ILPLG** EIRENE can write out information for a finite element mesh generator to produce a grid of triangles for a multiply connected 2D domain with cracks and holes. The various (inner and outer) boundaries are given as polygonal lines, which are composed of selected standard grid surface segments (NLPLG option) and/or additional surfaces (2 ≤ RLBND < 3 option).

This flag identifies closed polygonal lines composed of additional surfaces given by the 2-point option and/or of standard surfaces in the x-y-plane. For example if ILPLG(I) =

NN, for surfaces I = I1, I2, . . . , IN, (NN a positive integer) then these IN surfaces form a closed polygonal line in the x-y-plane. The region inside this closed line is part of the computational domain. By a negative integer value of NN a closed polygonal region can be excluded from the computational domain, i.e., a hole in the domain is specified by these surfaces. EIRENE writes an output file appropriate for a finite element mesh generator (available from FZ-Jülich) to produce a triangular discretization of the resulting (possibly multiply connected) domain. This option can be used to discretize arbitrarily complex 2D domains with internal and external boundaries given by the additional or non-default standard surfaces.

These finite element grids can be combined with the regular grids by using the problem specific geometry routines (see section 3) or the code interfacing routines INFCOP (see section 4).

**Surface coefficients and boundaries of surfaces**

The surface equation:

$$\begin{aligned} &A0LM + A1LM \cdot x + A2LM \cdot y + A3LM \cdot z + \\ &\quad + A4LM \cdot x^2 + A5LM \cdot y^2 + A6LM \cdot z^2 + \\ &\quad + A7LM \cdot xy + A8LM \cdot xz + A9LM \cdot yz = 0 \end{aligned} \tag{2.1}$$

The positive surface normal $(n_x, n_y, n_z)$ depends upon the point of impact (x,y,z) and is defined by the vector

$$\begin{aligned} n_x &= A1LM + 2 \cdot A4LM \cdot x + A7LM \cdot y + A8LM \cdot z \\ n_y &= A2LM + A7LM \cdot x + 2 \cdot A5LM \cdot y + A9LM \cdot z \\ n_z &= A3LM + A8LM \cdot x + A9LM \cdot y + 2 \cdot A6LM \cdot z \end{aligned}$$

For a plane surface the following reduction is valid:
$(n_x, n_y, n_z)$ = (A1LM,A2LM,A3LM)

The boundary of the valid part of the surface may be described by 4 different options, depending upon the value of the flag RLBND.

**RLBND = 0**

No boundary inequalities specified, i.e. the whole surface is seen by the test particles.

**0 < RLBND < 2**

= 1 Only that part of the surface, which lies inside the right parallelepiped defined by the two vectors (XLIMS1, YLIMS1, ZLIMS1) and (XLIMS2, YLIMS2, ZLIMS2), is seen by the particles. I.e. the three inequalities

$XLIMS1 \le x \le XLIMS2$

$YLIMS1 \le y \le YLIMS2$

$ZLIMS1 \le z \le ZLIMS2$

are checked at the point of intersection (x,y,z).

= 1.5 Complement to RLBND = 1.

Only the surface element outside the parallelepiped is seen by the particles.

**RLBND $\geq$ 2**

In this case the surface will be defined by the input of the coordinates of at least 2 and at highest 5 points on a plane surface. If there are only 2 points, the surface is parallel to one axis. If there are 3 or more points, then the boundary of this plane surface is a closed polygon $(P_1, \ldots, P_n, P_1)$. Therefore, the correct order of points at input is relevant. The orientation of the positive surface normal vector is defined by the first 3 points, and it is given by the vector product $(P3 - P1) \times (P3 - P2)$. Thus, the orientation can be reversed e.g. by interchanging $P_2$ and $P_3$.

= 2.1 plane surface parallel to z axis. The surface equation of this plane reads ax + by + c = 0 with the coefficients a,b and c such that the points $P_1, P_2$ lie on this surface and the valid part of that surface ranges from $P_1$ to $P_2$ in the xy-plane. The z-coordinates of these two points define the boundaries in z direction

= 2.2 Complement to RLBND = 2.1

= 2.4 as RLBND=2.1 option, but with z and y exchanged. I.e., now the y coordinates of the points $P_1, P_2$ are the boundaries of the surface ax+bz+c=0 in y direction.

= 2.5 Complement to RLBND = 2.4

= 2.7 as RLBND=2.1 option, but with z and x exchanged. I.e., now the x coordinates of the points $P_1, P_2$ are the boundaries of the surface ay+bz+c+0 in x direction.

= 2.8 Complement to RLBND = 2.7

= 3 plane triangle defined by the corners $P_1, P_2, P_3$

$P_1$=(P1(1),P1(2),P1(3))

$P_2$=(P2(1),P2(2),P2(3))

$P_3$=(P3(1),P3(2),P3(3))

= 3.5 complement to RLBND = 3; only The plane surface outside the triangle is seen by the test particles.

= 4 plane quadrangle; surface inside the polygon

$(P_1, P_2, P_4, P_3, P_1)$.

Here $P_1, P_2, P_3$ are as in the RLBND=3 option, and $P_4$ = (P4(1), P4(2), P4(3))

Thus this surface is the union of the triangles with vertices $P_1$, $P_2$, $P_3$ and $P_2$, $P_4$, $P_3$ respectively.

= 4.5 complement to RLBND = 4; only the part of the plane surface outside the quadrangle is seen by the test particles

= 5 plane quint-angle; surface inside the polygon $(P_1, P_2, P_4, P_5, P_3, P_1)$ $P_1$, $P_2$, $P_3$, $P_4$ as RLBND=4, and $P_5$ = (P5(1), P5(2), P5(3))

= 5.5 complement to RLBND = 5; only the part of the plane surface outside the quint-angle is seen by the test particles.

**RLBND ¡ 0**

-KL

The surface is bounded by L linear inequalities and by K second order inequalities.

$$ALIMS + XLIMS * x + YLIMS * y + ZLIMS * z \leq 0 \text{ (L inequalities)}$$

$$\begin{aligned} ALIMS0 + XLIMS1 \cdot x + YLIMS1 \cdot y + ZLIMS1 \cdot z \\ + XLIMS2 \cdot x^2 + YLIMS2 \cdot y^2 + ZLIMS2 \cdot z^2 \\ + XLIMS3 \cdot xy + YLIMS3 \cdot xz + ZLIMS3 \cdot yz \leq 0 \end{aligned}$$

(K inequalities)

The meanings of the input variables for the local reflection model are described below (see: section 2.6: "Input Data for Surface Interaction Models").
The meanings of the input variables for a transformation block are:

**ITINI, ITEND**

The transformation defined by the next 3 cards is carried out for additional surfaces from number ITINI through ITEND The transformation is carried out as soon as this transformation deck is found. Hence, if such a transformation deck is found after "additional surface" no. IS, then one must guarantee: ITINI ≤ IS and ITEND ≤ IS.

**XLCOR, YLCOR, ZLCOR**

Translation:

The origin of the coordinate system is shifted by the vector

XLCOR,YLCOR,ZLCOR

**XLREF, YLREF, ZLREF**

Reflection:

to be written

No transformation if XLREF,YLREF,ZLREF = (0,0,0)

**XLREF, YLREF, ZLREF**

Rotation:

The vector XLROT,YLROT,ZLROT defines the axis of rotation.

ALROT is the angle of rotation (in degrees)

No transformation if ALROT=0 or axis of rotation= (0,0,0)

## 2.4 Input Data for Species Specification and Atomic Physics Module

**General remarks**

EIRENE can handle up to NATM "atom" species, NMOL "molecule" species, NION "test ion" species, NPHOT "photon" species (lines) and NREAC different atomic, molecular or photonic reactions between these "test particles" and the "bulk ions" or electrons. There may be up to NPLS "bulk ion" species, and one electron gas derived internally from the assumption of local charge neutrality. Amongst the heavy background particles (the "bulk ions") may be species with charge state zero, i.e., neutral particles. By abuse of language, we refer to them as "bulk ions" as well, but we mean in this case: heavy background particles, i.e. more general objects.
NATM, NMOL, NION, NPHOT, NPLS and NREAC are specified in the parameter block "PARMUSR", see: "Parameter Statements" (section 3.1 below, for versions EIRENE$_{2000}$ and older), or are determined automatically from the input file.
Masses in EIRENE (RMASSA, RMASSM, RMASSI, RMASSP assigned to the objects "atoms", "molecules", "test ions", "photons" and "bulk ions", respectively) are in units of the proton mass, which is taken to be 1.0073 in atomic mass units.

At the beginning of input block 4 EIRENE first searches for the character string "INCLUDE" in the input card directly following the card

```
*** 4. DATA FOR .....
```

If such a character string is found here, then the rest of input blocks 4 and 5 is skipped and a proper "A&M model data file from a pre-compiled EIRENE AM-library can be used.
If such a library AM data set is not included, then the "original" EIRENE procedure is followed, to define general test particles (objects) (atoms, molecules, test ions, photons) and assign reactions (A&M data sets) to each of them from external or internal databases. This is described next:

First EIRENE reads NREACI "non-default" atomic data cards, (if any, i.e., if NREACI > 0)

```
      NREACI
      DO I=1,NREACI
        IR  FILNAM  H123  REAC  CRC  MASSP  MASST  DP RMN1
     .     RMX1 R2MN R2MX

C       (format: I3,1X,A6,1X,A4,Axxx,A3,2I3,3E12.4)
C   Since EIRENE tag Eirene_CR_models
C       (format: I3,1X,A8,1X,A4,Axxx,A3,2I3,3E12.4)
ENDDO
```

either from external A&M data files such as FILNAM = HYDHEL, = METHAN, = AMJUEL, = H2VIBR, for polynomial fits, or other data files of appropriate format.
RMN1, RMX1, R2MN, R2MX (unless all set to default, =0.0) define the validity range of the fit, or table, and additional cards containing information on the proper asymptotic behavior are then read (see below).
Since Jan. 2018 and younger versions direct access to atomic data from internal CR codes (evaluated on a cell by cell basis) is re-activated by

IR CRM H123 REAC CRC MASSP MASST

*C (format: I3,1X,A8,1X,A4,Axxx,A3,2I3,E12.4)*

The character string 'CRM' (A8) identifies which particular internal CR code is executed, cell by cell, and whether it is run in meta-stable resolved or meta-stable unresolved mode. Currently H and He codes are available. Documentation: unfinished. Since EIRENE tag Eirene_CR_models optional input cards control further options, such as population escape factors (opacity effects). The corresponding input format is

IROW_ESC, ICOL_ESC, POP_ESC

*C (format: 2I6,E12.4)*

The meaning of input variables is the same as above. This option bypasses data tables or data fits for collision radiative model rate coefficients, cooling rates, or population coefficients. The internal CR code is executed cell by cell in the initialisation phase. In older versions of EIRENE the intrinsic hydrogen collision radiative code had to be invoked from special, problem specific routines, as e.g. done for non-linear coupling between gas and radiation field since about 2002 in a particular version of MODUSR.f. Parameter DP (potential energy shift) is currently not available here.
For tabulated (rather than fitted) collision reaction rates, electron cooling rates, etc., with a format derived from the so called "**ADAS!** database format AFD11", or any other properly (2D) tabulated data source, i.e. with FILNAM = TAB2D, or FILNAM = ADAS the corresponding card reads:

IR TAB2D H123 REAC CRC MASSP MASST DP

*C (format: I3,1X,A6,1X,A4,Axxx,A3,2I3,E12.4)*

or, alternatively, if **ADAS!** ADF11 files are used directly:

IR ADAS H123 REAC CRC MASSP MASST DP

*C (format: I3,1X,A6,1X,A4,Axxx,A3,2I3,E12.4)*
ELNAME, IZ

*C (format: 4X,A2,1X,I3)*

In this case an additional card is read (containing variables: ELNAME, IZ) for each IR entry, see below, to identify the species and ion charge state within the data file (e.g. within an **ADAS!** ADF11 file folder).
For photonic emission and absorption data (line shapes), i.e. for FILNAM = PHOTON, the line is shortened to

IR PHOTON H123 REAC

*C (format: I3,1X,A6,1X,A4,Axxx)*

for reading from the spectroscopic line-shape database PHOTON, but additional lines are needed to specify the line broadening options. See below.
There also exists an option for specifying constant (or very simple functional dependencies) cross sections, reaction rate coefficients, reaction rates, etc. directly via the input file, without resorting

to an external database. This has proven useful for testing purposes, e.g. comparison with simple analytic or numerical solutions. In this case: FILNAM = CONST, further details: see below.
Later EIRENE will assign the relevant atomic, molecular or photonic processes to each test particle (and also to bulk particles) from either this set of NREACI processes, or from a small set of internal "default processes", which are hard wired in EIRENE.
In versions younger than 2002 each reaction in these databases can contain a string 'fit-flag = value', with IFTFLG=value. IFTFLG is used in EIRENE to identify the type of fitting expression to be evaluated with the fitting coefficients from the database.
By default: IFTFLG=0 for all data, i.e., single or double polynomial fits.
In addition to the fit coefficients themselves also the validity range of the fits, and additional coefficients for asymptotic expressions beyond this range, can the read from the data files (if available) or can be specified explicitly for each set of data in the input file.
All atomic rate coefficients and cross sections can be scaled with a constant factor (FREAC, see below), for sensitivity studies.
Excluded from this scaling option are all photonic processes and those elastic collisions for which EIRENE uses an interaction potential (H.0-type data), because here the scaling would not have the expected effect on transport, but rather it would only modify the effective small angle scattering cut-off in the binary collisions.
Atomic and molecular processes are always of the following type:

$$a \cdot A + b \cdot B \longrightarrow m \cdot M + n \cdot N + \Delta E$$

(Very few exceptions, with 3 or more groups of secondaries (N, M, L, . . . ), exist and can also be treated, but these are very rare applications. See below.)
Here $A, B, M$ and $N$ are labels for the type of pre- and post-collision particles. $a, b, m$ and $n$ are the stoichiometric coefficients, and $\Delta E$ is the amount of internal energy transferred into (or at the expense of) the kinetic energy of the collision products $M$ and $N$.
The following conventions are always in use:
At least one of A, B, M or N must stand for a "test particle", for otherwise this process is not relevant for the transport problem solved by EIRENE.

- particles $A$ are **always** "bulk particles", i.e., from the list of "bulk ions" (input block 5) or electrons.
- particle $B$ may be one "test particle". Then this process $A + B \rightarrow \ldots$ must be included in the list of reaction decks for this test particle B. Hence: $b = 1$ then.
- particle $B$ may also be one "bulk particle". Then this process may be included in input block 7 as a volume-source (recombination of two bulk particles into at least one test particle $M$ and/or $N$,
- particles $M$ and/or $N$ can be either secondary "test particles" or secondary "bulk particles".
- non-linear processes, i.e., processes in which both A and B should, in principle, stand for test particles, can be included by specifying particle A both as bulk particle in input block 5 and, simultaneously, as test particle in input block 4. The parameters for the "artificial bulk species" A are iterated in a sequence of EIRENE runs (option: NITER, input block 1, and code module MODUSR.f). In cases in which species M and/or N are also available

as ("real" or "artificial") bulk species, there is the choice to specify secondaries from a reaction either as test-particles and to continue the history with those after a collision event, or, alternatively, to specify them as bulk particles, and stop the trajectory. In this latter case one must add a volume source to launch these "secondaries" via a proper spatially distributed source in the next iteration. By using this option carefully, and combining it with the stratified source sampling technique, coupling to external codes/models can be made either more or less "implicit".

Some care is needed here to avoid double-counting.

Next EIRENE expects one so called "species block" for each test particle species. Later, in input block 5, it will also expect one species block for each of the heavy background particles, i.e., for the "bulk ions".
A "species block" has the format:

```
      ISPZ$  TEXT$  NMASS$  NCHAR$  NPRT$  NCHRG$  ISRF$  ISRT$  ID1$
      NRC$   NFOL$  NGEN$   NHSTS$  ID3$
C     (format:  I2,1X,A8,1X,12(I2,1X))
```

Default for ID1 is: ID1=0 In case ID1 $\leq$ 2, i.e., two or less post-collision particle (not counting electrons)

```
      DO  K= 1, NRC$
        IREAC$   IBULK$   ISCD1$   ISCD2$   ISCDE$   IESTM$   IBGK$
        EELEC$   EBULK$   ESCD1$   EDUMMY   FREAC$   EDPOT$
      ENDDO
```

New option in 2006: In case ID1 = 3, i.e., three post-collision particles (not counting electrons), needed for some more complex chemical reactions, such as particle rearrangement collisions $p + CH_4 \rightarrow CH_2 + H_2 + H$, etc.

```
      DO  K= 1, NRC$
        IREAC$   IBULK$   ISCD1$   ISCD2$   ISCD3$  ISCDE$   IESTM$
     .    IBGK$
        EELEC$   EBULK$   ESCD1$   EDUMMY  FREAC$   EDPOT$
      ENDDO
```

New option in 2006:
In case ID1 = 4, i.e., four post-collision particles (not counting electrons)

```
      DO  K= 1, NRC$
        IREAC$   IBULK$   ISCD1$   ISCD2$   ISCD3$  ISCD4$  ISCDE$
     .    IESTM$   IBGK$
        EELEC$   EBULK$   ESCD1$   EDUMMY  FREAC$   EDPOT$
      ENDDO
```

where **$** stands for either A (atoms), M (molecules), I (test ions), PH (photons) or P (bulk ions). I.e., a species block consists of one species specification card ISPZ$ . . . , and NRC$ "reaction decks" of two cards each.
New option in 2023:
ID3 has beem removed and for molecules, test and bulk ions LKIND$ has taken its place

```
      ISPZ$ TEXT$ NMASS$ NCHAR$ NPRT$ NCHRG$ ISRF$ ISRT$ ID1$
      NRC$  NFOL$ NGEN$  NHSTS$ LKIND$
C     (format: I2,1X,A8,1X,12(I2,1X))
```

For some particle species (in particular for hydrogenic atoms, molecules and molecular ions, and for helium atoms) EIRENE has default A&M data, to which it resorts, if no reaction cards are in the input deck (i.e., if NRC$=0, see below) for a particular species. These default models are described at the end of this section. They are overruled by the reaction cards.
In order to de-activate all possible reactions (also the default reactions) for a particular test particle species, e.g., to simulate the collision-less "Knudsen flow" of a gas or optically thin line radiation, one must set NRC$ = −1 for this test particle species.

Usually, a particular particle type and species in the species blocks is identified by an integer IPART$=LMN (3 digits). Here N fixes the type of the particle, M is the number of particles characterized by IPART$ and L is the species index within the specified group (type) of particles. The following values of N are currently in use:

- N=0: photons (versions 2004 and younger)
- N=1: atoms
- N=2: molecules
- N=3: test ions
- N=4: bulk ions
- N=5: electrons

**The Input block**

```
*** 4. Data for Species Specification and Atomic Physics Module
```

Note: in EIRENE$_{2003}$ or later the format of input flag REAC described below was generalized from A9 to Axxx, with xxx ≤ 50, to accommodate the longer reaction information for photonic processes into the same format. The old format A9 is of course automatically still recognized as special case.

```
      NREACI
      DO
```

**for "real particles"**, databases: FILNAM = HYDHEL, AMJUEL, METHAN, H2VIBR

```
        IR  FILNAM  H123  REAC  CRC  MASSP  MASST  DP RMN1
     .    RMX1
C       (format: I3,1X,A6,1X,A4,Axxx,A3,2I3,3E12.4)
        IF (RMN1.GT.0.) READ IFEXMN, FPARM(J),J=1,3
        IF (RMX1.GT.0.) READ IFEXMX, FPARM(J),J=4,6
```

**for "real particles"**, database: FILNAM = ADAS (or any table in an equivalent format as the **ADAS!** adf11 files)

```
            IR   ADAS   H123   REAC   CRC   MASSP   MASST   DP
C           (format: I3,1X,A6,1X,A4,Axxx,A3,2I3,3E12.4)
            ELNAME, IZ
C           (format: 4X,A2,1X,I3)
```

i.e.: one additional input card is read to identify the chemical element and charge state. Extrapolation flags RMN1 and RMX1 are not in use in this case and may be omitted. Default (hard wired) extrapolation schemes are used if the plasma density and temperature is outside the tabulated range.

**for "real particles"**, database: FILNAM = CONST. Fit parameters are directly read from input file, up to nine coefficients. In EIRENE$_{2005}$ or later: Optional : Further parameter FTFLAG, length: not more than 9 characters.

```
            IR   CONST   H123   FTFLAG   CRC   MASSP   MASST   DP
C           (format: I3,1X,A6,1X,A4,(A9),xxxX,A3,2I3,3E12.4)
```

i.e.: Reading of FTFLAG is optional. Default for FTFLAG: =0. Parameter REAC does not exist. Extrapolation flags RMN1 and RMX1 are not in use in this case and may be omitted.

**for "photons"**, database FILNAM = PHOTON

```
            IR   PHOTON   H123   REAC
C           (format: I3,1X,A6,1X,A4,Axxx)
            IPRFTYPE IPLSC3 IMESS IFREMD NRJPRT
            DO IFREMD
               II KENN  IK6
C              (format: I6,1X,A2,3X,I6)
            ENDDO
```

I.e.: after the first line there is a special format for this set of input cards in case of photonic processes, read from the spectral database PHOTON. In case *IFREMD > 0* the additional cards specify options for "foreign pressure line broadening".

Then the subroutine SLREAC is called, which picks the atomic data (fit coefficients) for reaction "IR" from the file FILNAM. It then stores this information on the arrays in Module COMAMF with label *IR*.

```
* 4A. atoms species cards
      NATMI
      DO 44  IATM= 1, NATMI
* read NATMI species blocks with $ = A
      44 CONTINUE
* 4B. molecules species cards
      NMOLI
      DO 46  IMOL= 1, NMOLI
* read NMOLI species blocks with $ = M
      46 CONTINUE
* 4C. test ions species cards
      NIONI
      DO 48  IION= 1, NIONI
```

```
* read NIONI species blocks with $ = I
      48 CONTINUE
```

and, for versions younger than 2003:

```
* 4D. test ions species cards
      NPHOTI
      DO 49   IPHOT= 1, NPHOTI
* read NPHOTI species blocks with $ = Ph
      49 CONTINUE
```

**Meaning of the input variables**

**NREACI** Total number of different reactions to be read.

The next block has different meanings for "real particles" and "photons". Cross section and rate coefficients are specified for particles, but emission and absorption line shapes are specified for photons.

**"real particles"**

An interaction potential $V(r)$, cross section $\sigma$ plus a reaction rate coefficient $\langle \sigma v \rangle$ for one process counts as one reaction, but higher order rate coefficients such as energy or momentum weighted rate coefficients for the same process count as new reaction and must be labelled by a different index $IR$ (see below).

Storage is provided for up to NREAC different additional reactions (see "Parameter Statements", section 3.1), i.e., one must guarantee NREACI.LE.NREAC

**IR** Labeling index of the reaction that is read in. The condition

IR.LE.NREACI

must be fulfilled (otherwise: error exit).

**FILNAM** Name of the data-set that contains the required reaction.

**HYDHEL** Hydrogen and Helium data [**kn:Janev**]

**METHAN** Hydrocarbons data [**kn:Ehrhardt**]

**H2VIBR** Data for $H_2$ molecules and their isotopomeres, including effects of vibrational and electronic excitation, as well as rates needed for radiation trapping calculations.

**AMJUEL** supplement to HYDHEL and METHAN for neutral gas transport Monte Carlo codes, e.g. multi-step reaction rates, etc.

**CONST** reaction IR is a collision process with explicitly specified fit coefficients for cross sections or reaction rate coefficients, (depending upon input flag "H123"), e.g., constant, power law, etc. These fitting coefficients are directly read from the formatted input file rather than from an external file.

**ADAS** Collisional radiative rate coefficients from any files tabulated in the same format as **ADAS!** adf11 files, which must be located in a directory, the path to which is specified in input block 1, section 2.1, by one of the "CFILE" cards described there: *CFILE ADAS path/adf11/*

**PHOTON** line emission and line absorption constants, as well as parameters for line shapes (line broadening constants).

**H123** Identification flag for the type of data: interaction potential parameters, cross section, rate coefficient, momentum loss rate coefficient, energy loss rate coefficient, etc.

**case 0:** FILNAM = CONST

currently available only for: H123 = H.1, = H.2, = H.5, or = H.8, i.e., only for single parameter fits.

in case IFTFLG = 10, 110, 210, ..., "single parameter", constant collision parameter. One more line

F(0) (REAL)

is read.

For all other values of IFTFLG, including the default IFTFLG=0, two more input lines with 9 fitting coefficients

F(I ), I=0,5 (REAL)

F(I ), I=6,8 (REAL)

are read, in subroutine SLREAC from the formatted input file.

The natural logarithm of cross-section: $\ln(\sigma)$ (H123 = H.1) or of a rate coefficient: $\ln(R)$ (H123 = H.2, = H.5, or = H.8) is computed as:

$\ln(\sigma) = \sum_{I=0}^{8} F(I) \cdot \ln(E_{lab})^I$

and, likewise, a rate coefficient $R$ is evaluated as:

$\ln(R) = \sum_{I=0}^{8} F(I) \cdot \ln(T)^I$

Here $E_{lab}$ is the relative energy of impact, but with the mass of the charged particle (i.e., for a target of neutral particles at rest, see below), and $T$ is the electron or ion temperature, depending on the type of impacting plasma particle.

This option permits specification of constant cross sections or rate coefficients (only the first of the nine parameters nonzero), or, e.g., specific power law energy dependencies or temperature dependencies (second of the nine parameters).

**case 1:** FILNAM = HYDHEL, METHAN, AMJUEL, H2VIBR

**H.1** single parameter fit for cross section $\sigma(\mathrm{cm}^2)$ as function of energy ELAB (eV), with ELAB = $m_{Lab}/2 \cdot v_{rel}^2$ (For the definition of $m_{Lab}$ = MASSP : see below.

**H.2** single parameter fit for rate coefficient $\langle \sigma v \rangle (\mathrm{cm}^3\ \mathrm{s}^{-1})$ as function of target temperature (eV), assuming "zero velocity" projectile

**H.3** double parameter fit of rate coefficient $\langle \sigma v \rangle (\mathrm{cm}^3\ \mathrm{s}^{-1})$ as function of projectile energy (eV) and target temperature (eV)

**H.4** double parameter fit of rate coefficient $\langle \sigma v \rangle (\mathrm{cm}^3\ \mathrm{s}^{-1})$ as function of target density ($\mathrm{cm}^{-3}$) and target temperature (eV)

**H.5** single parameter fit of target particle momentum weighted rate coefficient

$\langle\ \sigma\ v\ \cdot\ m_p\ \cdot\ |v_p|\ \rangle$ ($\mathrm{cm}^3\ \mathrm{s}^{-1}$ AMU cm $\mathrm{s}^{-1}$) as function of target temperature (eV)

**H.6** double parameter fit of target particle velocity weighted rate coefficient

$\langle \sigma v \cdot m_p \cdot |v_p| \rangle$(cm$^3$ s$^{-1}$ AMU cm s$^{-1}$) as function of projectile energy (eV) and target temperature (eV)

**H.7** double parameter fit of target particle velocity weighted rate coefficient

$\langle \sigma v \cdot m_p \cdot |v_p| \rangle$(cm$^3$ s$^{-1}$ AMU cm s$^{-1}$) as function of target density (cm$^3$) and target temperature (eV)

**H.8** single parameter fit of target particle energy weighted rate coefficient

$\langle \sigma v \cdot m_p/2 \cdot v_p^2 \rangle$ (cm$^3$ s$^{-1}$ eV) as function of target temperature (eV)

**H.9** double parameter fit of target particle energy weighted rate coefficient

$\langle \sigma v \cdot m_p/2 \cdot v_p^2 \rangle$ (cm$^3$ s$^{-1}$ eV) as function of projectile energy (eV) and target temperature (eV)

**H.10** double parameter fit of target particle energy weighted rate coefficient $\langle \sigma v \cdot m_p/2 \cdot v_p^2 \rangle$ (cm$^3$ s$^{-1}$ eV) as function of target density (cm$^3$) and target temperature (eV)

**H.11** single parameter fit for any other data, e.g. to be used in special user supplied programs.

**H.12** double parameter fit for any other data, e.g. to be used in special user supplied programs, (i.e. not generally understood by EIRENE, but can be used in problem specific "... USR" routines of EIRENE, e.g. for post-processing). The data fitted in H.12 are, therefore, typically not rate-coefficients, but often (not always) ratios between two rate coefficients.

Exception: the reduced population coefficients for hydrogenic plasmas,

i.e., 2.1.5a, ..., 2.1.5e, 2.2.5a, 2.2.5e, etc.

are currently automatically read in EIRENE post-processing routines in code section "output", routines: Ba_alpha, Ba_beta, ...

**case 2:** FILNAM = TAB2D or FILNAM =ADAS

**H.4** double parameter table of rate coefficient $\langle \sigma v \rangle$(cm$^3$ s$^{-1}$) as function of target density (cm$^{-3}$) and target temperature (eV) read from files containing 2D tabulated rate coefficients, i.e. rate coefficients tabulated vs. 2 independent parameters. This option uses the **ADAS!** adf11 file format as guidance and is developed/adapted from it for more general A&M data structures.

E.g. original **ADAS!** file-names starting with SCD... (ionization rate coefficients contain ionization rate coefficients to charge state Z, Z = 1, ..., $Z_{max}$) and filenames starting with ACD... contain recombination rate coefficients, from charge state Z, Z=1, $Z_{max}$. The particular chemical element ELNAME and the value of charge state $Z$ is specified in the next input card, which is read only in case of FILNAM = TAB2D or FILNAM = ADAS.

**H.10** same as for H123='H.4', but ionisation and recombination electron energy loss rate coefficients rather than reaction rate coefficients. In the original **ADAS!** files the names PLT... and PRB... are used for those quantities. Note: distinct from the AMJUEL database convention in case of **ADAS!** format the PLT rates (watt cm$^{-3}$)

do not contain the potential energy loss DP × SCD, and the PRB rates contain, in addition to their corresponding rates from AMJUEL, also a Bremsstrahlung contribution. Internally EIRENE subtracts this **ADAS!**-Bremsstrahlung from electron energy weighted PRB rates whenever such rates are read via the **ADAS!** format (i.e. whenever FILNAM = ADAS)

Note: in general the potential energy difference DP may or may not be included in the definition of energy rate coefficients H.8, H.9 and H.10 as read from a particular database. For example the energy rate coefficients may, in effect, be either a radiation loss rate or a total electron energy loss/gain rate.

This database specific convention may be altered by EIRENE input flag DP during the internal preparation of atomic data files. Proper choice of DP may then turn a radiation loss rate to a total energy loss/gain rate, and vice versa. (The use of flag DP is the same as in case of the AMJUEL database or any other external database). Default EIRENE electron energy source tallies then may have a different meaning, depending on the choice of input flag DP and the particular database used (**ADAS!**, or AMJUEL, etc.)

**case 3:** FILNAM = PHOTON to be written

**REAC** Name or number of the reaction as used in the file FILNAM.

If FILNAM = METHAN, or AMJUEL, or HYDHEL, or H2VIBR,

REAC is the number (label) of a cross section or of a rate coefficient fit. E.g.: REAC = 2.15.2 in file METHAN for the reaction $e + CH_4^+ \rightarrow CH_3^+ + H + e$.

If FILNAM = CONST,

then REAC can be used to input the flag IFTFLG. Whether or not REAC is to be interpreted as IFTFLG is identified by presence of the string FT (e.g.: 'REAC' = 'FT 110' would lead to IFTFLG = 110 for this reaction).

If FILNAM = ADAS,

then the character string REAC identifies the particular file within the directory identified by the path given for **ADAS!** files in input block 1, see card: "CFILE" DBHANDLE DBFNAME in block 1 (section 2.1). For example, REAC = SCD96, then the file SCD96_h.dat, is read, in this example this is for ionisation rate coefficients of H atoms. In this latter example the additional input species identifier ELNAME reads: H (for hydrogen). In Module eirene_read −adas.f, called from SLREAC.f, the corresponding data table is then read into EIRENE.

If FILNAM = PHOTON

to be written

**CRC** Identification for the type of the reaction process. Depending upon the value of this flag various assumptions are automatically made concerning the atomic data in the initialization phase.

CRC=EI electron impact collision, i.e., ionization, excitation or de-excitation or dissociation. (In older EIRENE version CRC=DS was also in use. This is now automatically identified with CRC=EI)

CRC=CX (quasi-) resonant charge-exchange

CRC=EL elastic collision

CRC=PI in-elastic ion impact collision (not fully implemented)

CRC=RC re-combination

CRC=PH photonic processes

CRC=OT other (e.g. population coefficients, equilibrium density ratios, etc.)

**MASSP** Mass number (**AMU!** (**AMU!**)) of the first particle involved in the collision, for the original data as being read from the data-file. EIRENE automatically re-scales data according to the mass number of the actual first particle involved in this particular collision process (e.g. if data are given for H atoms in the data-file, but are used for D or T atoms in an EIRENE run).

**MASST** Mass number (**AMU!**) of the second particle involved in the collision, for the original data as being read from the data-file. EIRENE automatically re-scales data according to the mass number of the actual second particle involved in this particular collision process.

The next input deck is read only in case FILNAM = ADAS:

**ELNAME** Name of element in adf11 style formatted file, e.g. $Fe$ for iron, $C$ for carbon, $W$ for tungsten, etc.

**IZ** Charge state in adf11 file. This value is always between 1 and $Z_{max}$. For ionization rate coefficients $IZ$ specifies the ionization rate coefficient for the ionization process from $IZ - 1$ to $IZ$, i.e. the final charge state. For recombination process $IZ$ specifies the recombination rate coefficient for the recombination process from $IZ$ to $IZ - 1$, i.e. the initial charge state for these processes.

The next input decks are read only in case FILNAM = PHOTON:

**IPRFTYPE**

**IPLSC3**

**IMESS**

**IFREMD**

**NRJPRT** to be written

Then $IFREMD$ cards are read

**II, KENN, IK6**

$II$: labeling index, irrelevant

$KENN$: to be written

$IR$: to be written

**Note:**
The nomenclature used in reference [**kn:Janev**] has lead to a certain confusion with regard to the masses of the particles involved in a particular collision process. A series of test calculations has revealed the following definitions implicit in the tables of references [**kn:Janev**] and [**kn:Ehrhardt**]:
For the cross section energy scale, the relevant **laboratory frame** mass is that of the charged particle, assuming an ion- or electron beam incident onto a cold neutral gas at rest (i.e., the mass in the energy scale is **neither** the reduced mass **nor** is the energy given in units energy/**AMU!**).
For rate coefficients, depending upon both the beam energy and the target temperature, however, the neutral particles are considered to be the beam particles, i.e., their mass is the relevant mass for the energy scale now, whereas the charged particles are considered as target, with their mass being the relevant mass for the temperature scale.
Following these conventions, EIRENE assumes that cross sections have been measured with the charged particle as projectile and the neutral target at rest.
MASSP is assumed to have been the relevant mass number of the original data for the energy scale in cross sections (H.1) and for the temperature scale in the rate coefficients (H.2, H.3, H.4). If MASSP = 0, MASSP is defaulted to the electron mass (**AMU!**). I.e., MASSP is the mass number of those collision partners in the **database**, which would play the role of bulk particles in an EIRENE run. But then an automatic re-scaling to the true EIRENE bulk ions mass RMASSP(IPLS) of the **bulk particle in a particular case** is carried out.
MASST is assumed to have been the relevant mass number in the **database** for the projectile energy scale in H.3, H.6 and H.9, i.e., for the test particles in an EIRENE run. Also this is then automatically re-scaled to the **test particle in a particular case**
EIRENE then automatically converts the energy/temperature scales in the data fits to the correct scales for the particular particles masses (isotopes) involved in a collision event during the simulation process, if these do not happen to coincide with MASSP and MASST.

**DP** (only for H.8, H.9, H.10) additional (e.g., potential) energy lost or gained per reaction, which is not already included in the rates. This may be needed due to the logarithmic fits used and if changes in sign arise. For example, in case of recombination of hydrogen ions, AMJUEL H.10, 2.1.8, DP=+13.6 must be specified.

In reaction decks H.8, H.9 or H.10 in AMJUEL, this value of DP is now automatically read (and overwrites the value specified here). Hence: in the EIRENE versions later than June96 this input flag is redundant.

(Be careful: check that this is really the case in any particular version of EIRENE)

The next flags RMN1, RMX1, RMN2, RMX2 control the options to extrapolate atomic data fits read from files in order to obtain proper asymptotic behavior of fits. The default data in EIRENE and some reactions in the files AMJUEL, HYDHEL, ... data-set already contain that information. RMN1, RMN2 control the first independent parameter (often: temperature), and RMN2, RMX2 control the second (if any) independent parameter (often: density) of the fits or tables.

**RMN1 = RMX1 = 0.0** EIRENE searches for extrapolation parameters in the database, and sets the values ELABMIN and ELABMAX for cross section data, TMIN, TMAX for rate coefficients etc. Currently this works for single parameter fits only. Check routine SLREAC.F

for the status wrt. to two-parameter fits. If it finds these extrapolation parameters, it uses their values as RMN1 and RMX1, respectively. In such cases the next card IFEXMN, ... and IFEXMX, ... must also be omitted, because the corresponding information regarding the extrapolation expression (and coefficients) is found from the atomic data file as well. The extrapolation expression is that of option IFEXM. = 5 (see below).

Likewise, in case of reaction rates, EIRENE searches for TMIN, TMAX (for single parameter fits H.2, H.5, H.8 and H.11) and additionally for PMIN and PMAX (if H.3, H.4, H.6, H.7, H.9, H.10 or H.12) and the extrapolation parameters again are expected in the atomic data file.

**RMN1** $\neq 0.0$ This option overrules any (if available) low parameter end asymptotics for the present reaction IR, which was possibly found in the atomic data file, and this allows to set this asymptotic behavior explicitly here in the input file.

RMN1 = minimum energy or temperature (in eV) for data as read from file. For energies (cross sections) and temperatures (rate coefficients) below RMN1 (eV) the data are extrapolated using the next input card

IFEXMN,FPARM(J),J=1,3.

See below, and e.g., subroutine CROSS for the various options.

Some cross section data, e.g. in the files AMJUEL, HYDHEL, etc. already contain the extrapolation information, and hence the default RNM1=0 can be used there.

**RMX1** $\neq 0.0$ same as above, but for high parameter range extrapolation. This option overrules any (if available) high parameter end asymptotics for the present reaction IR, which was possibly found in the atomic data file, and this allows to set this asymptotic behavior explicitly here in the input file.

**IFEXMN, FPARM(J), J=1,3**

various extrapolation expressions $ex_{low}(x)$ at the low end (x < RMN1, x = $E_{lab}$ or x = T) are available:

**IFEXMN = 1**

$ex_{low}(x) = 0$

(e.g., cross section with nonzero threshold at RMN1)

**IFEXMN = 2**

y=x/FPARM(1)

$ex_{low}(x) = FPARM(2) \cdot y^{FPARM(3)} \cdot \log(y)$

recommended for cross section extrapolation at high energy end, for reactions with nonzero threshold at FPARM(1) [eV].

**IFEXMN = 3**

y=log(x)

$\ln(ex_{low}(x)) = FPARM(1) + y \cdot FPARM(2)$

**IFEXMN = 4**

not in use

**IFEXMN = 5**

$y = \log(x)$

$\ln(ex_{low}(x)) = \sum_{I=1}^{3} FPARM(I) \cdot y^{I-1}$

**IFEXMN < 0**

linear extrapolation in log-log scale. Coefficients FPARM(J), J=1,2 are automatically determined, e.g., in function CROSS, FPARM(3) = 0.D0, and then extrapolation option IFEXMN = 5 is used with these coefficients (same as IFEXMN = 3 in this case).

**IFEXMX, FPARM(J), J=4,6**

same as above, but for high parameter range extrapolation $ex_{high}(x)$.

*4A. Atoms Species Cards

**NATMI** Total number of atomic species blocks

**IATM** irrelevant; labelling index

*4B. Molecules Species Cards

**NMOLI** Total number of molecule species blocks

**IMOL** irrelevant; labelling index

*4C. Test Ions Species Cards

**NIONI** Total number of test ion species blocks

**IION** irrelevant; labelling index

*4D. Photon Species Cards

**NPHOTI** Total number of photon species blocks

**IPHOT** irrelevant; labelling index

The meaning of the variables in a species block is as follows:

**TEXT$** Name of the species on printout and plots

**NMASS$** Mass number

**NCHAR$** Nuclear charge number

**NPRT$** Flux units carried by one particle of this type and species. The "WEIGHT" of a test flight has dimensions: "atomic flux", i.e., "equivalent atoms per second". For example, a methane molecule $CH_4$ should have $NPRT_{CH_4} = 5$, if H atoms and C atoms have $NPRT_H = NPRT_C = 1$.

NPRT is used to convert fluxes into equivalent atomic fluxes, for scaling, (see input flag FLUX in block 7), and for flux conservation in non-diagonal (species changing) events at surfaces and in the volume. (NPRT is irrelevant for atoms, and always set to one for them.)

**NCHRG$** Charge state (irrelevant for atoms and molecules, always set to zero for them)

**ISRF$** Species index flag for "fast particle reflection model" (see block 6A), irrelevant for molecules

**ISRT$** Species index flag for "thermal particle re-emission model" (see block 6A)

**ID1$** not in use in versions 98 to 2002. ID1 was the "sputtered species index" in versions older than "98", but this species index has then become a surface property, see input block 3. In versions younger than 2002 this flag can be used to increase the number of secondary test particle groups from 2 or less (default) to 3, if ID1=3 or to 4, if ID1=4. This option then allows more complex fragmentation processes of large molecules as compared to the default ID1=2.

**NRC$**

$> 0$ total number of reaction decks to be read for this species

$= 0$ default model is to be used.

$= -1$ no collision processes for this particle species at all

**NFOL$** controls the motion of test particles.

$\geq 0$ default test particle tracing model is to be used.

$= -1$ motion of test particle is not followed (static approximation). The test particle is destroyed immediately at it's point of birth by a collision. A collision estimator is used rather than a track-length estimator for all default volumetric tallies, as described in section 3.2.

**NGEN$** Maximum number of generations for this test particle species. May 2018: extended options for NGEN <0, "fluid limit".

$> 0$ Maximum number of generations for this test particle species. For each Monte Carlo particle history the generation counter XGENER is reset to zero at birth, after surface events and after a non-diagonal event (modification of type and/or species of a particle in a collision event). The generation counter XGENER is increased at entropy producing events (elastic and diagonal charge exchange collisions). If, along a trajectory, the condition $XGENER > NGEN \ldots$ is met, after a collision, no test particle secondaries are generated, i.e. the history is stopped. The post-collision particle flux, parallel momentum flux and energy flux is, instead, put into the tallies PGENA, VGENA and EGENA for atoms, resp., PGENM, VGENM and EGENM

for molecules, resp., PGENI, VGENI and EGENI for test ions, respectively and PGENPH, VGENPH and EGENPH for photons, respectively.

$< 0$ Flag for "fluid-limit", i.e. for critical "Knudsen number".

Define: FDLM=-10000/(NGEN+1)

Default: FDLM = $\infty$, no fluid limit.

E.g.: NGEN... =-10001, then FDLM=1. Smaller (even more negative) values of NGEN... lead to larger critical Knudsen numbers FDLM, and vice versa. If the collision mean free path at a collision event, compared to a local geometric dimension (cell size) falls below a certain value (FDLM) (fluid limit), then the trajectory is stopped and the above mentioned generation limit tallies are scored.

$= 0$ In case NGEN... = 0, (default) no generation limit is activated for atoms (NGENA = 0), molecules (NGENM = 0) or test ions (NGENI = 0). I.e., strictly, the default NGEN... = 0 is equivalent to specifying NGEN... = infinity.

**NHSTS$** (only in versions 2004 and younger): NHSTS=-1 turns off the trajectory plot for this particular species. (default: NHSTS=0)

**LKIND$** index of the atomic species with which this species scales. $\$ \in \{M, I, P\}$ Only used within SOLPS-ITER

**IREAC$** Identification flag for collision data. IREAC$ = IR, and IR is the labelling index from the reaction card (see above).

The next three flags control number, species and type of particles involved in this particular collision process.

**IBULK$** pre-collision bulk particle identifier

= NLM

**M** type flag ITYP for impacting bulk particle (electron (ITYP = 5) or bulk ion (ITYP = 4),

irrelevant, defaulted to M = 4 for bulk ion impact collision and defaulted to M = 5 for electron impact collisions

**L** irrelevant, defaulted to L = 1, because number of pre-collision bulk particles is known from the type CRC of each collision process.

**N** Species index of pre-collision bulk ion (corresponds to IPLS in bulk ion species cards, see block 5). Irrelevant for electron impact collisions, CRC=EI.

**ISCD1$** first heavy secondary particle group identifier

= IJKLM

**M** type flag ITYP for these first secondaries group (atoms (ITYP=1), molecules (ITYP=2), test ions (ITYP=3) or bulk ions (ITYP=4))

**L** number of secondaries of this type and species

**JK** Species index of secondary (IATM, IMOL, IION, or IPLS), two digits

**I** relative importance of this first secondary group as compared to the second secondary group. If $I \leq 0$, I is defaulted to I = 1

(currently not in use)

**ISCD2$** second heavy secondary particle group identifier

= IJKLM.

Same as ISCD1$, but for second secondary group.

**Note:**
The number of secondary electrons (if any) need not be specified, it is computed internally from charge conservation.
In case of collisions with only one heavy particle secondary (test particle or bulk "ion") (electron impact collision CRC=EI, or re-combination CRC=RC) ISCD1 is used and ISCD2 is irrelevant.
In case of charge-exchange (CRC=CX) ISCD1 is used for the new state of the impacting bulk particle after the collision, whereas ISCD2 controls the options for the post-collision state of that particle which was the test particle prior to the collision. Consistency of the particle masses of pre- and post-collision particles with this convention is checked in the initialization phase of each run. "CX" collisions mean: exchange of identity, i.e. scattering angle $\pi$ in the center of mass system. I.e. CRC=CX is used for resonant charge-exchange only. Non resonant charge-exchange, in which e.g. internal energy is converted into kinetic energy of products, is to be treated in the CRC=PI category of "general heavy particle collisions".
In case of elastic collision (CRC=EL), the ISCD1 and ISCD2 flags are are irrelevant, because the colliding particles retain their identity by default.
In case of electron impact collision (CRC=EI), the ISCD1 and ISCD2 flags are symmetric, i.e. they can be interchanged without any effect on the calculation.

**ISCDE$** flags for post-collision velocity space distributions for all particles involved in the collision event. This flag consists of 5 integer digits.

Let us write JKLMN = ISCDE

The meaning of some of these digits (LMN) has undergone changes in the years 2000 – 2005, M and N have become entirely redundant for collision energetics and may now, in versions younger than 2017, control other options (radiation loss, (appearance) potential difference, etc., see below)

We describe the old (1999 and older) and current meanings in a separate paragraph below.

**IESTM** Three digits IESTM=LMN. N, M and L are flags for the choice of estimators for particle, momentum and energy sink and source term contributions from the collision process. (Default: = 0 "track-length estimators".) In some cases, when track-length estimators can not be used because the corresponding momentum or energy weighted Maxwellian rate coefficients are not available in a particle run, (i.e. are missing in the preceding list of NREAC reaction data), then internally a switching from track-length to collision estimators is carried out, and a corresponding warning is printed.

**IBGK** Three digits IBGK=NML. Particle identifier for **BGK!** self and cross collisions. The format is the same as for IBULK, ISCD1 and ISCD2 species identifier flags. I.e., it has three digits, pointing to the type and to the species index within that type-group. The second digit is irrelevant and defaulted to one (only binary collisions are considered). This flag controls options for non-linear test-particle - test-particle collision.

Let ISPZ be the species index of a test-particle. If the collision is a self-collisions, this flag IBGK must point to the species ISPZ itself. Otherwise it must point to the 2nd test particle species ISPZ2, which is involved in the **BGK!**-cross collision term.

IBGK must be consistent with IBULK, which in this case is the "artificial background species" (= result from previous iteration) in the linearization of the collision operator.

I.e.: if test particle species $A$ collides with test particle species $B$ (of same or of different species and type), then a bulk species $AB_{bulk}$ must be present in input block 5, such that species $A$ and $AB_{bulk}$ have the same mass, nuclear charge and charge numbers. The density, flow field and temperature of collision partner $AB_{bulk}$ are iterated, using either the corresponding parameters estimated for species $A$ (in case of self-collisions) or using parameters obtained from the formulas for $n_{AB}, T_{AB}, V_{AB}$ (see section 1.9.1) for cross collisions between species $A$ and $B$.

The flag IBULK must point to the corresponding species IPLS=$AB_{BULK}$ in block 5.

An example for input specifications of non-linear **BGK!** collisions is given below in section 2.4.3. (Default: = 0, no iteration for non-linearity in collision kernel, no "artificial background species" involved in this collision, and no **BGK!**-tallies are updated for iterating the artificial background species.)

**EELEC** parameter EP for electron energy loss per collision. Its meaning depends on fifth digit of ISCDE flag and is described above.

**EBULK** parameter EP for pre-collision bulk ion energy loss per collision. Its meaning depends on fourth digit of ISCDE flag and is described above.

**ESCD1** parameter EP for velocity space distribution of secondaries. Its meaning depends on third digit of ISCDE flag and is described above.

**EDUMMY** (was earlier: **ESCD2**) in versions 1999 and older: the parameter ESCD2 = EP was used for velocity space distribution of the second secondaries group, whereas ESCD1 was used for the first group of secondaries. Now: this parameter ESCD2 is not in use anymore. Distribution of energy (kinetic energy release) EP over secondaries is now by default inversely proportional to the masses of secondary particles (as follows from momentum conservation in binary collisions). For backward compatibility of input files now internally: EP = ESCD1 + EDUMMY.

**FREAC** multiplier to scale the cross section (and rate coefficients) of this process. Default: FREAC = 1.0. If FREAC $\leq$ 0, it is also defaulted to 1.0. To turn off this process, set FREAC to a very small number: e.g.: FREAC = 1.0000 E-30

**EDPOT** Potential energy difference in the specified collision process. Default: EDPOT = 0.0. This energy allows to derive (implicit) radiation energy losses from the multi-step electron

collision process. The incident electron energy difference(loss or gain) per collision EELEC is the sum:

ESCD1 (kinetic energy released in educts, KER) plus

EDPOT (potential energy difference) plus

ERAD (radiated energy, e.g. from intermediate excited states).

The old "fluid limit" cut off options formerly specified at this place (the FLDLM flag) are now controlled by the NGEN flag, see above.

### 2.4.1 Collision kinetics

In this paragraph we describe the options controlling the energies of post-collision particles and the electron and bulk ion energy gain or loss rates. These options are controlled by the integer input flag ISCDE and the three real flags EELEC, EBULK, ESCD1 mentioned above.
**Very old options** (before around year 2000) will only briefly be described here, but in general these must be inferred from the coding on a case by case basis.
Let us write JKLMN = ISCDE
Each of the five digits did control a different type of particle: J: electrons, K: bulk ions, L: test ions, M: molecules and N: atoms. In addition to ESCD1 there was a second energy parameter ESCD2, in those days meant for the kinetic energy of the first and second heavy post-collision particle, respectively. Later ESCD2 became redundant, and only ESCD1 (the sum of both old values: ESCD1 + ESCD2, was used since then. Internally the code since then distributes the total kinetic energy release ESCD1 to the two post-collision particles according to the laws of energy and momentum conservation.
More: to be written . . . needs detective work in very old code . . .
**Current options:**
The energy losses and gains of all particles participating in a collision event are controlled by the input flag ISCDE, as well as by the energies EELEC, EBULK (pre-collision energies of electrons and heavy background particles, respectively and ESCD1 (total post-collision energy: KER: kinetic energy release) of heavy post-collision particles (excluding electrons).
Let us again write JKLMN = ISCDE
Digits M and N are not in use any more.
Let then furthermore the character # stand for J, K, L respectively. Then J controls electrons (if any involved), K controls pre-collision (plasma) bulk particles (if any involved), L the heavy secondary particles (if any).
Each of these single integer flags controls the meaning of the related energy parameters, which are read in the next card:
EP=EELEC (# $\hat{=}$ J), for the electrons involved in the collision process
EP=EBULK (# $\hat{=}$ K), for the pre-collision bulk ions
EP=ESCD (# $\hat{=}$ L), for the heavy secondary particles (this flag L is only in use for CRC=EI and CRC=PI collisions, whereas for EL, CX and RC type collision processes the kinetic energy of heavy products is determined by default models assigned to these types of processes, independent of the value of flags L and ESCD).
The following flags are in use:
For EL (elastic) collision processes: only parameter K is in use

For CX (charge-exchange) collision processes: only parameter K is in use
For EI (electron impact) collision processes: parameters J and L are in use
For PI (heavy particle) collision processes: parameters J, K and L are in use
For RC (recombination) collision processes: only parameter J is in use

**# = 0** constant loss or gain of EP (eV) per collision event.

If # ≙ L, and if there is more than one heavy particle secondary, then the energy loss/gain EP (reaction exothermicity, or kinetic energy release) is distributed over the secondary bulk particles inversely proportional to their masses. E.g. for reaction $e + CH_4 \rightarrow CH_3 + H + e$, and EP=ESCD=8.0 eV, then the fragment $CH_3$ would receive an extra energy of 0.5 eV, and the fragment $H$ would receive 7.5 eV, both isotropically in the center of mass system of the collision.

**# = 1**

**for # ≙ J** (spatially dependent) bulk electron temperature dependent loss $EP = -(1.5 \cdot T_e)$ per collision, where $T_e$ is the local electron temperature at the point of collision.

**for # ≙ K** (spatially dependent) bulk ion energy dependent loss $EP = -(1.5 \cdot T_i + E_{Drift})$ per collision, where $T_i$ is the local bulk ion temperature at the point of collision, and $E_{Drift}$ is the kinetic energy of the drift motion (VXIN,VYIN,VZIN) of the background ions, see input block 5. Both contributions are evaluated for background bulk species IPLS which is the bulk collision partner in this particular reaction. The local drift energy contribution $E_{Drift}$ is added only in case NLDRFT=TRUE, see input block 1 (section 2.1)

**for # ≙ L** a fraction EP=ESCD of the energy loss EBULK of the impacting bulk particle is distributed equally over all (if more than one) heavy particle secondaries of group 1 and 2 respectively. This option is currently not in use

**# = 2** Currently not fully implemented:

Let # stand for a pre-collision bulk particle: # ≙ J or # ≙ K. The energy loss of the impacting bulk particle is calculated from the velocity and energy weighted rate coefficients. In this case EP must be the labelling index IR of the reaction card by which these moments have been read from the data-file.

Let now # stand for a post-collision heavy particle group, i.e # ≙ L. Then the velocity vector of the secondary particle is sampled from a cross section weighted (and optional, if NLDRFT, shifted by the local drift velocity of the pre-collision bulk particle) Maxwellian distribution at the local ion temperature $T_i$. The cross section is taken at the relative velocity of the impacting test particle and a particle sampled from the above mentioned (shifted) Maxwellian. This option permits to identify one particular collision partner from the background population, correctly accounting for the energy dependence of the cross section.

**# = 3** KREAD option: description to be written, see example of default hydrogen molecule reaction model, below, there: reaction 6 and 7. In principle an energy weighted loss rate coefficient is read in, a mean energy loss per collision is derived from that.

If this option is used for heavy particles secondaries (i.e. the third digit of ISCDE has the value 3), than this energy is again distributed inversely proportional to the masses over all heavy particle secondaries.

**# = 4** to be written

**# = 5** to be written

#### default resonant charge-exchange model

The default CX model is set up in order to define collision kinetics and mean free paths without need for integrating cross sections into rate coefficients. It is based on total cross sections only, plus a few approximations made, but it retains thermal force effects.
In the default (minimal) model hydrogenic atoms (here: D atoms) undergo resonant charge-exchange with hydrogenic ions (here: $D^+$ ions), utilizing the cross section data for reaction 3.1.8 (again from reference [**kn:Janev**], but with improved low energy asymptotics). The assumptions for processes labeled as CX are, generally: pure electron capture from the neutral by the ion (="exchange of identity" in symmetric charge-exchange), scattering angle $\pi$ in center of mass system, zero exothermicity, i.e. perfect conservation of kinetic energy in center of mass system before and after collision. This set of assumptions translates in the procedure to first sample the velocity of a bulk ion going into the collision, and then to continue the former test particle trajectory with unmodified velocity, with its charge increased by one, and to continue the former bulk ion, also with unmodified velocity, with its charge decreased by one (i.e., as neutral particle in case of singly charged bulk ions).
The ISCDE parameters (reaction kinetics) for default CX processes are zero: ISCDE = JKLMN = 00000 (or: = 00001, which has the same meaning because the fifth digit N is presently irrelevant). The second digit (i.e., here K = 0) controls the options for the velocity of the background ion going into the collision. For the default model choice K=0 the following model is activated:
With respect to the distribution of ions going into a CX collision $f_{in\ CX}(\vec{v}_i)$ the background ion velocity distribution $f_i^{CX}(\mathbf{v}_i)$ is assumed to be a *drifting, isotropic in the plasma frame, and mono-energetic distribution*, with the root mean square speed $\overline{v_i} = \sqrt{3\ kT_i/m_i}$ taken as representative speed of the ions, in the ion (plasma) frame. I.e., this isotropic plasma frame distribution is shifted by $\mathbf{v}_{Drift,ipls}$ in the laboratory frame:

$$f_i^{CX}(\mathbf{v}_i - \mathbf{v}_{Drift,i})d^3v_i = \frac{1}{4\pi}\frac{1}{(v_i)^2}\delta(v_i - \overline{v_i})d^3v_i \quad v_i = |\mathbf{v}_i| \tag{2.2}$$

The $f_i^{CX}$ weighted reaction rate coefficient itself is approximated, namely it is evaluated as

$$\langle \sigma_{CX} \cdot v_{rel} \rangle \approx \sigma(v_{eff}) \cdot v_{eff} \tag{2.3}$$

with an effective relative velocity

$$v_{eff} = \sqrt{3kT_i/m_i + (\mathbf{v}_0 - \mathbf{v}_{Drift,i})^2}$$

This approximation to the rate coefficient, for this particular model choice of $f_i^{CX}$, is almost exact. It is assuming that the effect of the scalar product $-\mathbf{v}_i \cdot \mathbf{v}_0$ in the relative velocity between ion- and neutral velocity becomes negligible after integrating $\sigma(v_{rel}) \cdot v_{rel}$ over solid angle, which would

vanish indeed exactly if $\sigma \propto 1/v_{rel}$ (notation: "Maxwellian Molecule collision kinetics", or "Langevin cross section" resulting from interaction potentials $V(r)$ which scale as $V(r) \propto 1/r^4$). It is furthermore assumed in this default model that the ions undergoing CX carry a mean momentum of $m_i \mathbf{v}_{Drift,i}$. The mean energy of ions undergoing charge-exchange in this approximation is $1.5 \cdot T_i$ plus the kinetic energy in their drift motion $m_i/2(\mathbf{v}_{Drift,i})^2$. These two averages are used for the corresponding contribution in the CX momentum- and energy exchange rate coefficients, respectively. The resulting energy and momentum exchange rates would, again, be fully correct in case of "Maxwellian molecule collisions" :$\sigma \propto 1/v_{rel}$
The velocity of the impacting ion is sampled without this "Maxwellian Molecule" approximation being made. I.e., consequently, sampling is from the a drifting, isotropic and mono-energetic distribution $f_i^{CX}$ (2.2) **weighted** by $\sigma(v_{rel}) \cdot |\mathbf{v}_{rel}|$, with $v_{rel}$ the relative velocity between the two collision partners. The sampling distribution $f_{in\ CX}$ for the velocity $\mathbf{v}_i^{CX}$ of ions going into a CX event depends on the pre-collision test particle velocity $\mathbf{v}_0$ as well and is (as also in general cases) given by:

$$f_{in\ CX}(\mathbf{v}_i^{CX}, \mathbf{v}_0) = \left[\sigma(v_{rel}) \cdot v_{rel} \cdot f_i^{CX}(\mathbf{v}_i)\right] / k_{CX} \tag{2.4}$$

where the normalisation constant (rate coefficient) $k_{CX}$ is the rate coefficient obtained by averaging over $f_i^{CX}$: $k_{CX} = \langle \sigma(v_{rel}) \cdot v_{rel} \rangle (E_0, f_i^{CX})$.
(Note: from this relation is is clearly seen that, in general, $f_i^{CX} = f_{in\ CX}$ if and only if $\sigma(v_{rel}) \propto 1/v_{rel}$).
Summarizing: The current default CX model approximations consist in:

a) selecting a particular distribution $f_i^{CX}$ (2.2) rather than e.g. a drifting Maxwellian

b) evaluating the reaction rate coefficient by (2.3) rather than a full rate coefficient integration

c) approximating the energy and momentum exchange rate tallies by again employing the "Langevin" $\sigma \propto 1/v_{rel}$ approximation

d) but carrying out the full (relative velocity selective) reaction kinetics, i.e. here without making the $\sigma \propto 1/v_{rel}$ approximation, and, hence, retaining e.g. thermal force effects.

Instead of the weighting mentioned above, which would require knowledge of $k_{CX}$, a rejection method is used to avoid the weighting, and without need of knowing the integral $k_{CX}$, but still to simulate the same collision integral. Currently this rejection method, for each individual collision process, is based upon pre-evaluation (initialisation phase of a run) of the product $\sigma(E_{rel})v_{rel}$ in the hard wired energy range 0.1 eV to $1 \times 10^4$ eV, determination of the the maximum SGVMAX of that product in this energy range. Then during the test particle tracing a comparison "on the fly" (and rejection, if appropriate) is carried out, of the actual value of this product with RANF × SGVMAX, RANF being a uniformly distributed random number. If the relative collision energy of the incident test particle and the sampled bulk collision partner happens to fall outside this energy range for rejection, then rejection sampling is abandoned and the sample is accepted.
Check in subr. VELOCX to find out which of the two physically equivalent methods are in use.
Default reactions 5.3.1 and 6.3.1 (added in April 2006, to simplify input for pure He plasma simulations) are the two resonant charge-exchange processes

$$\mathrm{He} + \mathrm{He}^+ \rightarrow \mathrm{He}^+ + \mathrm{He}$$

$$He + He^{++} \rightarrow He^{++} + He$$

The weighting (or rejection-technique) was absent in versions younger than 99 for this particular type of (default) charge-exchange collision integral approximation, which, strictly speaking, had therefore led to a slight violation of the second law of thermodynamics (H-Theorem), due to an inconsistency between the in-scattering term (determined by $\sigma_{CX}$ and the out-scattering term (determined by $\langle \sigma_{CX} v_{rel} \rangle$) in the Boltzmann CX collision integral, although mass and energy had been strictly conserved in each collision.
Alternative to the choice ISCDE = 00000 made in the default model one may also use ISCDE = 01000 (with all the rest kept identical). In this case the rate coefficients are evaluated in the same way as described above, but neutrals emerging from CX collisions are sampled from the local drifting Maxwellian distribution of ion collision partners rather than from a drifting isotropic mono-energetic distribution.

## 2.4.2 Default atomic and molecular data

If default atomic data are requested for some test particle species, these are activated by setting the flag NRCM(IATM) = 0 (or NRCM(IMOL) = 0, NRCI(IION) = 0, respectively, or NRCP(IPLS) = 0 in input block 5) for a particular atom IATM (molecule IMOL or test ion IION, or (block 5) bulk ion IPLS), and by omitting the corresponding reaction cards
IREAC$, . . . and
EELEC$, . . .
Then EIRENE tries to find, automatically, the corresponding species types and indices of the collision products in the species blocks. It then tries to define all variables in the cards
IREAC$, . . . and
EELEC$, . . .
for collision rates and collision kinetics, using default assumptions.
If a test particle is to be assigned no collision processes at all, neither from the reaction list in block 4, nor a default process, then NRCM(IATM)=-1, and analogous for the other types and species of particles.

### default model for atomic H (=H,D,T) and He test particles

Assume that the bulk ion $H^+$ is specified as bulk ion species no. “a”, $D^+$ as bulk species “b”, $T^+$ as bulk species “c” and $He^+$ as bulk ion species no. “d”, then the default atomic collision models for H (and isotopes) and He are identical to a model that would result from the following specifications in input block 4.
For atoms D (likewise for H or T), e.g., IATM=1, and atoms He (e.g., IATM=2):

```
* ATOMIC REACTION CARDS   NREACI=
5
   1 HYDHEL H.2  2.1.5      EI   0   1
   2 HYDHEL H.1  3.1.8      CX   1   1
   3 HYDHEL H.2  2.3.9      EI   0   4
   4 HYDHEL H.1  5.3.1      CX   4   4
   5 HYDHEL H.1  6.3.1      CX   4   4
*NEUTRAL ATOMS SPECIES CARDS:   NATMI SPECIES ARE CONSIDERED,
```

```
NATMI= 2
 1 D          2   1   1   0   1 -1   0   2
      1    115    *14       0 00000
 -1.3600E 01   0.0000E 00
      2    *14    111    *14 00000
  0.0000E 00   0.0000E 00   0.0000E 00
 2 He         4   2   1   0   2   2   0   3
      3    115   **14       0 00000
 -2.4588E 01   0.0000E 00
      4   **14    211   **14 00000
  0.0000E 00   0.0000E 00   0.0000E 00
      5  ***14    211  ***14 00000
  0.0000E 00   0.0000E 00   0.0000E 00
```

Here $*$ is the species index of the $D^+$ bulk ion in input block 5, $**$ is the species index of the $He^+$ bulk ion in input block 5, and $***$ is the species index of the $He^{++}$ bulk ion in input block 5. If a corresponding background species is not found in block 5, then the corresponding default process is de-activated.
Hence: D and He atoms are ionized by electron impact, using the Corona data 2.1.5 and 2.3.9, respectively, from reference [**kn:Janev**], and with constant electron energy loss per ionization of 13.6 eV and 24.588 eV, respectively.

#### default model for hydrogenic molecules $H_2$ (and isotopomers) and their ions

For hydrogenic molecules ($H_2, D_2$, HT,. . . ) or molecular ions ($H_2^+, D_2^+, HT^+$,. . . ) default dissociation, ionization and dissociative recombination data are available.
EIRENE uses the 6 reaction rate coefficients H.2 2.2.5, 2.2.9, 2.2.10, 2.2.11, 2.2.12 from the data-set HYDHEL, [**kn:Janev**] and H.2, H.8 rate coefficients for 2.2.14 from the data-set AMJUEL. (In versions older than Spring 2015 data: H.2 2.2.14 were taken from HYDHEL and H.8 was not available, but a hard coded approximation to H.8 was used internally. This older default is very close, but not strictly identical to the newer one.)
EIRENE again tries to identify the reaction products from the mass and nuclear charge number of the respective molecule IMOL (or test ion IION). In order to distinguish $D_2$ from $HT$, is also uses the name TEXTM (or TEXTI), by looking for the appearance of the character “D” in the name (then: $D_2$, or $D_2^+$) or for “H” or “T” (then: $HT$, or $HT^+$). The default reaction kinetics are set as if they would have been specified by the following species blocks:

```
* ATOMIC REACTION CARDS   NREACI=
7
  1 HYDHEL H.2  2.2.9      EI   0   2
  2 HYDHEL H.2  2.2.5      EI   0   2
  3 HYDHEL H.2  2.2.10     EI   0   2
  4 HYDHEL H.2  2.2.12     EI   0   2
  5 HYDHEL H.2  2.2.11     EI   0   2
  6 AMJUEL H.2  2.2.14     EI   0   2
  7 AMJUEL H.8  2.2.14     EI   0   2
* NEUTRAL MOLECULES SPECIES CARDS:   NMOLI SPECIES, NMOLI=
1
```

```
 1 D2          4   2   2   0   0   1   0   3
      1    115    113      0 00000
 -1.5400E 01   0.0000E 00   0.0000E 00   0.0000E 00
      2    115    121    000 00000
 -1.0500E 01   0.0000E 00   3.0000E 00   3.0000E 00
      3    115    111    *14 00000
 -2.5000E 01   0.0000E 00   5.0000E 00   5.0000E 00
* TEST ION SPECIES CARDS:   NIONI ION SPECIES, NIONI=
1
 1 D2+         4   2   2   1   0   1   0   3 -1
      4    115    111    *14 00000
 -1.0500E 01   0.0000E 00   4.3000E 00   4.3000E 00
      5    115    *24        00000
 -1.5500E 01   0.0000E 00   0.4000E 00   0.4000E 00
      6    115    121    000 30300
  7.0000E 00   0.0000E 00   7.0000E 00   0.0000E 00
```

As above, $*$ stands for the species index of the $D^+$ bulk ion.
An exact reproduction of an EIRENE run (identical test particle trajectories) with the hydrogen default model requires also the the sequence of processes to be the same as described here, due to random sampling of events at points of collision along Monte Carlo trajectories.
Note that for reaction 6, dissociative recombination of the $D_2^+$ molecular ion the energy dependence of the cross section leads to a mean electron energy loss per event $\overline{E}_{el}$ of about $0.896 \cdots T_e$, see reference [**kn:Reiter93**], rather than the perhaps expected $3/2 \cdot T_e$. This approximation is used in the default model for the electron energy loss $E_{el}$ per collision and for the total kinetic energy release $E_K$ shared by the two product atoms. The reaction card no. 7 above: AMJUEL H.8 2.1.14, together with the rate coefficient AMJUEL H.2 2.2.14 for this process, provides values identical to this default. Due to the linear form (on a log-log scale) of this AMJUEL H.2 rate coefficient, an algebraic manipulation of the AMJUEL fit coefficients results in the exact electron energy loss for this process, without independent fitting of energy weighted rates.

$$\langle \sigma v_e E_{el} \rangle = \overline{E_{el}} \cdot \langle \sigma v_e \rangle = kT_e \langle \sigma v_e \rangle \cdot \left( 3/2 + \frac{d \ln \langle \sigma v_e \rangle}{d \ln (kT_e)} \right) \tag{2.5}$$

The velocity distribution of the dissociation products is isotropic in the center of mass system (taken to be the system moving with the incident molecule), and the dissociation energy is shared by the dissociation products so as to preserve momentum. I.e., in case of unsymmetrical hydrogenic molecules the two dissociation products to not receive the same share of the dissociation kinetic energy release (KER), but instead KER is distributed inversely proportional to their masses.
In case of mixed molecules, such as $DT$, $HT$ or $HD$ some rate coefficients are automatically split into two half and assigned to the proper product atomic particles.

#### default model for hydrogenic ions

For hydrogenic bulk ions ($H^+, D^+, T^+$) default (radiative) volume recombination rates are available [**kn:Gordeev**]. The associated electron energy losses are derived in closed analytic form for this expression.

There is currently no fully equivalent counterpart to this in the AMJUEL or HYDHEL polynomial fits. Rather close to the minimal default volume recombination model, al least at low Te, is the choice:

```
C single step recombination rate to ground state
  1 HYDHEL H.2  2.1.8       RC   0   1
C corresponding electron energy loss/gain
  2 HYDHEL H.8  2.1.8       RC   0   1
```

For helium bulk ions ($He^+$) analytic default (radiative) volume recombination rates are given in [**kn:Janev**], reaction 2.3.13. The associated electron energy losses are derived in closed analytic form for this expression inside EIRENE. Again: no fully corresponding identical fits for these expressions are available currently in the external databases.

**Further examples**

to be written

### 2.4.3 Neutral–Neutral collisions in BGK! approximation

More generally: the **BGK!** approximation (section 1.9.1) is employed for (bilinear) test-particle - test-particle collisions processes. From the input flags IBGKA(KK), IBGKM(KK), IBGKI(KK) (see above, chapter 4) for a collision processes KK for atoms, molecules and test-ions, respectively, certain internal flags are set and options are activated to enable iterative solutions of the transport problem with these non-linear effects. This iteration is carried out for the parameters density, temperature and flow velocity of a virtual background species, one virtual species for each **BGK!** process KK. The **BGK!** collision integrals are then simulated simply by random sampling the post-collision velocity from a drifting Maxwellian of the corresponding virtual bulk species.

In input blocks 4a, 4b or 4c any test particle species ISPZ1 may have assigned to it one (or more than one) so called "**BGK!**" collisions, as identified by the process flag IBGK...(KK) .ne. 0. If that is the case, this test particle species is labeled, additionally to its label ISPZ1, also as a "**BGK!**-species" no. IBGK_SP and the corresponding internal flags NPBGKA(IATM), NPBGKM(IMOL) or NPBGKI(IION) are non-zero for this species. For each "**BGK!**-species" IBGK_SP automatically additional tallies, so called "**BGK!**-tallies" are scored, by a call to UPTBGK.F from scoring routine UPDATE.F. These additional tallies are then used to derive parameters for the iterated virtual background (plasma) species. For each **BGK!** collision process KK a "virtual" background species IPLS (input block 5) is assigned in a unique way. For these bulk particle species IPLS the internal flag IBGK = NPBGKP(IPLS,1) points to a particular **BGK!** collision process IBGK and hence also to a particular test particle species ISPZ1, which collides with this bulk species IPLS via that **BGK!** collision kernel no. IBGK.

In case of cross collisions a second internal flag ISPZ2= NPBGKP(IPLS,2) points to the second test particle species involved this is particular collision process IBGK.

In the example considered here we have the 4 neutral–neutral collision processes: D on D2, D on D and D2 on D2, D2 on D. In input block 4 there are three **BGK!** relaxation rate coefficients (note: the rate coefficient for D2 on D is equal to the one for D on D2, hence 3 rate coefficients for 4 processes are sufficient).

```
C     .
C     .   some 20 other reaction cards in this example
C     .
```

```
 21 CONST   H.2              EL   1   1
 -2.1091E+01   0.2500E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00
  0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00
 22 CONST   H.2              EL   2   2
 -2.0589E+01   0.2500E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00
  0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00
 23 CONST   H.2              EL   1   2
 -2.0357E+01   0.2500E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00
  0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00   0.0000E 00
```

In input block 4a (for atoms), we specify 4 collision processes for D atoms. Processes no. 3 and 4 are neutral–neutral collisions.

```
1 D          2   1   1   0   1 -1   1   4
C (i.e.: 4 processes to be specified for this D atom)
C      .
C      .  2 other process decks, e.g. CX and electron impact ionization
C      .
     21   2014    (=IBULK)    1001      0    111   (=IBGK, self-collision)
 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
     23   2214    (=IBULK)    1001      0    112   (=IBGK, cross collision with IMOL=1)
 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
```

In input block 4b (for molecules), we specify 7 collision processes for D2. Processes no. 6 and 7 are neutral–neutral collisions.

```
1 D2         4   2   2   0   0   1   0   7
C (i.e.: 7 processes to be specified for this D2 molecule)
C      .
C      .   5 other process decks.
C      .
     22   2114   (=IBULK)     1001      0    112    (=IBGK, self-collision)
 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
     23   2314   (=IBULK)     1001      0    111    (=IBGK, cross collision with IATM=1)
 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
```

In input block 5 there are at least 23 bulk ion species in this example. Of those species no. 20 to 23 are "artificial" **BGK!** species for neutral–neutral self-collisions specified above by IBULK = 2014, 2114, 2214 and 2314 in blocks 4a and 4b above.

```
C      .
C      .   19 other bulk ion species decks.
C      .
20 D(B)          2   1   1   0   1   1   0   0
21 D2(B)         4   2   2   0   1   1   0   0
22 DD2(B)        2   1   1   0   1   1   0   0
23 D2D(B)        4   2   2   0   1   1   0   0
```

### 2.4.4 Fitting expressions (IFTFLG)

For the databases HYDHEL, AMJUEL, H2VIBR and METHANE the default fitting expressions (IFTFLG=0) are single or double parametric polynomial expressions for the logarithm $\ln(F)$ of function $F$, in the independent variables $\ln(p_1), \ln(p_2)$, as described in [**kn:Janev**] or [**kn:Ehrhardt**] for the databases HYDHEL and METHANE, respectively.
The following options are available in recent versions of the code:
**a) for interaction potentials (elastic processes):**

IFTFLG = 0 : default, not in use
IFTFLG = 2 : generalized Morse potential as described in [**kn:Bachmann**]
**b) for cross sections, rate coefficients, rates:**
IFTFLG < 100 : rate coeff. or cross section, [$cm^3\ s^{-1}$] or [$cm^2$].
IFTFLG > 100 : rate = density * rate coeff, [s] (e.g. used for spontaneous decay, e.g. radiative decay.
mod(IFTFLG,100) = 10: only one fitting coefficient is read, i.e. the cross section, rate coefficient or rate is constant.
**c) for photonic processes (line shape profiles:**
to be written

## 2.4.5 Internal EIRENE A&M Data structures and conventions

This section is currently being written. . .

### 2.4.5.1 The CREAC array (prior to first SVN version 2008)

The array CREAC(1:9,-1:9,KK), KK=1,NREAC, in module COMXS is used to store (polynomial) fit coefficients for reaction no KK, as read in input block 4 from databases HYDHEL, AMJUEL, METHANE, H2VIBR. Other options, such as those for photonic processes, for internal CR codes, or for tabulated data, or parameters for asymptotic behaviour outside the fit validity range, had their own independent data structures.
For a given process KK the fits coefficients for:
interaction potentials were stored on CREAC(1:9,-1,KK),
those for cross-sections on CREAC(1:9,0,KK),
and those for Maxwellian averaged rate coefficients on CREAC(1:9,1:9,KK).

The second argument here controls either the particle energy (in eV) or plasma density (in particles $1 \times 10^{-8}\ cm^{-3}$) dependence in rate coefficients.
Higher moments, such as energy-weighted rate coefficients, have been read in and stored on CREAC with a different value of KK.
In 2008 all data structures for atomic/molecular or photonic processes or CR codes have been accommodated into a single, more general array of generalized data type REACTION_DATA

### 2.4.5.2 Data structure REACDAT (from SVN versions (2008))

Atomic, molecular and photonic collision data are stored internally on the data structure REACDAT, which replaces and generalizes the older fixed array format CREAC. The purpose is to accommodate also data formats other than the old polynomial fits read from data files AMJUEL, HYDHEL, etc., e.g. to allow also tabulated data (1D and 2D tables), in-build full CR codes, as well as the various input data formats for line broadening mechanism (line shape functions) in case of radiation transport.
REACDAT(KK=1,NREAC) is an array of datatype REACTION_DATA.
Each element of REACDAT(IR) holds

- logical variables: LPOT, LCRS, LRTC, LRTCMW, LRTCEW, LPHR, LOTH
- pointers: POT, CRS, RTC, RTCMW, RTCEW, PHR, OTH (the data themselves)

- real: ETH
- real: RTMAX, ERTMAX
- integer: NOSEC (number of secondary particle groups)

The logicals denote the presence or absence of different types of data for a particular collision the process KK.
logical variable internal code Corresponding pointer meaning LPOT POT interaction potential (or: differential cross section, scattering angle, . . . ) are available LCRS CRS Cross-section data is available LRCT RTC Rate coefficient is available LRTCMW RTCMW Momentum weighted rate coefficient is available LRTCEW RTCEW Energy rate coefficient is available LPHR PHR Photonic process (line-shape, Einstein coefficients, population escape factors. . . ) is available LOTH OTH Other reaction type data (reduced population coefficients, equilibrium density ratios, etc.)
If the logical variable is set to .FALSE. the corresponding pointer is "unassociated".
NOSEC specifies the number of secondary particles produced by the reaction.
Pointers POT, CRS, RTC, RTCMW, RTCEW, OTH, PHR are of datatype FIT_FORMS.
Datatype FIT_FORMS consists of an integer variable IFIT and 4 pointers POLY, TAB2D, LINE and CR. . .

- POLY is a pointer to a data structure for polygon fits (POLY_DATA)
- TAB2D (old versions: ADAS) is a pointer to TAB2D (old: ADAS) data (TAB2D_DATA or ADAS_DATA)
- LINE points to a data structure describing a photonic line (LINE_DATA)
- TAB1D (old versions: HYD) points to a data structure, out: unfinished codes removed in 2018)
- CR. . . points to a data structure used for collisional radiative models

Of the 5 pointers one and only one can be associated. IFIT indicates which one is set.
RC1MIN, RC1MAX, RC2MIN, RC2MAX: ranges for first and second parameter in the fit formula. Outside the range an approximation should be used IFEX1MN, IFEX1MX, IFEX2MN, IFEX2MX: option indicating approximation models FP1L, FP1R, FP2B, FP2T: fit coefficients for approximation model
Data structure POLY_DATA is used to store data belonging to polynomial fit. It consists of a 2 dimensional array DBLPOL for holding the fit coefficients. Variables RC1MN and RC1MX describing the validity range of the fit, flags IFEX1MN and IFEX1MX and array FPARM1(6) specifying extrapolation options.
Data structure TAB2D_DATA describes data read from database with tabulated rate coefficients, e.g. **ADAS!**. It consists of two integer variables NDENS and NTEMP denoting the number of densities and temperatures in the fit table. Arrays DENS and TEMP hold the densities and temperatures respectively. Array FIT contains the 2d fit. The arrays DDE and DTE are used to precalculate delta_DENS and delta_TEMP.
Data type LINE_DATA deals with all parameters specifying a photonic process. It comprises a reaction name REACNAME, profile parameters E0, E1, AIK, G1, G2, C2, C3, C4, C6, B12, B21

and flags IGND, IRCART, IPROFILETYPE, IFREMD, NRJPRT, IMESS. Additionally there are arrays C6A(12), IPLSC6(12), KENN(12).
Data type COLRAD holds all data needed to describe the available collisional radiative model such as IFLAV: type/flavour of the CR. . . model (H, H2, He, . . . ) IFORMUL: formulation used (1 or 2), i.e. meta-stable condensed or meta-stable resolved IROW_ESC, ICOL_ESC, POP_ESC: population escape factor and the matrix position in which the factor should be applied.

#### 2.4.5.3 The MODCOL array

:
The integer array IFLAG = MODCOL(CT, I2 = 0, . . . , 4, IR) is the internally used identifier for the various collision process options and related data availability. It is constructed in the initialization phase (routines XSTEI, XSTCX, XSTEL, XSTPI, ) of a run from the atomic/molecular data read in input block 4 (or hard wired from the minimal default atomic collision models, in case of NRC. . . = 0 for a certain test particle component).

The first index CT characterizes the type of a process. From this "type" parameter also certain default assumptions made in the code for this type of process are inferred:

**CT = 1** electron impact processes ('EI' in the names of corresponding variables, some older variable names also: 'DS').

H.2 or H.4 data required, optional: additional H.8 or H.10 data for electron cooling rates.

**CT = 2** not in use. In very old versions (1990 and older): DS, dissociation process, now combined with EI type processes

**CT = 3** charge-exchange ('CX' in the names of corresponding variables).

Fully kinetic collision integral, but fixed scattering angle $\pi$ in COM system. (H.1 and/or H.2 or H.3 data required. Optional: H.9 data)

**CT = 4** heavy particle collisions ('PI' in variable names)

**CT = 5** elastic processes ('EL' in variable names).

Either relaxation process approximation (H.2 data) or fully kinetic collision integral (H.0, H.1 and H.3 data required. Optional: additional H.9 data for mean ion energy loss). In versions from 2017 and later, H.0 may be omitted. The process is then is simulated as effective (condensed) scattering with an isotropic post-collision distribution in COM system, and the H.1, H.3 data are interpreted as diffusion cross section and rate coefficient, respectively.

**CT = 6** recombination processes ('RC' in variable names).

**CT = 7** photonic processes ('PH' in variable names, not fully implemented. ).

For any process IR one or more than one data set may be required. The availability of data read from input block 4 is coded as IFLAG = 0 (data not available) or else IFLAG $\neq$ 0. In the latter case the value of IFLAG contains further information on the format (independent parameters) of the corresponding data set and the resulting collision kinetics.

**I2 = 0** format/availability of interaction potential data (or of differential cross sections): H.0

**I2 = 1** format/availability of cross section data: H.1. Cross sections are needed, in addition to rate coefficients, mainly for heavy particle interactions: CX, EL, PI, for velocity space sampling of the incident background medium collision partner. Both the test particle and bulk particle velocities are then used for post-collision velocity space sampling (e.g. COM scattering angle, etc.)

**I2 = 2** format/availability of rate coefficient data: either H.2, or H.3, or H.4

**I2 = 3** format/availability of momentum averaged rate coefficient data: either H.5, or H.6 or H.7

**I2 = 4** format/availability of energy averaged rate coefficient data: either H.8, or H.9, or H.10

Meaning of IFLAG:
to be written

#### 2.4.5.4 storage saving mode

The storage saving mode: atomic/molecular data “on the fly” routines Available and tested (Jan 2016)
ftabcx3 for modcol(3,2,ir)=1 (H.2 cx rate coefficients vs. Ti)
ftabel3 not connected
ftabpi3 not connected
ftabei1 done

## 2.5 Input for Plasma Background

**General remarks**

The background medium (mostly plasma) consists of NPLS (sometimes in the code: NPLSI) different so called "bulk particle" species, also referred to as "bulk ions", by abuse of language. For each of these species arrays of the parameters: particle density, temperature, flow velocity (3 Cartesian components), are defined, one value per parameter and per grid cell. There are, in total, NSBOX cells in a run.
From the assumption of quasi-neutrality, the specified charge state of each background ion species (including charge state = 0 for neutral background species), and the data for density $n_i$ of background species labeled $i$, DIIN(IPLS,icell), an array of NSBOX data for the **electron density** $n_e(\mathrm{cm}^{-3})$ (DEIN(icell) is computed in each of the cells (see next subsection: "derived background data").
Further background tallies (or "input tallies") defined in this block are related to a magnetic field, electric field, cell volumes (unless computed by EIRENE defaults).
For each "background tally" ITALLY there is a flag INDPRO(ITALLY), which describes the type of profiles, the usage of input parameters, or how a tally is transferred from an external file or code into an EIRENE run. The meaning of variable INDPRO is described below.
I.e.: the background medium is described by several blocks of NSBOX data each, (i.e. one datum per grid cell), so called "input tallies", namely

- one array for the electron temperature $T_e$ (eV) (TEIN),
- either one common or NPLS distinct (one for each bulk ion species) arrays for the ion temperature(s) $T_i$ (eV) (TIIN),
- NPLS arrays (one for each bulk ion species), ion densities $n_i$ ($\mathrm{cm}^{-3}$) (DIIN),
- either one common or NPLS distinct (one for each bulk ion species) arrays for the (Cartesian) drift velocities $\mathbf{V} = (V_x, V_y, V_z)$ (VXIN, VYIN, VZIN), either in $\mathrm{cm\,s}^{-1}$ or Mach number units.

**Special case: Magnetic field input**

The unit vector $\mathbf{b} = (b_1, b_2, b_3)$ (BXIN, BYIN, BZIN), parallel to the magnetic field is set to (0.,0.,1.) by default, i.e. pointing in z- (or toroidal-) direction. The default magnetic field strength is $|\mathbf{B}| = 1$ T in all grid cells. This default setting may be overruled by using the *INDPRO(5)* flag and the corresponding input parameters (see below). Mostly the full magnetic vector field is read from external files or routines (*INDPRO(5)* $> 3$ options). For testing purposes also quite simple preprogrammed magnetic fields can be set (*INDPRO(5)* $\leq 3$ options). In this latter case either a (radial or x-) profile of a vector field $\mathbf{b} = (0.0, b_2, b_3)$ (which is parallel to the $x$ = const. "flux" surfaces) or, alternatively (since 2016) of a vector field $\mathbf{b} = (b_1, 0.0, b_3)$ can be defined. In these two cases a B-field "pitch" profile can be selected.
For example, in the first case, ($INDPRO(5) = 1, \ldots, 3$, and geometry level $LEVGEO = 1, 2, 3$), the profile input parameters $B1, \ldots, B5$ control the x- or radial profile pitch$(x) = b_2/B_{tot}$. From this a cell centered B-field unit vector is internally constructed, by assuming that the first set of coordinate surfaces (x- or radial surfaces) are flux-surfaces (B-field aligned and constant pitch).

The magnetic field direction then has components (0., pitch, $\sqrt{1 - \text{pitch}^2}$) in the three coordinate directions. 1D simulations (x-grid only) are then across the magnetic field.
An analogous procedure is applied in the second case, but there the pitch is defined as pitch$(x) = b_1/B_{tot}$ via the input profile parameters $B1, \ldots, B5$, allowing 1D simulations (x grid only) parallel to the magnetic field (PITCH = 1.0 = const).
Currently only $INDPRO(5) = 3$ profile option (piecewise constant profile) allows to define also the B-field strength (BFIN) [T] together with the field pitch, solely by the input flags $B1, \ldots, B5$ described below.
Alternatively and in all other options an external data source for the B-field vector in Cartesian coordinates ($INDPRO(5) = 4, \ldots, 7$, all geometry levels $LEVGEO$) is used directly. In this latter case the transfer of magnetic field data (BXIN,BYIN,BZIN and BFIN) for each cell into EIRENE proceeds from user supplied profile subroutine PROUSR ($INDPRO(5) = 5$) or, ($INDPRO(5) = 4$) from external file or, ($INDPRO(5) = 6, 7$), via work array RWK, and then from subroutine PROFR).

#### Other input tallies: ADIN, VOL, WGHT

The same applies to the "additional plasma profile array" ADIN, $INDPRO(6)$, to permit usage of EIRENE output and graphics routines for additional (derived) plasma profiles such as pressure, energy fluxes, collision rate coefficients (see input block 14 for the latter), etc., which are not directly needed in an EIRENE run but which may be available from an external plasma code, or are derived from the atomic or molecular data sets included in a particular run. For all those: $INDPRO(6) = 5, \ldots, 7$ in input block 5 only, or the options in input block 14 (routine AMDIAG.f).
It also applies to the space and species dependent weight function WGHT (to be used for non-analog sampling techniques, INDPRO(7)), which is defaulted to 1.0D0 for all cells and all NSPZMC test particle species: NSPZMC = NATM + NMOL + NION + NPHOT. WGHT is currently unused, and could in principle be eliminated from all current runs, in particular when storage limitations are a concern.
It furthermore also applies to the zone volume array VOL cm$^{-3}$, $INDPRO(12)$. Defaults are the cell volumes computed in EIRENE subroutine VOLUME from the standard mesh data.
Plasma data in the additional zones IADD, IADD=NSURF+1,NSURF+NADD outside the standard mesh are defaulted to the "EIRENE vacuum data" given below. Plasma data other than these "vacuum values" (meaning: no collision processes in such cells) in these zones *have* to be specified either explicitly in input block 8 (see below), or in the user supplied subroutine PLAUSR or with the INDPRO=7 option (see below, this section).
Cell volumes in the additional cell region are defaulted to 1.0 cm$^{-3}$, unless $INDPRO(12) = 4, 5, 6, 7$.

#### Vacuum zones: no collision processes

A standard mesh zone ICELL is automatically identified as a vacuum zone with regard to plasma species IPL (no collisions with the bulk particles IPL there, *LGVAC(ICELL,IPL)=.TRUE.*, for electrons: IPL=NPLS+1)
if at least one of the following conditions is met:
for electrons: IPL=NPLS+1:
TEIN(ICELL) $\leq$ TVAC,
DEIN(ICELL) $\leq$ DVAC

for bulk ion species IPL=IPLS:
TIIN(ICELL,IPL) ≤ TVAC,
DIIN(ICELL,IPL) ≤ DVAC
and for all background species, IPL=0:
LGVAC(ICELL,I)=TRUE for I=1,NPLSI and for I=NPLS+1.
The EIRENE vacuum data are defaulted (in subroutine PLASMA) to:
DVAC = 1.0000E 02 $cm^{-3}$
TVAC = 2.0000E-02 eV
Furthermore, zero plasma drift velocities are set in cells which are treated as vacuum zones.
VVAC = 0.0000E 00 cm $s^{-1}$
for all three Cartesian components and for all NPLS background particle species.
Note: A cell can be considered a vacuum cell with respect to electrons, (LGVAC(. . . ,NPLS+1) = TRUE), without being a vacuum cell for all background ions. This is because, by abuse of language, also neutral particle species may be used as background "ion" species (NCHARP = 0), to include neutral–neutral collisions, e.g., by the iterative option NITER (input block 1).

**The Input Block**

```
*** 5. Data for Plasma background
      NPLSI
      DO 51   J= 1, NPLSI

* read NPLSI species blocks with $ = P

        ISPZ$ TEXT$ NMASS$ NCHAR$ NPRT$ NCHRG$ ISRF$ ISRT$
     .    ID1$
        NRC$  NFOL$ NGEN$  NHSTS$ ID3$
C      (format: I2,1X,A8,1X,12(I2,1X))

C End of NPLSI species cards reading
   51 CONTINUE

      (INDPRO(J), J=1,12)
      IF (INDPRO(1).LE.5) THEN
        TE0  TE1  TE2  TE3  TE4  TE5
      ENDIF
      IF (INDPRO(2).LE.5) THEN
        (TI0(I) TI1(I) TI2(I) TI3(I) TI4(I) TI5(I) I=1,NPLSI)
      ENDIF
      IF (INDPRO(3).LE.5) THEN
        (DI0(I) DI1(I) DI2(I) DI3(I) DI4(I) DI5(I) I=1,NPLSI)
      ENDIF
      IF (INDPRO(4).LE.5) THEN
        (VX0(I) VX1(I) VX2(I) VX3(I) VX4(I) VX5(I) I=1,NPLSI)
        (VY0(I) VY1(I) VY2(I) VY3(I) VY4(I) VY5(I) I=1,NPLSI)
        (VZ0(I) VZ1(I) VZ2(I) VZ3(I) VZ4(I) VZ5(I) I=1,NPLSI)
      ENDIF
      IF (INDPRO(5).LE.5) THEN
```

```
        B0   B1   B2   B3   B4   B5
      ENDIF
      IF (INDPRO(12).LE.5) THEN
        VL0   VL1   VL2   VL3   VL4   VL5
      ENDIF
C     ''Option-Card'' for additional information for input tallies
C     only for EIRENE versions starting from
C     Eirene_revised_input_tallies
```

If an "Option-Card" is encountered then read an arbitrary number "input tally option cards" of the format

```
ITAL  IOPT
```

Currently option INDPRO=6 and INDPRO=7 are not available for cell volumes (input tally no. 12).

**Meaning of the Input Variables for Plasma Parameters**

The meaning of the variables in the bulk ion species cards is as in block 4 (section 2.4) for the test particle species cards specified there.

**New options since 2003:**

However, after the ID3 flag, there may be 2 additional input flags: (only for $\text{Eirene}_{2003}$ and younger):

```
... CDENMODEL NRE
```

Format: ...,1X,A10,1X,I2.

These flags, if included, control additional options to set the density, temperature and flow field of the selected species IPLS. If the string CDENMODEL is found here for a species IPLS, then, after reading the species and reaction cards for IPLS, one (default) or NRE further input cards are expected.

**CDENMODEL**

**= 'Multiply'** docu to be written

**= 'Constant'** docu to be written

**= 'fort.13'** Read data for this species from file fort.13.

The additional card (only one) reads (format 2I6):

```
ISP  ITP
```

ISP is the number of a bulk species on the file fort.13 (e.g. written in an earlier EIRENE run, see NFILEL option in input block 1). The parameters (fields of density, temperature, flow velocity) for species IPLS are set from those of species ISP on fort.13

ITP is the type of the particle, i.e., always: ITP=4 for this option. Internally set: ITP=4, bulk particle type, independent of input value for ITP.

**= 'fort.10'** Read data for this species from file fort.10.

The additional card (only one) reads (format 3I6):

ISP ITP ISTR

ISP, ITP is the number and type of a test particle species on the file fort.10, respectively, from stratum no. ISTR (i.e. written onto fort.10 either in the present run or an earlier EIRENE run, see NFILEN option in input block 1). The parameters (fields of density, temperature, flow velocity) for species IPLS are derived from those of species ISP, ITP, ISTR on fort.10.

Note: at the end of a full EIRENE run the corresponding bulk particle profiles are reset to those evaluated in this run, for further use in post processing routines such as the diagnostic module (see section 2.11), printout, plotting or iterative mode (input flags NITERI, NITERE, and user subroutine MODUSR, see section 2.1)

**= 'Saha'** Set parameters from Saha-equilibrium of ionization states. The additional card (only one) reads:

ISP ITP ISTR ...

to be written

**= 'Boltzmann'** Set parameters from Boltzmann-equilibrium of excited states. The additional card (only one) reads (format: 3I6,6x,2E12.4)

ISP ITP ISTR G_BOLTZMANN DELTA_E

docu to be written

**= 'Planck'** (only for background photons) to be written

**= 'Corona'** Set density parameters from Corona-equilibrium to a "donor species", using an excitation rate coefficient from atomic database FILNAM and a fixed radiative decay rate $A = A_{CORONA}$. The additional card (only one) reads (format: 3I6, 1X, A6, 1X, A4, A9, A3, E12.4):

ISP ITP ISTR FILNAM H123 REACTION CR A_CORONA

ISP, ITP is the number and type of a (donor) particle species, either one of the already defined bulk species IPLS: ISP=IPLS, (for ITP=4), or from the file fort.10 (for ITP=0,1,2 or 3) from stratum no. ISTR (i.e. written onto fort.10 either in the present run or an earlier EIRENE run, see NFILEN option in input block 1). I.e. the density $ni(IPLS)$ at each position $x$ (grid cell) for this background species IPLS is specified via:

$$ni(IPLS, x) = n_{IS}(x) \times \frac{Rate_{IS}(x)}{A}$$

Here $Rate_{IS}$ is the rate for excitation from species $IS$ to $IPLS$ from the A&M data file $FILNAM, \ldots$, and $A$ is a radiative decay rate $A = A_{CORONA}$ read from this input card. The parameters (fields of density, temperature, flow velocity) for donor species IS are taken to be those of species [ISP, ITP, ISTR].

I.e.: this model constructs a background species IPLS which is in Corona-equilibrium with these IS = [ISP,ITP, ISTR] donor particles.

**= 'Colrad'** set parameters density, temperature and flow velocity from collisional radiative equilibrium with one or more (NRE) "donor-species", using reduced population coefficients from atomic database FILNAM. I.e. the density ni(IPLS) at each position $x$ (grid cell) for this background species IPLS is specified via:

$$ni(IPLS, x) = \sum_{IS} n_{IS}(x) \times POP_{IS}(x)$$

The NRE (for NRE different donor-states) additional cards read:

ISP ITP ISTR FILNAM H123 REACTION CR

ISP, ITP is the number and type of a particle (donor) species $IS$, either one of the already defined bulk species IPLS: ISP=IPLS, (for ITP=4, then ISTR is irrelevant), or from the file fort.10 (for ITP=0,1,2 or 3) from stratum no. ISTR (i.e. written onto fort.10 either in the present run or an earlier EIRENE run, see NFILEN option in input block 1). The parameters (fields of density, temperature, flow velocity) for donor species IS are set from those of isotope [ISP, ITP, ISTR]. And by multiplication with a reduced population coefficient $POP$ specified via $[FILNAM, H123, \ldots]$ these are turned into excited state or other (ionised state) populations. The $POP$ coefficients are typically from externally applied collisional radiative models.

I.e. this model constructs a background species IPLS which is the sum of NRE species, each of which being in local collision-radiative-equilibrium with its donor isotope $IS$.

**INDPRO** Flag-array for the type of profile. The last digit (between 1 and 9) controls the type of profile. A second digit and/or the sign control further options, as described below. A third digit for each entry can be used to switch from a cell-wise constant profile (default) to a smooth (interpolated into cells) profile. This option is currently being tested for the magnetic field input profiles. I.e. INDPRO(5)=103 provides the same magnetic field as INDPRO(3)=3, however the magnetic field is evaluated on the fly (along particle trajectories) by interpolation, at the very point of particle position rather than at the cell center (=const. per cell).

Options INDPRO = 1, 2, 3 are preprogrammed 1 D profiles with the first coordinate (x or radial) taken as independent parameter. I.e. profiles are constant in the other 2 coordinate directions: y and z.

Options INDPRO = 4, 5, 6, . . . allow more general, and higher dimensional profile forms.

INDPRO is an array of length 12. Each element in this array controls one particular input tally, namely:

**INDPRO(1)** for TEIN

**INDPRO(2)** for TIIN By the default (*0 < INDPRO(2) < 10*): one common $T_i$ profile for all "bulk ion" species is set. I.e., read only one profile card.

**New options since 2001:**

Values of INDPRO(2) larger than 10: use only the last digit, and one $T_i$ profile card must be read for each bulk ion species IPLS. For example: *INDPRO(2)=15* or *=25*, means: one separate ion temperature must be specified for each bulk ion species, and the profile type is 5. (*INDPRO(2)=-5* would do the same.)

**INDPRO(3)** read NPLSI cards, for DIIN, IPLS = 1,NPLSI

**INDPRO(4)** read NPLSI cards, for VXIN, VYIN, VZIN, IPLS = 1,NPLSI

**New options since 2001:**

By the default (*0 < INDPRO(4) < 10*): one separate flow field for each bulk species is set, velocity is given in $\mathrm{cm\,s^{-1}}$.

Negative value of INDPRO(4) means: Mach number units instead. The sound speed *cs* is taken to be the isothermal ion acoustic speed of species IPLS.

*ABS(INDPRO(4)* is then used as flag for the choice of profile type.

Values of INDPRO(4) larger than 10: use only the last digit, and only one common flow field is set for all bulk ion species. Note the difference to the $T_i$ options: there the meaning of INDPRO larger than 10 was exactly opposite to the meaning here for the flow fields (due to historical reasons and for backward compatibility of input files. EIRENE had originally by default one single common ion temperature, but one flow field for each bulk ion species).

**INDPRO(5)** for a 1D (radial or x-) profile of PITCH (later, in initialization phase, converted into Cartesian unit B-field vector BXIN,BYIN,BZIN)

By the default (*0 < INDPRO(5) < 10*) the pitch is defined as $B_y/B_{tot}$ (LEVGEO=1), $B_\theta/B_{tot}$ (LEVGEO=2), or as $B_{pol}/B_{tot}$ (LEVGEO=3), where $B_{pol}$ is the direction along the polygons.

Negative values of INDPRO(5) (only for LEVGEO=1): use *ABS(INDPRO(5)*, and the pitch is then defined as $B_x/B_{tot}$ (only for LEVGEO=1), allowing 1D simulations, using the first (x-) grid only, along the B field. This option was only added during 2016.

**New options since 2001:** In case INDPRO(5)=3 (flat profile) the two redundant input parameters B2, B3 are used to define a constant B-field strength [T], see below under profile type INDPRO=3.

**INDPRO(6)** for ADIN

**INDPRO(7)** for WGHT (to be written)

**INDPRO(12)** for VOL.

For the input tally profiles no. 6 and 7 (ADIN, WGHT) only the options INDPRO=5 or INDPRO=6 exist.

For the profile no. 12 (cell volumes) the options INDPRO(12) = 4, 5, 6, or INDPRO(12) = 7 are active options. For all other values of INDPRO for these latter four profiles the default profiles described above ("General remarks") are set.

For each profile up to 6 parameters P0, ..., P5 are read, e.g. TE0, ..., TE5 for the $T_e$-profile, TI0, ..., TI5 for the $T_i$-profile, and so on.

Depending upon the value of INDPRO one of the profile routines PROFN, PROFE, PROFS, etc. is called from subroutine PLASMA.

**INDPRO = 1-4**

NR1STM plasma data are defined on the one- dimensional zone-centered grid

RHOZNE(J), J=1, . . . ,NR1STM (NR1STM = NR1ST - 1 ) .

Vacuum data are specified in zone NR1ST.

These NR1ST data are copied NBLCKS times to define

NBLCKS identical profiles, totally : NBLCKS · NR1ST = NSURF data (subroutine MULTI).

Here the number of copies is NBLCKS = NMULT · NP2ND · NT3RD. see "Input Data for Standard Mesh" (block 2).

**INDPRO = 1**

(see subroutine PROFN)

Parameter P5 must be a position inside the first (radial) grid

RHOSRF(1) ≤ x ≤ P5:

$$P(x) = P1 + (P0 - P1) \cdot \left(1 - \left(\frac{x - RHOSRF(1)}{P5 - RHOSRF(1)}\right)^{P2}\right)^{P3}$$

in particular : $x = RHOSRF(1) \rightarrow P(x) = P0 x = P5 \rightarrow P(x) = P1$

P5 ≤ x ≤ RHOSRF(NR1ST) :

P(x) = P1 · exp(- (x - P5) / P4)

**INDPRO = 2**

(see subroutine PROFE)

Parameter P5 must be a position inside the first (radial) grid

RHOSRF(1) ≤ x ≤ P5 :

P(x) = P0 · exp(- (x - P1) / P2)

in particular : x = P1 → P(x) = P0

P5 ≤ x ≤ RHOSRF(NR1ST) :

P(x) = P(P5) · exp(- (x - P5) / P4)

(P3 : no meaning)

**INDPRO = 3**

(see subroutine PROFS, PROFS(P0,P1,P5,PVAC)

Parameter P5 must be a position inside the first (radial) grid

RHOSRF(1) ≤ x ≤ P5 : P(x) = P0

P5 ≤ x ≤ RHOSRF(NR1ST) : P(x) = P1

Parameters P2, P3, P4 have no meaning, except for the magnetic-field (INDPRO(5) = 3): Here PROFS(P0, P1, P5, PVAC) is used to define the pitch angle profile, and it is also used to set the B-field strength (T) by calling PROFS(P2, P3, P5, PVAC)

**INDPRO = 4**

read from input stream fort.P0

**INDPRO = 5**

EIRENE calls a user supplied plasma profile routine:

Subroutine PROUSR (RHO, INDEX, P0, P1, P2, P3, P4, P5, PVAC, NDAT).

NDAT data must be defined on

RHO(J), J = 1, NDAT e.g. by using the profile parameters P0, . . . , P5 and any user supplied profile function. See section 3.6.

(NDAT=NSURF)

The flag INDEX is as for the "INDPRO = 6" option, see below, and section 3.6 and chapter 4. It determines, which of the background medium profiles has to be provided at a particular call to subroutine PROUSR.

PVAC is the EIRENE vacuum value for the profile in question.

**INDPRO = 6**

EIRENE calls a the routine:

Subroutine PROFR (RHO,INDEX,N1I,N1,NDAT).

N1I (e.g., NPLSI) profiles, of NDAT = NSURF data each, are read onto RHO(I1I, J), J = 1, NDAT, I1I = 1, N1I from a work array RWK. N1 (e.g., NPLS) is the leading dimension of the array PRO as specified in the calling program. The data must be written onto array RWK in the initialization phase, e.g., in subroutine INFCOP or MODUSR. The location of a particular profile on this work array is determined by the value of INDEX, see chapter 4 below. By this option plasma parameters may directly be transferred into EIRENE from other files, e.g. from plasma transport codes.

NDAT (= NSURF) is the number of cells in the "Standard Mesh". Parameters in the additional cell region

ICELL = NSURF + 1, . . . , NSURF + NRADD

are set to the "EIRENE vacuum values"

For further details of subroutine INFCOP see section 2.1 ("Input Data for operating mode"), input variable NMODE and section 2.14 and chapter 4 for an example.

**INDPRO = 7**

Same as INDPRO = 6 option, except that NDAT = NSBOX rather than NDAT = NSURF. Hence in this case background data are read from RWK also for the additional cell region

ICELL=NSURF+1,NSURF+NRADD.

**INDPRO = 8**

to be written

For EIRENE versions starting from Eirene_revised_input_tallies additional input for use with input tallies is available.

**"Option-Card"** If EIRENE finds a card containing the string "OPTIONAL" it assumes that there might be an arbitrary number of input cards describing additional options for input tallies.

**ITAL** index of input tally, -NTALI ¡ ITAL ¡ 0

**IOPT** option flag

< 0 switch off tally ITAL

= 0 keep default

= 1 explicitely switch on tally ITAL

= 2 prepare for smoothing, interpolate tally ITAL to cornerpoints

= 3 switch on gradients for tally ITAL

### 2.5.1 Derived Background Data

Given the data describing the background medium (plasma) in each cell (on common block COMUSR) a set of further, "derived background data" profiles is computed and stored also on COMUSR. This is done by a call to subroutine PLASMA_DERIV. The following quantities are defined there:

**DEIN** Electron density ($cm^{-3}$), derived from all NPLSI ion densities and the charge states NCHRGP(IPLS) of each ion species.

**EDRIFT** Kinetic energy (eV) in drift motion, for each background species IPLS, derived from the mass MASSP(IPLS) and the flow velocity VXIN,VYIN,VZIN.

**LGVAC** Vacuum flag (logical) to identify regions, in which no collision data for all or certain background species are computed, i.e., the collisionalities are set to zero in these cells for those background species. See also explanation above, with respect to the EIRENE vacuum data TVAC,DVAC,VVAC.

**NSTGRD** flag for special treatment of some selected cells

= 0 default

= 1 This cell is a dead cell, not to be seen by the particles (isolated from the computational domain. NSTGRD(ICELL)=1 can be specified in problem specific routines (. . . USR) or in routines interfacing EIRENE with external codes: INFCOP (code segment: COUPLE. . . )

= 2 indicates that this cell belongs to a grid cut, i.e., is not a valid cell for particle tracing.

= 3 indicates that this cell is not a real cell but only the storage position for spatially averaged tallies. E.g.: cell no. NR1ST, 2*NR1ST, etc.

## 2.6 Input Data for Surface Interaction Models

**General remarks**

An outline of surface interaction models in general terms was given in section 1.4. As for the implementation of such models in EIRENE, there are two parts to the surface interaction data block. The first part contains data which are general to the EIRENE reflection model ("Block for General Reflection Data"), at all surfaces. The second part may be different for each individual surface element (label: $MSURF$) and thus must be specified for each reflecting "non-default" surface of the "standard mesh" and for each reflecting "additional surface". It is referred to as "Block for Local Reflection Data". If this block is missing and if the surface $MSURF$ is neither transparent ($ILIIN < 0$) nor purely absorbing ($ILIIN = 2$) nor a "mirror surface" ($ILIIN = 3$) nor a "periodicity surface", ($ILIIN \geq 4$) then the default reflection model (i.e. $Fe - Target$, recycling coefficient = one, etc.) is activated for this surface.
The surface interaction model comprises the following information:

1. $p_f$ (RPROBF): probability for reflection of a "fast" particle
   (= 0. for incident molecules),
2. $p_t$ (RPROBT): probability for re-emission as "thermal particle"
   (with surface temperature, see the flags EWALL)
3. $p_a$ (RPROBA): probability for absorption
   (surface sticking, pumping, etc.)
4. $p_s$ : probability for (physical) sputtering
5. $p_c$ : probability for (chemical) sputtering
6. For the reflected "fast particles" and for the sputtered particles the type, species and the distribution of energy, polar and azimuthal angle of emitted particles must be specified.
7. For the re-emitted thermal particles the type, species and only the distribution of energy must be given; the angular distribution follows the cosine law by default.

All these data may be functions of incident energy $E_{in}$, of incident polar angle $\theta_{in}$ against the surface normal and of projectile and target species. The three probabilities $p_f$, $p_t$ and $p_a$ must add up to 1 (one); $p_s$ and $p_c$ are sputtering yields per incident particle and, thus, may occasionally be larger than 1. They are called a probability here only by abuse of language.
All this information is contained in each of three so called "Reflection Models", namely the "Database Reflection model" (see references [**kn:Eckstein**], [**kn:Bateman**]), the "Modified Behrisch Matrix model" (references [**kn:Behrisch**], [**kn:Nicolai**]) or the "User Supplied Reflection model" (see section 3.3), and in several so called "Sputter Models" ([**kn:NucFus**], [**kn:Sputer95**], [**kn:Roth96**].
These models may, to some extend, be modified by the input flags described here. For example, a surface may be purely absorbing (for all test particles incident onto it, $ILIIN = 2$ option), acting like a mirror for the neutral test particles, by the $ILIIN = 3$ option (see section 2.3A and 2.3B), or enforce periodicity, e.g., because of symmetry conditions ($ILIIN \geq 4$ option). In these two latter

cases all test particles of all species are reflected with the species unchanged, with probability one and elastically: $E_{out} = E_{in}$. In case of the "mirror reflection option" the normal component $\mathbf{v}_n$ of the incident velocity vector $\mathbf{v}$ is reversed: $\mathbf{v}_n \rightarrow -\mathbf{v}_n$. In case of a "periodicity surface" both the position and the speed unit vector are altered according to the specific periodicity of the configuration.
If, instead, a surface reflection model is chosen ($ILIIN = 1$), then some of the input flags described here (RECYCF, RECYCT, RECYCS, RECYCC) as well as the flags (RINTEG, EINTEG, AINTEG) may be used to modify conveniently the otherwise rather unhandy reflection data and formulas.

**The Input Block for General Reflection Data**

```
*** 6A. Data for General Reflection Models
      NLTRIM
C     ''Path-Card'' for TRIM surface reflection files
C     (machine specific, may be omitted)
```

If a "Path Card" is included, then read an arbitrary number of "Target-Projectile Specification Cards" of the format

```
A_on_B
```

Up to NHD6 (see PARAMETER statements, section 3.1) data files for different Target-Projectile combinations can be read in EIRENE versions older than 2001. No such limitation exists in more recent versions (due to dynamic allocation of storage).
If no "Path Card" has been specified, or after the "Target-Projectile Specification Cards", continue reading general species sampling distributions, used for source particle species sampling as well as for sampling post-surface event species as part of the surface reflection models

```
      (DATD(I)  I=1,NATM)
      (DMLD(I)  I=1,NMOL)
      (DIOD(I)  I=1,NION)
      (DPLD(I)  I=1,NPLS)
C (this card is to be read only in case NPHOT > 0)
      (DPHT(I)  I=1,NPHOT)
      ERMIN  ERCUT  RPROB0  RINTEG  EINTEG  AINTEG
```

Note that the species index distribution cards must always be read, even if NATM, NMOL, NION or NPLS are zero. In this latter case at least one (irrelevant) parameter must be read. E.g. if NION=0, i.e. no test ion species in particular run, then still one card with the irrelevant parameter DIOD(1) must be read. An exception from this rule is the photon species distribution DPHT: this card is only read if there are "photonic test particles" in the run, i.e. if NPHOT > 0. This exception was necessary for backward compatibility of input file format after introducing photons as new type of species, around 2001.
Next, optionally, an arbitrary number of surface models, labelled by the character string modname, may be read.

```
SURFMOD_modname
ILREF  ILSPT  ISRS  ISRC
ZNML  EWALL  EWBIN  TRANSP(1,N)  TRANSP(2,N)  FSHEAT
RECYCF  RECYCT  RECPRM  EXPPL  EXPEL  EXPIL
```

```
C (following line may be omitted, then: default sputter
C  model, see below)
      RECYCS  RECYCC  SPTPRM  ESPUTS  ESPUTC
```

next: within each SURFMOD-sub-block there may be an arbitrary number of lines

```
       VARNAME  SPZNAME    VALUE
C      (format: CHARACTER*8,X,CHARACTER*8,*)
```

**Meaning of the Input Variables**

**NLTRIM** TRIM database is used, if "Database Reflection Model" is specified in at least one block for local reflection data. Data are read from data-set FT21 (no "Path Card" specified, old option), or from the domain specified by the "Path Card" (new option, see next card). If the old option is used, then the complete TRIM file is read, containing the first 12 TRIM target-projectile combination data-sets listed in section 1.4, i.e., the files H_on_Fe to T_on_W.

If the parameter NHD6 < 12, (section 3.1) then only the first NHD6 files are read from FT21.

**"Path Card"** This card is machine specific. If EIRENE finds a card containing the string 'PATH' or 'path', it assumes that this card specifies the path to the domain containing the TRIM surface reflection data files.

**A_on_B** Name of a particular TRIM data file in the domain specified by the path card. E.g., H_on_Fe would include the data file for hydrogen onto iron into the EIRENE run. Up to NHD6 such "Target-Projectile Specification Cards" may be included. For a complete list of such files currently available see again section 1.4.

Note: if during a Monte Carlo simulation a projectile A hits a target B, for which no TRIM data file has been specified, but still NLTRIM=TRUE, then EIRENE searches the "closest" of all its TRIM data files (with respect to a reduced mass argument) and uses this target-projectile combination together with a reduced mass scaling of incident particle energy. Hence, a TRIM data set is chosen such that the reduced mass scaling factor is as close to one as possible amongst the files available.

**DATD** distribution for sampling the species index of reflected or otherwise emitted atoms. The NATMI relative frequencies

DATD(IATM) IATM = 1,NATMI

are used to produce the corresponding cumulative distribution DATM in order to facilitate sampling (inversion method). Normalization of DATD such that

$\sum_{IATM} DATD(IATM) = 1$

is carried out internally.

**DMLD** as for DATD, but for molecules. Cumulative distribution is DMOL.

**DIOD** as for DATD, but for test ions. Cumulative distribution is DION.

**DPLD** as for DATD, but for bulk ions. Cumulative distribution is DPLS.

The activation of these distributions at surface events during the particle history generation process is controlled by the surface species index flags ISRF$, ISRT$ read in the species sub-blocks of block 4 and 5 (sections 2.4 and 2.5). By default the following conventions are used (and can be overruled by calls to subroutines REFUSR, SPTUSR only, see section 3.3):

**ISRF$** fast particle reflection species flag:

> 0 If ISRF$ ≤ NATMI, then ISRF$=IATM, the species labelling index for the reflected fast atom.

If ISRF$ > NATMI not in use, warning and error exit.

= 0 no fast particle reflection for this species: $p_f = 0$

< 0 not in use, warning and error exit.

**ISRT$** thermal particle re-emission species flag:

**incident atoms, test ions and bulk ions:**

> 0 If ISRT$ ≤ NATMI, then ISRT$ = IATM, the species labelling index for the re-emitted thermal atom.

If ISRT$ > NATMI, then IATM is sampled from the distribution DATM(IATM).

= 0 no thermal particle re-emission for this species: $p_t = 0$

< 0 molecule is re-emitted, and -ISRT$ is used to identify the species labelling index IMOL for the re-emitted molecule, exactly as described above for re-emitted atoms. The relevant sampling distribution for species index IMOL in case -ISRT$ > NMOLI is DMOL(IMOL). This option can e.g. be used to specify the distribution of vibrationally excited molecules $H_2(v)$ emitted from a surface after bombardment by incident $H$ atoms, the so called Eley Rideal mechanism: Surface(with implanted $H$)+$H \rightarrow H_2(v)$

**incident molecules:**

> 0 If ISRT$ ≤ NMOLI, then ISRT$ = IMOL, the species labelling index for the re-emitted thermal molecule.

If ISRT$ > NMOLI, then IMOL is sampled from the distribution DMOL(IMOL).

= 0 no thermal particle re-emission for this species: $p_t = 0$

< 0 not in use, warning and error exit: "species **index** out of range"

In code versions older than year 2000 this ISRT<0 options was also used to activate sampling the new species index IMOL from distribution DMOL. In order to avoid un-intentional species sampling at surfaces this option was disabled from year 2000 on.

These latter two surface-species index flags are considered to be particle properties, hence they are read in the particle specification blocks 4 and 5 (sections 2.4 and 2.5). The flags in the next card can be used to modify the preprogrammed fast particle reflection models, to some extend at least.

**ERMIN** For incident particle energies below ERMIN, the "fast" particle reflection model is switched off. Only the "thermal" particle model is used.

**ERCUT,RPROBF**

These variables may be used to modify the default "Behrisch Matrix" reflection coefficients for particles incident on a surface at low energies $E_{in}$. The original data [**kn:Behrisch**] are used only for $E_{in} >$ ERCUT and for normal incidence $\vartheta_{in} = 0$.

In the range ERMIN $< E_{in} <$ ERCUT the Behrisch Matrix particle reflection coefficient $p_f(E_{in}, \vartheta_{in} = 0)$ is replaced by a smooth cubic interpolation curve $p_f(E_{in})$ such that $p_f(E_{in}, \vartheta_{in} = 0)$ = RPROBF .

The original "Behrisch Matrix" is recovered by setting ERCUT $\leq 0$.

The next three flags modify the particle, energy and momentum reflection coefficients, and/or the resulting distributions in post-reflection energy and angle, respectively. These flags only apply to incident particles of type 1, 3, 4 and 5, i.e., not for incident molecules because for those RPROBF = 0 by default.

**RINTEG**

$> 0$ Fixed (independent of energy and angle of incidence) particle reflection coefficient.

The fast particle reflection probabilities RPROBF are set to $p_f = MIN(1-p_a, RINTEG)$, regardless of the reflection model selected by the flag ILREF in block 6B. $p_a$ is kept as specified, and the thermal component probability $p_t$ is then recomputed as $p_t = 1 - p_f - p_a$.

RINTEG $\geq 1 - p_a$ enforces the fast particle reflection model for all un-pumped incident particles, i.e., RPROBF = RINTEG is internally reset to $1 - p_a$.

$= 0$ Default: fast particle reflection probability as defined by the reflection model chosen.

$< 0$ The fast particle reflection probabilities RPROBF are set to $1.0 - p_a$. Hence: same as RINTEG=1.0

**EINTEG**

$> 0$ Fixed (independent of energy and angle of incidence) energy reflection coefficient.

This choice replaces the energy random sampling procedure in the fast particle reflection model by an reflection assumption: $E_{out} = E_{in} \cdot EINTEG$.

$= 0$ Default: no modification of energy distribution in the reflection model for fast particle reflection.

$< 0$ elastic ($E_{out} = E_{in}$) reflection for all particles reflected according to the fast particle reflection model. Hence: same as EINTEG=1.0

**AINTEG**

$> 0$ Fixed (independent of energy and angle of incidence) momentum reflection coefficient ("accommodation coefficient") $\alpha = 1 - AINTEG$

This choice replaces the angle random sampling procedure in the fast particle reflection model by activating features of the so called "Maxwell's boundary model"

(Philosophical Transactions of the Royal Society, 1879), in which a fixed fraction AINTEG ($\leq$ 1.0) is reflected at a specular angle. The remaining "evaporated" fraction (1-AINTEG) is reflected with a cosine distribution (Lambertian). In the original Maxwell boundary model the evaporated fraction is re-emitted according to an inward directed Maxwellian flux distribution (taken at wall temperature) while the specular fraction is with unchanged energy $E_{in} = E_{out}$. This full Maxwell boundary condition with accommodation coefficient $\alpha$ = 1−AINTEG is specified with the additional choices: EINTEG=1.0, EWALL =-TW (TW the wall temperature, see further below: EWALL flag.

= 0 Default: no modification of angular distributions in reflection model for fast particle reflection.

< 0 specular reflection for all particles reflected according to the fast particle reflection model. Hence: same as AINTEG=1.0

**Note:**
Some "General Reflection Model" data of input block 6A are copied identically NLIMI + NSTSI times onto appropriately dimensioned arrays (with the same names) in order to make them dependent upon the surface labelling index as well.
The array index is then the surface number.
Presently this "localization option" is available for the flags

RINTEG, EINTEG, AINTEG

Overwriting general reflection data for some specific surfaces can be done via a call to the entry RF0USR of the user supplied reflection routine REFUSR. RF0USR is called in the initialization phase of an EIRENE run (see section 3.3).
By a similar strategy, some of the species independent flags in the next sub-block 6B can be made species-dependent. See, for example, RECYCF, RECYCT, RECYCS and RECYCC below.

```
*** 6B. Data for Local Reflection and Sputtering Models}
```

The next 3 (or 4) lines comprise the sub-block for local reflection data. Such sub-blocks can be included in each "surface deck" (block 3A, 3B) to overwrite the default reflection model for each individual surface.

```
      ILREF  ILSPT  ISRS  ISRC
      ZNML EWALL EWBIN TRANSP(1) TRANSP(2) FSHEAT
      RECYCF  RECYCT  RECPRM  EXPPL  EXPEL  EXPIL
C (following line may be omitted then: default sputter model,
C  see below)
      RECYCS  RECYCC  SPTPRM  ESPUTS ESPUTC
```

An arbitrary number of such sub-blocks of 3 (or 4) lines each may also be defined under a certain label SURFMOD_modname and be included in the input file in block 6 directly after the ERMIN, ... deck
In blocks 3a and 3b surfaces may be assigned a particular local reflection model by a card reading SURFMOD_modname. Many surfaces may then be linked to the same surface reflection sub-block.
Example:

```
SURFMOD_BERYL_SPT_300K
      1       2       0       0
 9.04000E+02 -2.60000E-02 0.00000E+00 0.00000E+00 0.00000E+00 2.80000E+00
 1.00000E+00 1.00000E+00 1.00000E+00 1.00000E+00 5.00000E-01 1.00000E+00
 1.00000E+00 1.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00
SURFMOD_CARB_SPT_1153K
      1      12       0       0
 1.20600E+03 -1.00000E-01 0.00000E+00 0.00000E+00 0.00000E+00 2.80000E+00
 1.00000E+00 1.00000E+00 1.00000E+00 1.00000E+00 5.00000E-01 1.00000E+00
 1.00000E+00 1.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00
TRANSP1   D2        0.5000E 00
TRANSP2   T2        0.2500E 00
RECYCF    D         0.0000E 00
RECYCT    D2        0.9500E 00
SURFMOD_CARB_SPT_812K
      1       2       0       0
 1.20600E+03 -7.00000E-02 0.00000E+00 0.00000E+00 0.00000E+00 2.80000E+00
 1.00000E+00 1.00000E+00 1.00000E+00 1.00000E+00 5.00000E-01 1.00000E+00
 1.00000E+00 1.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00
```

**Meaning of the Input Variables**

**ILREF** Flag for choice of local reflection model

= 1 TRIM database reflection model is used. NLTRIM must be .TRUE.

= 2 "modified Behrisch Matrix model" is used

= 3 user supplied reflection model (see section 3.3: Subroutine REFUSR)

Default: ILREF = 2

**ILSPT** Flag for choice of local sputtering model

Let ILSPT = MN, with M and N single digit integers each. Then N controls the options for physical sputtering, and M controls chemical sputtering. See subroutine SPUTER.

**N = 0** no physical sputtering at this surface

**N = 1** constant physical sputtering rate (see parameter RECYCS below)

**N = 2** modified Roth–Bogdansky formula for sputter yield, Thompson energy distribution and cosine angular distribution for emitted particles (see references [**kn:Sputer95**] and [**kn:Roth96**]).

**N = 9** (was option N=3 in Eirene-2004 and older)

user supplied sputtering model (see section 3.3: Entry SPTUSR to subroutine REFUSR)

**M = 0** no chemical sputtering at this surface

**M = 1** constant chemical sputtering rate (see parameter RECYCC below)

**M = 2** "Roth formula" for chemical sputter yield, thermal distribution for emitted particles (see reference [**kn:Roth99**]), "weak flux dependence option A6".

**M = 3** "Roth formula" for chemical sputter yield, thermal distribution for emitted particles (see reference [**kn:Roth99**]), "strong flux dependence option A7".

**M = 4** "Roth formula" for chemical sputter yield, thermal distribution for emitted particles (see reference [**kn:Roth99**]), "new flux dependence option A8 (2004)".

**M = 5** not in use

**M = 6** "Haasz–Davis 1998 formula" for chemical sputter yield

**M = 7** "Haasz–Davis 1998 formula" for chemical sputter yield, and multiplicative factor for flux dependence (Roth, 2004).

**M = 9** (was option N=3 in Eirene-2004 and older)

user supplied sputtering model (see section 3.3: Entry SPTUSR to subroutine REFUSR)

Default: ILSPT=0

The next two surface-species index flags ISRS$ and ISRC$ control the species of sputtered particles. Hence they are considered surface properties and are read in the surface specification decks in blocks 3a, 3b and/or 6 (sections 2.3.1, 2.3.2 and 2.6).

**ISRS$** sputtered particle species flag (physical sputtering).

$> 0$ both the sputtered particle and the reflected particle (if any) will be followed. Their contribution to surface particle and energy fluxes is stored in surface averaged tallies 1 to 24, i.e., sputtered particles are not explicitly distinguished from reflected particles in the particle and energy balances. Furthermore the "sputtered flux surface tallies" 33 to 37 (section 6.1.2) in older versions before 2002, and on tallies 51 to 81 (section 6.1.1) else, are also updated. The species index of the sputtered particle (atom) is IATM=ISRS. Hence, on input, one must have $1 \leq$ ISRS $\leq$ NATMI, otherwise: error exit.

$\leq 0$ Same as ISRS $> 0$, however, the species index of the sputtered particle is determined automatically from comparing the charge and mass numbers of the available atomic species (input block 4a) with the corresponding surface material (nuclear mass and charge) data of the surface element. If no suitable atomic test particle is is found, then only sputter tallies are scored, but no sputtered particles are then subsequently traced.

$= 0$ Same as ISRS $\leq 0$, but in this case a sputtered particle is NOT followed, even if its atomic test particle species could be identified.

Only the reflected particles are followed. and only their contribution to surface particle and energy fluxes is stored in regular surface averaged tallies 1 to 24. Sputter tallies are still scored.

Note: ISRS=ISRS(ISPZ,MSURF), so the above described options for sputtered particle species, as well as for either only scoring fluxes or even tracing these sputtered particles, can be made dependent on the incident species index ISPZ.

**ISRC$** sputtered particle species flag (chemical sputtering).

$> 0$ both the sputtered particle and the reflected particle (if any) will be followed. Their contribution to surface particle and energy fluxes is stored in surface averaged tallies 1 to 24, i.e., sputtered particles are not explicitly distinguished from reflected particles in the balances. Furthermore the "sputtered flux surface tallies" 25 to 28 are updated. The species index of the sputtered particle (atom) is IATM=ISRC, if ISRC $\leq$ NATMI, or (molecules) IMOL, if ISRC = NATMI+IMOL and NATMI $<$ ISRC $\leq$ NATMI+NMOLI. Hence, on input, ISRC $\leq$ NATMI+NMOLI.

$= 0$ There is no chemical sputtering for the particular surface element and incident species (note: ISRC=ISRC(ISPZ,MSURF), i.e., $p_c = 0$ here.

Only the reflected particles are followed. Their contribution to surface particle and energy fluxes is stored in surface averaged tallies 1 to 24.

$< 0$ Same as ISRC $> 0$, however, the species index of the sputtered particle is determined automatically from comparing the charge and mass numbers of the atomic species (input block 4a) with the corresponding data of the surface element. I.e., in case of Carbon surfaces the sputtered particle is a C-atom, if such an atom has been specified in input block 4a

**ZNML** = KLMN (4 digits)

**KL** atomic weight of wall material. Note: the nearest integer of the mass number in the TRIM runs is used. For example, a copper target is specified in the TRIM files with an atomic weight of 63.54, and the corresponding TRIM file is used for surfaces with KL=64.

**MN** nuclear charge number of wall material

Example: Carbon: ZNML=1206.

Example: Molybdenum: ZNML=9642.

Example: Copper: ZNML=6429.

Default: ZNML = 5.626E3 (stands for Fe).

**EWALL**

$< 0$ -EWALL = TW is a (surface-) temperature (eV) in a Maxwellian flux distribution for the thermal particle energy. The resulting mean energy is $E_{mean} = 2 \cdot TW$.

$> 0$ +EWALL = Energy of mono-energetic (thermal) particles.

The relation between surface temperature TW and the mean energy of particles then reads EWALL = $E_{mean} = 1.5 \cdot TW$.

$= 0$ Energy is sampled from a Thompson distribution, using the flag EWBIN (see below) as parameter for the surface binding energy

Default: EWALL = +0.0388 ($\simeq$ TW = 0.026 eV $\simeq$ 300 K)

Note that the $EWALL > 0$ option enables EIRENE to include boundary conditions in "one speed transport equation" approximations, which often are of great interest in general linear transport theory. The $EWALL < 0$ option, together with the EINTEG and AINTEG flags described above, provides the Maxwell boundary conditions with given wall accommodation coefficient (see above).

**EWBIN** see above, EWALL = 0 option.

Default: EWBIN = 0.0 (irrelevant for "Default Model")

**TRANSP(1)** Semi-transparency for particles incident from the positive side. Renders a non-transparent surface (ILIIN > 0) semi-transparent. The probability of a test flight to pass through the surface is [TRANSP]. This transparent fraction of the current is not scored on surface tallies, i.e. for this fraction of surface currents the ILIIN=0 option is used, see input block 3B. Hence: the probability for reflection/re-emission/absorption according to ILLIN setting is [1-TRANSP]. Only this fractions [1-TRANSP] of the surface currents are scored as "incident flux" and, if applicable, "re-emitted flux" surface tallies. In order to also score the net current of the transparent flux fraction [TRANSP] on surface tallies, an additional "very nearby" fully transparent scoring-surface can be used.

Default: 0.0 (i.e., fully reflecting surface).

Irrelevant for transparent surfaces (with ILIIN $\leq$ 0).

**TRANSP(2)** Semi-transparency for particles incident from the negative side on a non-transparent surface. Otherwise same as for TRANSP(1) option.

Default: 0.0 (i.e., fully reflecting surface).

Irrelevant for transparent surfaces (with ILIIN $\leq$ 0).

**FSHEAT** surface sheath potential factor. The sheath potential is $FSHEAT \cdot T_e$, with $T_e$ the electron temperature at the point of incidence. This sheath potential is applied if ions (test ions or bulk ions) hit a non-transparent surface.

If FSHEAT $\leq$ 0.0, then a sheath potential computed from the local background plasma flow conditions is used (function SHEATH), assuming ambipolar flow, a Boltzmann distribution for electrons and zero secondary electron emission. See section 1.5. In case of zero (undefined) background plasma flow velocity at the place of incidence, a default of FSHEAT = 2.8 is used (corresponding to $T_e = T_i$, and a single ion species $D^+$ plasma flowing at ion acoustic speed parallel to the $\vec{B}$-field into the sheath.

Default: FSHEAT = 0.0

**RECYCF, RECYCT** Multiplier for reflection probability RPROBF:

Particles can be re-emitted from surfaces by either the "fast reflection" model or by a "thermal emission" model. Flag RECYCF controls (scales) the "fast particle reflection" probability $p_f$.

The probability $p_f = RPROBF(E_{in}, \Theta_{in}, ispez, wall)$ for the "fast" particle reflection model, as specified by other flags for this surface, is modified to

$$RPROBF(E_{in}, \Theta_{in}, ispez, wall) = AMIN(RECYCF \cdot RPROBF, RECYCT), \quad (2.6)$$

where RPROBF was evaluated from the reflection model specified by ILREF. Note the cut-off at recycling coefficient RECYCT (defined below).

The total recycling coefficient RECYCT = $p_f + p_t$ is unchanged by flag RECYCF. Hence, by the use of RECYCF not only the fast particle reflection probability, but also the probability

for thermal particle emission $p_t$ is altered to maintain a total recycling coefficient at this surface of RECYCT.

Default: RECYCF = 1 for incident atoms, test ions and bulk ions.

Default: RECYCF = RPROBF =0 for incident molecules.

**RECYCT** Recycling coefficient (must not be negative):

A flux $RECYCT \cdot Influx$ is re-emitted from a surface, for any $Influx$ of particles of any species, where all fluxes are measured as "atomic fluxes" (=fluxes of nuclei). RECYCT hence defines the sticking probability $p_a$ [and hence also the pumping speed (2.7)] of any surface in EIRENE, for all incident species.

The fraction $p_a$ = (1 - RECYCT) of incident (atomic) flux will be absorbed at the surface. The non-sticking, i.e. the re-emitted fraction $RECYCT = 1 - p_a$ is split into a "fast" and a "thermal" component.

A fraction $p_f$ = RPROBF [see (2.6)] of the incident particles is reflected as described by the "fast particle reflection model". However, by relation (2.6) it is ensured that RPROBF is always less than or equal to RECYCT.

The fraction $p_t$ = RPROBT = (RECYCT - RPROBF) will be re-emitted by the "thermal particle reflection model". This flag is to be used to define an effective pumping speed at certain surfaces, see section 2.6.1 below.

Note: RPROBF = 0 for incident molecules (ITYP=2) by default.

Default: RECYCT = 1., i.e. $p_a = 0$

**RECPRM** free model parameter for user supplied recycling models $ILREF = 9$.

Default: RECPRM = 0.

**EXPPL** (only for ILREF = 2 option)

incident angular dependence of fast particle reflection coefficient. The formula

$$R(\Phi) = 1 - (1 - RPROBF) \cdot \cos^{EXPPL}(\Phi)$$

is used, $e1 = EXPPL$, see (1.59), where:

**RPROBF** Reflection probability from "Behrisch Matrix" model, which is valid only for normal incidence.

| | |
|---|---|
| $\Phi$ | Angle of incidence against surface normal |
| R($\Phi$) | Reflection probability for particles incident with angle $\Phi$ |

note: $R(\Phi) = 1$ for $\Phi = 90°$ and EXPPL $> 0$.

Default: EXPPL = 1. (recommended from a comparison with the TRIM database)

**EXPEL** (only for ILREF = 2 option)

as EXPPL, incident angular dependence for the energy reflection coefficient.

Default: EXPEL = 0.5 (recommended from a comparison with the TRIM database). e2 = EXPEL, see (1.60) and (1.61)

**EXPIL** (only for ILREF = 2 option)

incident angular dependence for angular distribution of re-emitted atom or molecule;

= 0 cosine distribution (Lambertian)

> 0 mixed cosine-specular model.

the specular contribution increases according to (1.62) with angle of incidence $\Phi$ and with e3 = EXPIL (recommended: $EXPIL \leq 1$).

> 100 specular reflection angles

Default: EXPIL = 0.

**RECYCS** The meaning of this flag for the "physical sputtering" options depends upon the value of the first digit N of ILSPT:

**N = 0** no physical sputtering, YIELD1 = 0. RECYCS is irrelevant.

**N = 1** constant physical sputtering yield, YIELD1 = RECYCS

**N = 2** RECYCS is a multiplier for the sputtered particle flux YIELD1. YIELD1 is computed from the incident species, energy, angle and surface parameters by the sputter model $N = 2$. Hence: the sputtered particle yield YIELD1 as computed from subroutine SPUTER is modified to

$YIELD1 = RECYCS \cdot YIELD1.$

**N = 9** RECYCS is a free model parameter, which can be used in the user supplied sputter model for any particular surface element.

Default: RECYCS = 1.

**RECYCC** The meaning of this flag for the "chemical sputtering" options depends upon the value of the second digit M of ILSPT:

**M = 0** no chemical sputtering, YIELD2 = 0. RECYCC is irrelevant.

**M = 1** constant chemical sputtering yield, YIELD2 = RECYCC.

**M = 2** RECYCC is a multiplier for the sputtered particle flux YIELD2. YIELD2 is computed from the incident species, energy, angle and surface parameters by the sputter model $M = 2$. Hence: the sputtered particle flux YIELD2 as computed from subroutine SPUTER is modified to

$YIELD2 = RECYCC \cdot YIELD2.$

**M = 9** RECYCC is a free model parameter, which can be used in the user supplied sputter model for any particular surface element.

Default: RECYCC = 1.

**SPTPRM** free model parameter for user supplied sputtering models N=3, M=3.

Default: SPTPRM = 0.

**ESPUTS** (new: March 2015) parameter (flag) for energy of physically sputtered particle. Currently not in use.

Default: ESPUTS = 0.

**ESPUTC** (new: March 2015) parameter (flag) for energy of chemically sputtered particle. By default: chemically sputtered particles are released from the wall by the "thermal surface emission model", as also used for the recycling/reflection thermal emission, i.e. with wall temperature (as specified by EWALL parameter, see above) and either a cosine (for monoenergetic emission at EWALL = 1.5 TWALL) angular distribution, or sampling from a stationary Maxwellian flux distribution at -EWALL = TWALL.

If ESPUTC .GT. 0, then the the chemically sputtered particles are released with a monoenergetic distribution at E0 = ESPUTC, and a cosine angular distribution.

Default: ESPUTC = 0.

**Note:**
All data in a block for "local reflection data" are, in general, independent of the type and species of the incident particle. Some are, however, copied identically NPHOTI + NATMI + NMOLI + NIONI + NPLSI times onto appropriately dimensioned arrays (with the same names). This is currently done for the parameters:

```
ISRS    ISRC
TRANSP(1,  ...)   TRANSP(2,  ...)
RECYCF  RECYCT  RECPRM  EXPPL  EXPEL  EXPIL
RECYCS  RECYCC  SPTPRM  ESPUTS ESPUTC
```

The first index is the species index, the second index is the surface number.
Overwriting local reflection data for some specific incident particle species can be done via a call to the entry RF0USR of the user supplied reflection routine REFUSR. This entry is called in the initialization phase of an EIRENE run, from subroutine REFLEC. Likewise, the entry SP0USR of the user supplied sputter routine SPTUSR is called in the initialization phase from subroutine SPUTER. For further details on user supplied surface interaction routines see section 3.3.
For example: RECYCT(3,5) is the recycling coefficient for incident species 3 onto surface no. 5.
If surface models are defined in input block 6 by the SURFMOD_modname label, (rather than individually for each surface in blocks 3a or 3b), then these variables may be made species dependent by adding an arbitrary number of lines to a SURFMOD-deck, each of which overwriting the specification for a particular particle species. The name of the species (blocks 4 and 5) must be uniquely determining one of the test particle or bulk species.
In the example of SURFMOD decks given above, e.g., the first such additional line overwrites the original value of TRANSP(1, ...) (=0.0) for species D2 with the value 0.5. I.e., all surfaces, to which the reflection model modname=*CARB_SPT_1153*K is assigned, have a recycling coefficient =1.0 for all incident species, except for the D2 molecules, for which these surfaces are made transparent with probability 0.5, if these molecules are incident from the positive side. For T2 molecules incident from the negative side this transparency is set to 0.25. The other species dependent modifications in this SURFMOD deck are self-explaining.

### 2.6.1 effective pumping speed

The flag RECYCT described in the previous section is also used to specify surface pumping in the following way:
Let $A$ be the surface area [cm$^2$] of a surface (additional surface or non-default standard surface) as seen by test particles, to which a given pumping speed $S$ is to be assigned. Then the pumping

speed $S$ [L s$^{-1}$] for particles with temperature $T$ [K] and mass $m$ [AMU] is related to the sticking fraction $p_a = 1 - RECYCT$ by

$$S = A \cdot (1 - RECYCT) \cdot 3.638 \cdot \sqrt{T/m} = \tilde{A} \cdot 3.638 \cdot \sqrt{T/m} \tag{2.7}$$

If the surface area is given in [m$^2$], $S$ in [m$^3$ s$^{-1}$], then the numerical factor becomes 36.38.
Note: the first two factors define an *effective exposed area* $\tilde{A}$, across which particles with a thermal velocity would be removed. The remaining factors are the (thermal) effective velocity of particles lost across this area: $S = \tilde{A} \cdot v_{th}$ [volume /time]
The pump-throughput (pumped flux), (mass flow rate) $Q$ is the product of pressure $P$ in front of surface $A$ and pumping speed $S$:
$Q_{pump} = P \cdot S$,
and, for a given type of gas ($m$) and gas temperature $T$ the pump-throughput is given e.g. in flux units s$^{-1}$, (or: amp), or as mass flow with dimension pressure times volume per time.
The temperature $T$ and type of gas (given by $m$) are part of the specification of $S$, e.g. to be found from the technical specification of the pump. $S$ may also depend on the pressure for some pumps: $S = S(P, T, m)$. However in the molecular flow regime $S$ may often assumed to be a constant (independent of pressure) for a given temperature and type of gas.
But normally even in this case the specified pumping speeds $S$ for vacuum pumps do not follow the $\sqrt{T/m}$ dependence in equation (2.7) for different types of gas. E.g. pumping speeds for $H_2$ vs. He of "real pumps", for same temperatures, are usually not simply related by a factor $1/\sqrt{2}$, and those for $D_2$ and He may not be identical either. In a simulation of a gas mixture then a species dependent flag RECYCT (to produce a species dependent effective area $\tilde{A}$(ISPZ) has to be specified to reproduce a given ratio of pumping efficiencies for different types of gas, for a particular type of pump (see the "Note" at the end of previous section: RECYCT(ISPZ) option) to reproduce the proper species dependence (and pressure dependence, if any) of the pumping system.
Probably sometimes, when $S = S(P)$, even RECYCT has to be made dependent on the pressure $P$ found in front of the surface $A$ (then e.g. by an iterative procedure)

### 2.6.2 Pressure Feedback Loop

This type of boundary condition modifies the recycling probability RECYCT by means of a proportional-integral loop so that the pressure in a given cell matches a given reference value. It is activated by setting ILREF = 4 in the desired reflection model in block 6a of the Eirene input file. Two new values, related with the reference cell and pressure, have to be provided. The first one in the line:

```
ILREF ILSPT ISRS ISRC REFCELL
```

in which REFCELL is an integer referencing the cell in the Eirene mesh in which the pressure will be computed. This value can be easily obtained with triang by means of the (I)nquire function. The second value is provided in the line:

```
RECYCS RECYCC SPTPRM ESPUTS ESPUTC REFPRESS
```

in which REFPRESS is the reference pressure in $Pa$.
This loop will not provide values of RECYCT larger than 1 or below 0. The coefficients of the proportional-integral loop are hard-coded.

## 2.7 Input data for Initial Distribution of Test Particles

**General Remarks**

The primary source (and: initial distribution of test particles, in time dependent mode) is given as a function $Q(i, \mathbf{r}, t, \mathbf{v})$. $Q$ is the density of the probability distribution from which the species index $isp$, the starting point $\mathbf{r}$, the velocity vector $\mathbf{v}$ and the starting time $t$ are sampled in subroutine LOCATE. $(isp, \mathbf{r}, t, \mathbf{v})$ is the state of the starting particle, which then will be "followed" (traced) in subroutine FOLNEUT or FOLION. There are five primary types of spatiotemporal distributions for primary sources ("Strata"), namely Point sources, Line sources (to be written), Surface sources, Volume sources and "Census sources". The first four are uniform in a time-interval (or time-independent), and the fifth one is an initial distribution (in volume) at a given point in time. As the transport equation (within each iterative step, if any) is linear, various sources of test particles can be treated subsequently and the responses can then be linearly superposed. EIRENE can handle up to NSTRA (see: PARMUSR, section 3.1) different strata and it prints output from each single stratum as well as from the sum over strata. The source strength is prescribed for each stratum separately (input card: FLUX(ISTRA), ...). Subdividing the total source into such strata can be useful to increase the efficiency of the code, ("stratified source sampling", see section 1.3.3.2 or any textbook on Monte Carlo integration), or if the contribution of such sub-sources is of interest by itself.
In order to facilitate sampling each stratum can be subdivided further into a number (NSRFSI) of "sub-strata". Random sampling is done by firstly sampling the sub-stratum ISRFSI, and then (conditional) the initial state of the test flight within this sub-stratum.
Random numbers from the multivariate distribution $Q$ are generated by a sequence of uni-variate conditional distributions (see (1.33)). Source sampling from $Q$ in EIRENE is based on the (formal) decomposition

$$\begin{aligned} Q(\mathbf{r}, t, isp, \mathbf{v}) &= Q_1(\mathbf{r}, t) \times Q_2(isp|\mathbf{r}, t) \times Q_3(\mathbf{v}|isp, \mathbf{r}, t) \\ &= \sum_i \int d\mathbf{v}\, Q(\mathbf{r}, t, i, \mathbf{v}) \times \int d\mathbf{v}\, Q(isp, \mathbf{v}|\mathbf{r}, t) \times Q(\mathbf{v}|isp, \mathbf{r}, t) \end{aligned} \tag{2.8}$$

i.e. the source sampling routines in EIRENE (samvol.f, samsrf.f, etc.) start by sampling the spatio-temporal birth point coordinates $\mathbf{r}_0, t_0$ from $Q_1(\mathbf{r}, t)$ (obtained by integrating $Q$ over velocity space $\mathbf{v}$ and summing over all species), then next sampling the species index $i$ from the conditional distribution $Q_2(i)$, which is conditional on $\mathbf{r}_0, t_0$ and integrated over velocity space, and finally then sampling the velocity $\mathbf{v}_0$ from $Q_3(\mathbf{v})$ conditional on species $i_0$ and space-time $(\mathbf{r}_0, t_0)$.
To facilitate random sampling from $Q_k$ for a particular stratum $k$, each stratum can be further subdivided into a sum of "substrata":

$$Q_k = \sum_j w_{k,j} Q_{k,j} \quad \text{with normalisation} \quad \sum_j w_{k,j} = 1 \tag{2.9}$$

There are NSRFSI substrata, and the weights $w_{k,j}$, $j = 1, \ldots, NSRFSI$, are used for first determining a particular substratum $j_0$ by sampling, and then sampling from distribution $Q_{k,j_0}(\mathbf{r}, \mathbf{v}, i, t)$ of stratum $k$ as described above.
The total CPU resources (and storage for the census array, see section 2.13) are distributed over the strata. This distribution is controlled by some of the input flags in this input block, as

described below. EIRENE stops working on a particular stratum if either the assigned CPU-time for this stratum has been reached (Message: "No further CPU time for this stratum"), or if all requested histories have been calculated (Message: "All histories for this stratum completed") or, in case of time dependent problems, if the storage on the census array of this stratum has been filled up (Message: "Census array filled for this stratum").

**The Input Block**

```
*** 7. Data for primary sources
      NSTRAI
      INDSRC(ISTRA), ISTRA=1,NSTRAI
      ALLOC  AMPTS
      DO 70, ISTRAI=1,NSTRAI
        TXTSOU
        NLAVRP  NLAVRT  NLSYMP  NLSYMT
        NPTS  NINITL  NEMODS  NAMODS  NMINPTS
        FLUX  SCALV   IVLSF  ISCLS  ISCLT
     .    ISCL1  ISCL2  ISCL3  ISCLB  ISCLA
*Species index distribution
        NLATM  NLMOL  NLION  NLPLS NLPHOT
        NSPEZ
*Distribution in physical space and time
        NLPNT  NLLNE  NLSRF  NLVOL  NLCNS
        NSRFSI
        DO 75, J=1,NSRFSI
          INUM     INDIM    INSOR    INGRDA(1)  INGRDE(1)
                                     INGRDA(2)  INGRDE(2)
                                     INGRDA(3)  INGRDE(3)
          SORWGT  SORLIM  SORIND  SOREXP  SORIFL
          NRSOR   NPSOR   NTSOR   NBSOR   NASOR   NISOR
          SORAD1  SORAD2  SORAD3  SORAD4  SORAD5  SORAD6
   75   CONTINUE
*Distribution in velocity space
       SORENI  SORENE  SORVDX  SORVDY  SORVDZ
       SORCOS  SORMAX  SORCTX  SORCTY  SORCTZ
   70 CONTINUE
```

**Meaning of the Input Variables for primary sources**

**NSTRAI** Number of different sources ("Strata"), which are computed one after the other and are linearly superimposed at the end of the run.

(NSTRAI ≤ NSTRA, see "Parameter-Statements")

**INDSRC**

= 0–5 the input data for stratum ISTRA are read here, but may be modified in some user routine (SAMUSR) or interface routine (INFCOP, at entry IF2COP(ISTRA))

= 6 no input data for stratum ISTRA are read here. The definition of this stratum must be entirely in some problem specific routine (IF2COP, etc.). See section 3.4 for one such example, namely the default surface recycling source model as specified in coupled B2-EIRENE runs.

= −1 the input data for stratum ISTRA are read here, and no attempt is made to modify these. I.e. IF2COP(ISTRA) is not called.

**ALLOC** Allocation of CPU-time to stratum weighted as

(1-ALLOC)*NPTS+ALLOC*FLUX

**AMPTS** (available since 2014 in master version) Multiplier, used simultaneously for permitted total CPU time NTCPU (input block 1), and for specified number of MC histories NPTS(ISTRA) (see below). Default: AMPTS=1.0

**TXTSOU** Text to characterize the stratum (name of the source) on the printout file.

**NLAVRP** = .TRUE. not in use

**NLAVRT** = .TRUE. not in use

**NLSYMP** = .TRUE.

Symmetrize profiles with respect to poloidal (y-) co-ordinate $x^2$, i.e., with respect to the poloidal surface $x^2$ = PSURF((NP2ND+1)/2) in case NP2ND is an odd integer, or with respect to the cell center $x^2$ = PZONE(NP2ND/2) in case NP2ND is an even integer.

**NLSYMT** = .TRUE.

ditto, but for toroidal (z-) co-ordinate, i.e., for toroidal surface

TSURF((NT3RD+1)/2) or TZONE(NT3RD/2) respectively.

**NPTS**

> 0 Maximum number of test particle histories.

If there is more than one stratum (NSTRAI > 1) then the total CPU-time NTIME (input block 1) will be distributed proportional to NPTS to the single strata. NPTS is the maximum number of test-particles only if sufficient CPU time is available. Otherwise a message "NO FURTHER COMPUTATION TIME FOR THIS STRATUM" is printed and the particle loop for the respective stratum is stopped.

= 0 this stratum is "turned off".

< 0 no limitation in the number of particles. The entire CPU time assigned to this stratum will be used up. (NPTS is reset to the largest integer on the machine. Hence: some care is needed here in case of multiple strata, in combination with the ALLOC-options to assign CPU time to individual strata).

**NINITL**

> 0 seed for initialization of random number generator. The results for all those individual strata can be reproduced exactly for which the same number of test-flights is computed as in a previous run.

= 0 no initialization of random numbers for the particular stratum. In case of the first stratum, the default initialization is used. Runs can only be reproduced, if the same number of test-flights is computed for each stratum. Somewhat weakened correlation between subsequent runs as compared to the NINITL > 0 option.

< 0 truly random initialization (determined by machine clock). These runs cannot be reproduced exactly. Subsequent runs are uncorrelated. (E.g.: recommended for stochastic approximation procedures in nonlinear applications.

**NEMODS** Flag to select one of the preprogrammed source energy conditional distributions given the source particle's position and species. (See: "distribution in velocity space", below)

**NAMODS** Flag to select one of the preprogrammed conditional source angular distributions given the position, species and energy of the source particle. (See: "distribution in velocity space", below)

**NMINPTS** Minimum number of test particles enforced for this stratum, independent of NTCPU flag in input block 1. Hence: setting this flag may increase EIRENE run time above the CPU-time assigned to a run in the first input line in input block 1. Default: NMINPTS = 0

**FLUX, SCALV**

**SCALV=0** (default) FLUX = Source strength in Ampere.

FLUX is the scaling factor for all surface- or volume averaged tallies.

FLUX is an "atomic flux" (or: an "atomic ion flux"). Each source particle may carry a different flux NPRT(ISPZ) (initial weight) depending on the species ISPZ (see below: distribution for the species index). NPRT is specified in the blocks 4 and 5. The total "atomic" source particle flux for each stratum is scaled to be FLUX. For example, a $H_2$ molecule source, with $\mathrm{NPRT}_{H_2}$ = 2, is treated as if a flux of FLUX/1.602E-19/2 $H_2$-molecules per second is emitted, resulting in an equivalent "atomic flux" FLUX/1.602E-19 per second.

**SCALV≠0** The default scaling of tallies with FLUX can be overruled by this flag. The common scaling factor for all surface- and volume averaged tallies is determined such that one particular tally (selected by the ISCL. . . -flags described below) has the prescribed value SCALV. This determines the scaling of all other volume averaged and surface averaged tallies. By this option, for example, one can set the neutral particle

density to a prescribed value in one particular cell. Hence, one can prescribe the local Knudsen number for nonlinear applications including neutral–neutral interactions.

**IVLSF**

= 1 The following ISCL. . . -flags select one particular volume averaged tally

= 2 The following ISCL. . . -flags select one particular surface averaged tally

**ISCLS** species index of selected tally

**ISCLT** tally number of selected tally (refer to tables 6.3 and 6.4)

**ISCL1, ISCL2, ISCL3, ISCLB, ISCLA**

**IVLSF=1**

cell numbers NRCELL, NPCELL, NTCELL, NBLOCK, NACELL, respectively.

if (NPCELL = 0) or (NTCELL = 0), then ISCL1 = NCELL, the cell number in the 1-dimensional arrays (see end of section 2.2.1).

The additional cell region is specified by NRCELL=0, NPCELL=1, NTCELL=1, NBLOCK=NBMLT+1 (see section 2.2) and the proper value of NACELL.

**IVLSF=2** to be written

**Distribution for the species index ISP**

**NLATM** = .TRUE.

Atomic source. History starts in subroutine FOLNEUT with type index ITYP=1, species index ISP = IATM and initial weight NPRTA(IATM) (see block 4A)

**NLMOL** = .TRUE.

Molecule source. History starts in subroutine FOLNEUT with type index ITYP=2, species index ISP = IMOL and initial weight NPRTM(IMOL) (see block 4B)

**NLION** = .TRUE.

Test ion source. History starts in subroutine FOLION with type index ITYP=3, species index ISP = IION and initial weight NPRTI(IION) (see block 4C)

**NLPHOT** = .TRUE.

Photon source. History starts in subroutine FOLNEUT with type index ITYP=0, species index ISP = IPHOT and initial weight NPRTPH(IPHOT) (see block 4D)

Not all options for direct photon sources are fully programmed. Currently we mostly use the bulk particle volume recombination source (see next) to simulated radiative decay from excited states as birth profile for (bound-bound) line-photons.

**NLPLS** = .TRUE.

Bulk ion source. Initial co-ordinates of a bulk ion with species index ISP = IPLS and initial weight NPRTP(IPLS) (see block 5) are generated as primary source particles, then a surface reflection model or a volume re-combination model is called and atoms, molecules or test ions with species index either IATM, IMOL, IION or IPHOT are created.

One and only one of these five variables must be .TRUE..

**NSPEZ** Species index of the source particle

$\geq 1, \leq$ **NATMI, NMOLI, NIONI, NPLSI** NSPEZ is the (fixed) species index of the source particle. No random sampling for the species index is done.

$>$ **NATMI, NMOLI, NIONI, NPLSI** (depending upon the type of the source particle) the species index is sampled from the distribution DATM, DMOL, DION, DPLS, respectively. The "surface species distributions" DATM, DMOL, DION and DPLS are read in the block "Data for General Reflection Models", see section 2.6.

$= 0$ the species index of the particle is directly sampled from the "analog distribution" WEISPZ, i.e., no biased source species sampling. WEISPZ is defined internally by the code.

The distribution WEISPZ is currently defined only for NLPLS=TRUE sources, and here only for surface recycling sources using the STEP-function option (in SAMSRF, see further below this section and section 2.7.1). There it is set according to the local bulk ion flux composition.

WEISPZ may also be transferred into a run via user specified source sampling (SAMUSR.f, see section 3.4).

In all other cases NSPEZ must be positive.

$< 0$ The "surface species distributions" DATM, DMOL, DION and DPLS are considered as biased source species distributions, whereas the analog (physical) distribution is provided automatically by the array WEISPZ from the source sampling routines SAMPNT, SAMLNE, SAMSRF or SAMVOL respectively. An appropriate weight correction is carried out after sampling from DATM, DMOL, DION or DPLS, respectively, in subroutine LOCATE.

Note: If a step function (function STEP, see below) is used for sampling the start position of a test particle on a surface, then the species index NSPEZ automatically also fixes the choice of the index ISPZ for the spatial step function STEP(ISTEP, ISPZ, ...) selected by the flags SORLIM and SORIND (=ISTEP) (see below). This default can be overruled when SORIND has three digits.

### Distribution in physical space

In this section options are described to sample the starting point $\mathbf{r}_0$ of a new test particle. Together with the starting position also a vector $\mathbf{C}(\mathbf{r}_0) = \mathbf{C} = (CRTX, CRTY, CRTZ)$ is generated, which may be used to distinguish one particular direction for the distribution in velocity space (see paragraph below: distribution in velocity space). Internally this reference vector $\mathbf{C}$ is

always normalized to unit length 1. **C** is interpreted as an outward pointing vector (at the source point). Outflows (e.g. plasma fluxes onto a surface segment) have a positive velocity component in direction of this vector. Emitted particles (recycling, reflection, emission from a point source) have a negative velocity component in this reference direction **C**.
**C** is irrelevant for isotropic velocity distribution.
It is stressed again that the reference unit vector $\mathbf{C} = (C_X, C_Y, C_Z)$ controlling un-isotropic angular distributions is always taken as an *outward pointing (surface-normal) vector.* Outflowing background plasma particles (species option NLPLS = TRUE, see above) onto a surface element first have a positive velocity component in direction of **C**: $\mathbf{V}_b\dot{\mathbf{C}} > 0$. Recycled test particles then are emitted/reflected with a negative velocity component, i.e $\mathbf{V}_t\dot{\mathbf{C}} < 0$. Even in case of other sources, when no surface normal can be assigned naturally (point sources (gas puff), volume sources) this reference vector **C** points *away* from the direction of preferential test particle emission.

**NLPNT** = .TRUE. Point Source

**NLLNE** = .TRUE. Line Source (not ready)

**NLSRF** = .TRUE. Surface Source

**NLVOL** = .TRUE. Volume Source

**NLCNS** = .TRUE. Initial conditions source (sampling from census array), for time-dependent mode of operation, see input blocks 1. and 13.

One and only one of these 5 variables must be .TRUE.

**Point source** Flags for the "distribution in physical space" not mentioned here are irrelevant for point sources.

**NSRFSI (=NPNTSI)** Total number of different points, over which the starting points for this stratum are distributed (corresponds to "sub-strata" option for surface and volume sources, there to facilitate sampling of spatial coordinates).

The next deck of 4 input cards is read NSRFSI times, one deck for each "sub-stratum".

**INUM** irrelevant; labelling index for sub-strata

**SORWGT** Relative frequency for starting point labelled INUM. The sum of SORWGT for all NPNTSI points is normalized to one internally.

**NRSOR**

> 0 x- or radial cell number NRCELL of the zone containing the point source.

= 0 NRCELL is found automatically from the "standard mesh" zoning.

< 0 only for surface sources (NLSRF), see below.

**NPSOR** ditto, for y- or poloidal cell number NPCELL

**NTSOR** ditto, for z- or toroidal cell number NTCELL

**NBSOR** standard mesh block number NBLOCK. Defaulted to NBLOCK = 1, if NBSOR $\leq 0$

**NASOR** additional cell number NACELL, if point source is located outside the standard mesh. Defaulted to NACELL = 0, if at least one of the variables NRCELL, NPCELL or NTCELL are larger than zero.

**NISOR** polygon index IPOLG. Meaningless if NLPLG = .FALSE.

**SORAD1** x-co-ordinate of source point X0

**SORAD2** y-co-ordinate of source point Y0

**SORAD3** z-co-ordinate of source point Z0

**SORAD4, SORAD5, SORAD6**

are the x,y and z components of a reference directional vector **C** = (CRTX, CRTY, CRTZ) which may be used to distinguish one particular direction for the distribution in velocity space (see below). Internally this vector is normalized to length 1. Irrelevant for an isotropic velocity distribution.

### Line source

to be written

### Surface source

Flags for the "distribution in physical space" not mentioned here are irrelevant for surface sources.

**NSRFSI** Total number of different surfaces, or surface segments, over which the starting points for this stratum are distributed ("sub-strata", to facilitate sampling of spatial coordinates).

The next deck of 4 input cards is read NSRFSI times, one deck for each "sub-stratum".

**INUM** irrelevant; labelling index for sub-strata

**INDIM**

= 0 source on "additional surface" ASURF (see block 3B)

= 1 source on "standard surface" RSURF, x- (or radial) mesh
(see block 2A and 3A)

= 2 source on "standard surface" PSURF, y- (or poloidal) mesh
(see block 2B and 3A)

= 3 source on "standard surface" TSURF, z- (or toroidal) mesh
(see block 2C and 3A)

= 4 source on a surface composed of one or more segments of radial and/or poloidal polygons. The further details of the spatial distribution are defined in code coupling routines, i.e., the code coupling routine IF1COP must be called. Special versions of IF1COP are available, e.g., in the code segments $COUPLE_{B2}$, $COUPLE_{B2.5}$ (coupling to B2 (BRAAMS) multi-fluid plasma code), $COUPLE_{DIVIMP}$ (coupling

to DIVIMP impurity ion kinetic transport code) or $COUPLE_{Ufile}$ (TRANSP-code format).

The position on the surface is sampled from a (piecewise constant) step function defined from plasma fluxes onto that surface vs. arc-length. The flags INSOR, INGRDA and INGRDE described below are set automatically in this option and hence need not be specified.

**INSOR** number of the surface in the mesh RSURF, PSURF, TSURF or ASURF respectively. (Redundant in case INDIM=4)

**INGRDA, INGRDE** same as IRPTA, IRPTE flags in input block 3a. Defines subrange on standard surfaces, on which the source is distributed. Irrelevant for sources on additional surfaces.

**SORWGT** Relative frequency for starting points on surface labelled INUM. The sum of SORWGT for all NSRFSI surfaces is normalized to one internally.

**SORLIM** = KLMN

if SORLIM $\leq 0$, the user supplied Subroutine SAMUSR is called to sample all 3 initial co-ordinates (X0,Y0,Z0), see section 3.4.

if SORLIM > 0, then one of the preprogrammed options is used (the digits L,M and N are relevant only for surface sources). In this case:

**N** Index to select one of the preprogrammed distributions in radial or x-direction on the surface.

**M** Index to select one of the preprogrammed distributions in poloidal or y-direction on the surface.

**L** Index to select one of the preprogrammed distributions in toroidal or z-direction on the surface.

**K** Index to select one of the preprogrammed distributions for the starting time.

**M,N,L = 0** The respective co-ordinate is computed from the 2 others and from the equation for surface number INSOR.

Thus, one and only one of these 3 digits must be equal to 0, because the birth-point for a surface source is determined already by two co-ordinates and the labeling o index of the surface.

If INDIM=0, any one of the 3 digits can be the 0, depending upon the particular equation for the surface ASURF(INSOR).

In case INDIM=1, one has to set N=0 (is now done automatically), and the poloidal (or y) and toroidal (or z) co-ordinate is sampled according to the flags M and L.

Correspondingly in case INDIM=2 one must specify M=0, and in case INDIM=3 the flag L=0 has to be set (is redundant).

**L,M,N = 1** $\delta$-distribution at (a+b)/2

**L,M,N = 2** Uniform distribution on the interval [a,b]

**L,M,N = 3** Truncated exponential decay with decay length $\lambda$ on the interval [a,b]. I.e. the sampling distribution reads:

$f(x) = c \cdot \exp(-x/\lambda)$ if $x \in [a, b]$ and $f(x) = 0$ elsewhere,

with normalizing constant

$c = \{\lambda(\exp[-a/\lambda] - \exp[-b/\lambda])\}^{-1}$

**L,M,N = 4** Step-function (see below: Function STEP, section 2.7.1) (only one of either L or M or N should be 4)

**K = 1** $\delta$-distribution at TIME0 for time of particle birth. (A delta function source in time for the kinetic equation in integral form corresponds to an initial condition for time-dependent linear kinetic integro-differential equation).

**K = 2** Uniform distribution in [TIME0,TIME0+DTIMV] for time of particle birth.

Default: K=2 in time-dependent mode (NTIME >0) and K=1, TIME0=0 in time-independent mode (NTIME = 0), see section 2.1.

**SORIND** Flag to choose one from the various step functions (SORLIM-option 4), which have been defined in the initialization phase. Up to NSTEP (PARMUSR, see section 3.1) step functions can be described there. SORIND is the labelling index of the selected step function.

Each step function STEP(ISTEP, ...) can consist of step functions for fluxes of up to NSPZ species, see section 2.7.1. NSPZ depends upon the initialization of this function. By default the source species index NSPEZ is used when sampling from step functions.

New option (Aug. 2006), e.g. for testing isotope effects: If SORIND $\geq$ 100, then the 3$^{rd}$ digit is used to select the species index from step function ISTEP. I.e.: Let SORIND = LMN, then MN is used to sample from step function ISTEP = MN for species ISPZ = L. This concerns the spatial distribution. The species index itself of the sampled particle is still determined by the flag NSPEZ, see above.

**SOREXP** Decay length $\lambda$ in the exponential distribution (option 3)

**SORIFL** The first of the 4 digits can be used to overrule the default orientation of the surface normal at the birth point, or if ILSIDE = 0 for this particular surface. If this digit is nonzero, a value 1 would lead to a test flight originating from the surface as if a particle has been incident onto this surface in the positive direction, and the value 2 means that this imaginary particle has been striking in the negative direction.

The last 3 digits of SORIFL act as LMN of the ILSWCH flag described in section 2.3B assuming incidence in the positive direction.

If any of this 3 digits equals zero, than the ILSWCH flag for this particular surface is activated. (See also: section 2.3B, input flag ILSWCH)

**NRSOR** as for point source, but additionally:

If NRSOR < 0, NRCELL is found from the step-function data (see below: Function STEP, section 2.7.1) (if N is equal to 4) or is returned from SAMUSR (if SORLIM < 0)

**NPSOR** as for point source, but additionally:

If NPSOR < 0, NPCELL is found from the step-function data (see below: Function STEP, section 2.7.1) (if M is equal to 4) or is returned from SAMUSR (if SORLIM < 0)

**NTSOR** as for point source, but additionally:

If NTSOR < 0, NTCELL is found from the step-function data (see below: Function STEP, section 2.7.1) (if L is equal to 4) or is returned from SAMUSR (if SORLIM < 0)

**NBSOR** as for point source, but additionally:

If NBSOR < 0, NBLOCK is found from the step-function data (see below: Function STEP, section 2.7.1) (if N is equal to 4) or is returned from SAMUSR (if SORLIM < 0)

**NASOR** as for point source, but additionally:

If NASOR < 0, NACELL is found from the step-function data (see below: Function STEP, section 2.7.1) (if N is equal to 4) or is returned from SAMUSR (if SORLIM < 0)

**NISOR** as for point source, but additionally:

If NISOR < 0, IPOLG is found from the step-function data (see below: Function STEP, section 2.7.1) (if N is equal to 4) or is returned from SAMUSR (if SORLIM < 0)

The six parameters in the next card SORAD... are used to define the sampling intervals in the three co-ordinate directions x, y and z. For some geometry options and surface types (i.e., *radial, poloidal, toroidal,* or *additional*) these boundaries of the sampling interfaces, in some co-ordinates, are automatically found from the specified range [INGRDA, INGRDE] in the corresponding grid, and the corresponding SORAD... values are then not used for those coordinates. In case of doubt see initialization phase of subroutine SAMSRF to find out which options precisely are available, or contact the EIRENE team at FZ-Jülich.
Only in case of *additional surfaces* the information on this deck is always fully used.

**SORAD1** Left endpoint a of interval [a,b] for x- or radial co-ordinate

**SORAD2** Right endpoint b of interval [a,b] for x- or radial co-ordinate

**SORAD3** as SORAD1, for y- or poloidal co-ordinate

**SORAD4** as SORAD2, for y- or poloidal co-ordinate

**SORAD5** as SORAD1, for z- or toroidal co-ordinate

**SORAD6** as SORAD2, for z- or toroidal co-ordinate

The reference direction unit vector $\mathbf{C} = (CRTX, CRTY, CRTZ)$ for the velocity space distribution (see paragraph below: velocity space sampling) is, by default, the "positive outward normal vector" of surface INUM at the birth point. It thus need not be specified for the surface source option. (see "Standard Mesh Surfaces" and/or "Additional Surfaces").

**Volume source**
Flags for the "distribution in physical space" not mentioned here are irrelevant for volume sources.

**NSRFSI** Total number of subregions of the standard mesh in which the starting points for this stratum are distributed ("sub-strata", to facilitate sampling of spatial coordinates).

The next deck of 4 input cards is read NSRFSI times, one deck for each "sub-stratum".

**INUM** irrelevant; labelling index for sub-strata

**INDIM, INSOR** not in use for volume sources

**INGRDA, INGRDE** same as IRPTA, IRPTE flags in input block 3a. Defines subregion of standard mesh, on which the source is distributed.

**SORLIM** if SORLIM $\leq 0$, the user supplied Subroutine SAMUSR is called to sample all 3 initial co-ordinates (X0,Y0,Z0), see section 3.4.

If SORLIM $> 0$, then the preprogrammed option is used.

**SORIND** identifies volume recombination reaction IRRC for bulk plasma species IPLS, as specified in input block 5, reaction decks for bulk ions. (E.g., to distinguish between effects from three-body, radiative and di-electronic recombination).

SORIND=KREC=IRRC

If more than one recombination process is specified for bulk ion species IPLS, then IRRC is the number of one particular such process, counted by the sequence of input in input block 5. See printout activated by the TRCAMD flag (block 11) for the correct value of IRRC in case of doubt.

In case SORIND=0 the sum over all relevant IRRC for the selected background species IPLS is taken as "'recombination"' source for IPLS. (Only for EIRENE version 2001 or younger).

Note: Both the source strength FLUX and the relative weight of subregions (if any) SORWGT are automatically determined from the volume source data on the atomic data array TABRC1(IRRC,ICELL).

**SORAD1** unused. Parameter for user supplied sampling routine SAMUSR

**SORAD2** unused. Parameter for user supplied sampling routine SAMUSR

**SORAD3** unused. Parameter for user supplied sampling routine SAMUSR

**SORAD4, SORAD5, SORAD6**

are the x,y and z components of a reference directional vector **C** = (CRTX, CRTY, CRTZ) which may be used to distinguish one particular direction for the distribution in velocity space (see below). Internally this vector is normalized to length 1. Irrelevant for an isotropic velocity distribution. (Same as for point sources, see above)

**Distribution in velocity space**

Together with a position (and time) $\mathbf{r}_0, t_0$ of a source particle, species index $isp$, also a specific reference vector $\mathbf{C} = (CRTX, CRTY, CRTZ)$ for sampling in velocity space has been selected,

e.g.: a surface normal vector for surface sources, or a preferential direction of emission from a point source or volume source, etc., see below.
Let $\mathbf{C} = (C_X, C_Y, C_Z)$ be this reference directional unit vector defined together with the point of birth of the test particle.
Furthermore, a set of local background ("plasma") data at the birth point has been provided during spatial source sampling. I.e.:

$$n_i(ip, \mathbf{r}, t), T_e, T_i(ip, \mathbf{r}, t), \mathbf{V}_i(ip, \mathbf{r}, t), E_i(ip, \mathbf{r}, t), Sh \tag{2.10}$$

with $ip = 1, NPLS$, all background medium particle species, for ion density, electron and ion temperature, ion flow velocity, mean incident ion energy and target surface sheath potential, respectively. These local "birth point background parameters" are either set in case of step-function sampling, FUNCTION STEP, or can be provided in case of user-source-sampling (SAMUSR), i.e., if SORLIM < 0, or are computed from the known cell numbers and the input background tallies at the place of birth.
Depending upon the value of the flag NEMODS, the following energy distributions $f(E|\mathbf{r}_0, t_0, isp)$ are available. Note that this conditional energy distribution $f(E|\dots)$ may also depend upon the further input parameters
**SORENI, SORENE, SORVDX, SORVDY, SORVDZ**
and is conditional on the birth position (and time) $\mathbf{r}_0, t_0$ of the test particle via local plasma parameters $T_e, T_i, \mathbf{V}_i, \dots$, as well as on the selected species index $isp$ of the newly launched test particle.
$T_i, \mathbf{V}_i$ and $E_i$ may be different for different background ion species $ip, ip = 1, NPLS$. The choice of a particular $ip_0 = IPL$ from these NPLS background species indices, for velocity space sampling, if test particle $isp$ is to be launched, is described below, digit K of the NEMODS flag. Furthermore, let $\mathbf{V}_{i,\pi}(ip_0)$ and $\mathbf{V}_{i,\sigma}(ip_0)$ be the components of the velocity drift vector $\mathbf{V}_i(ip_0)$ normal and parallel, respectively, to a (possibly fictitious) surface element (similar notation as in section 1.5.1), the orientation of which is locally defined at the initial position of the test particle by the above mentioned reference "outward normal unit vector" $\mathbf{C}$.
In the earliest applications of EIRENE to plasma target surface recycling sources only the electrostatic sheath action was included. The velocity $\mathbf{V}_{i,\pi}$ corresponded then to the parallel (to a magnetic field $\mathbf{B}$) plasma flow velocity (often: sonic) onto a target surface which was assumed to be essentially orthogonal to the magnetic field $\mathbf{B}$ (e.g. a poloidal limiter side surface). In case of toroidal belt limiters or poloidal divertor targets the velocity $\mathbf{V}_{i,\pi}$ is interpreted as the parallel plasma flow velocity at the electrostatic sheath entrance. I.e. the bending of the sonic or supersonic flow $\mathbf{V}_{i,\|}$ parallel to the magnetic field at the magnetic pre-sheath entrance to a sonic flow $\mathbf{V}_{i,\pi}$ normal to the target entering the electrostatic sheath, by the action of the magnetic pre-sheath field, is not carried out inside EIRENE. Check section 1.6 for generalizations, e.g. to also include magnetic pre-sheath models inside EIRENE.

**NEMODS** = KLMN

The choice of one of the following energy distributions is made depending upon the value of the first digit N of NEMODS:

**N = 1** Mono-energetic source $f(E) = \delta(E - E0)$ with fixed energy

E0 = SORENI (eV)

**N = 2** Mono-energetic source $f(E) = \delta(E - E0)$ with energy

E0 = SORENI $\cdot T_i(ip_0)$ + SORENE $\cdot T_e$

**N = 3** (only for surface sources) As N=2, but with added sheath acceleration, see section 2.7.2 below

**N = 4** Mono-energetic source f(E) = $\delta$ (E-E0) with energy

E0 = EMAX$(T_i(ip_0), V_{i,\pi}(ip_0), V_{i,\sigma}(ip_0))$

where EMAX is the mean energy from a truncated (one-sided) shifted Maxwellian flux at temperature $T_i(ip_0)$ drifting by a velocity

$(V_{i,\pi}(ip_0), V_{i,\sigma}(ip_0))$.

One obtains (see (1.77), omitting unnecessary indices) in chapter 1) by integration EMAX = $\gamma_E T_i(ip_0)$

with $\gamma_E = (\mathcal{V}_\pi^2 + 2 + \mathcal{V}_\sigma^2) + 0.5\frac{g(\mathcal{V}_\pi)-1}{g(\mathcal{V}_\pi)}$,

$\mathcal{V} = V/\sqrt{\frac{2T_i}{m_i}}$, and $g(x) = 1 + \sqrt{\pi}x[1 + \mathrm{erf}(x)]\exp(x^2)$.

(The normalized velocities $\mathcal{V}, \mathcal{V}_\pi, \mathcal{V}_\sigma$ used here coincide with the isothermal Mach number if $T_i(ip_0) = T_e$, see again section 1.5.1).

**N = 5** (only for surface sources) As N=4, but with added sheath acceleration, see section 2.7.2 below

E0 = EMAX$(T_i(ip_0), V_{i,\pi}(ip_0), V_{i,\sigma}(ip_0))$ + ESHEAT

**N = 6** *surface source*

The velocity vector $\mathbf{V}_0 = (VX_0, VY_0, VZ_0)$ is sampled from a truncated Maxwellian flux $f_{flux} = \mathbf{V} \cdot \mathbf{C} f_{max}(T_i(ip_0), \mathbf{V}_i(ip_0), \mathbf{C})$ with a drifting Maxwellian $f_{max}$ at temperature $T_i(ip_0)$ and drift velocity $\mathbf{V}_i(ip_0)$. The (one-sided) flux is across a surface element described locally by the outward normal vector $\mathbf{C}$. The mean energy from this sampling distribution is that given by option N=4.

*point or volume source*

The velocity vector $\mathbf{V}_0 = (VX_0, VY_0, VZ_0)$ is sampled from a truncated Maxwellian density distribution $f_{max}(T_i(ip_0), \mathbf{V}_i(ip_0), \mathbf{C})$ at temperature $T_i(ip_0)$ and which is shifted by a velocity $\mathbf{V}_i(ip_0)$

**N = 7** (only for surface sources) As N=6, but with added sheath acceleration, see section 2.7.2 below

**N = 8** Mono-energetic source $f(E) = \delta(E - E0)$ with energy

$E0 = E_i(ip_0)$, with $E_i(ip_0)$ being the local mean incident ion energy, for example defined in connection with the spatial "step-function" option, see section 2.7.1.

**N = 9** (only for surface sources) As N=8, but with added sheath acceleration, see section 2.7.2 below

The choice of temperature parameters $T_e$ and $T_i(ip_0)$ in the energy distributions N = 4, 5, 6 or 7 of the plasma fluids entering the sheath region is controlled by the second digit M of NEMODS:

**M = 0** $T_e$ is the local electron temperature taken from input tally TEIN on the grid.

In case of NLPLS, the default parameter $T_i(ip_0)$ is the local ion temperature for bulk ion species $ip_0 = IPL$, IPL is the species index ISP of the source particle itself, by default.

In case of NLION, the default parameter $T_i(ip_0)$ is the local ion temperature for bulk ion species $ip_0 = IPL$, IPL is determined such that the mass and charge number and the charge state of the source particle IION match the mass, charge number and charge state of the bulk ion IPL. If no such bulk ion is found, $T_i(ip_0) = 0$.

These default choices of background medium species index $ip_0 = IPL$ can be overruled by the K digit of this input flag, see below.

In case of .not.(NLPLS.or.NLION), $T_i(ip_0)$ =0.

**M = 1** The local ion temperature $T_i$ is replaced by the input constant SORENI (eV) and $T_e$ is replaced by the constant SORENE (eV) in options N = 4 to N = 7.

**M = 2** not in use

**M = 3** $T_e$ is the local electron temperature, and $T_i$ is the local background ion temperature of ion species $ip_0 = IPL = K$, K is the fourth digit of the NEMODS flag, see below.

The choice of velocity parameters **V** in the energy distributions N=4,5,6 or 7 is controlled by the third digit L of NEMODS:

**L = 0** In case of NLPLS, the parameter **V** is the local ion drift velocity for bulk ion species IPL, IPL is the species index of the source particle.

In case of .NOT.NLPLS, **V** =0.

**L = 1** The local drift velocity vector **V** is replaced by the constant vector (SORVDX, SORVDY, SORVDZ) (cm $s^{-1}$), regardless of the point of birth.

**L = 2** The local drift velocity vector **V** is replaced by the constant vector (SORVDX, SORVDY, SORVDZ) · CS, where CS is the isothermal ion sound velocity for the species and at the temperatures chosen above (cm $s^{-1}$).

Therefore, (SORVDX,SORVDY,SORVDZ) are now understood as Mach numbers in x,y and z direction respectively.

**L = 3** **V** is the local background ion drift velocity of ion bulk species IPL=K, K is the fourth digit of the NEMODS flag, see below.

#### K options

The digit K (unless zero) is used to specify one of the background ion species IPL, from which the $T_i(IPL)$, $\mathbf{V}(IPL)$ parameters in the sampling distributions described above are taken.

### SORCOS, SORMAX

Depending upon the value of the flag **NAMODS** (see below) various different angular distributions may be selected. Each one depends upon the two parameters P = SORCOS and Q = SORMAX.

**SORCTX, SORCTY, SORCTZ**

If this input vector is NOT zero:

The unit outward pointing reference vector **C** mentioned above is overwritten and explicitly now given by normalization of SORCTX, SORCTY, SORCTZ.

Else (default):

As described above this vector **C** is internally found as the outward surface normal vector (in case of surface sources) at $\mathbf{r}_0$ or (in case of point, line, or volume sources) by the input parameters $\mathbf{C} = (SORAD4, SORAD5, SORAD6)$.

By this option, for example, the mean direction of emission from a surface or a gas puff can be influenced. In case of surface sources the distribution of the polar angle may then not necessarily be centered around the inner surface normal vector any longer, but may be tilted.

**NAMODS**

= 1 The polar angle $\vartheta$ against the reference unit vector $\mathbf{C} = (C_X, C_Y, C_Z)$ of the source particle's velocity is sampled from a cosine**P distribution around the "inner normal vector" (-1.0) ·**C**, i.e., $f(\vartheta)d\vartheta \sim \sin(\vartheta) \cdot \cos^p(\vartheta)d\vartheta$.

Important special cases:

**P = 0** isotropic distribution

**P = 1** cosine distribution

**P ≫ 1** close to $\delta$-distribution around vector $-1 \cdot \mathbf{C}$

= 2 The polar angle against $-1 \cdot \mathbf{C}$ is sampled from a Gaussian distribution with zero mean value, and the parameter P now is used for the standard deviation (degree) of that distribution.

The second parameter Q is the cut-off angle (degree) for the polar angle distribution **note:**

$Q \leq 180°$ is enforced internally

$Q \leq 90°$ is enforced internally for surface sources

$Q = 0$ for a beam, i.e., for an angular $\delta$-distribution at $-1 \cdot \mathbf{C}$

In all preprogrammed cases the azimuthal angle (around the axis **C**) is equally distributed from 0° to 360°.

### 2.7.1 Piecewise constant "Step-functions" for sampling

The statement:
A = STEP(NSPZI,NSPZE,NSMAX(ISTEP),ISTEP)
initializes (piecewise constant) step functions for random sampling by the inversion method, ISTEP is an index labeling one such a step function, and there may be NSTEP different step functions (species index) with one common set of abscissae.
The statement:
B = STEP1(IINDEX,ISTEP,RNF,ISPEZ)

converts a uniformly distributed random number RNF into a random number sampled from the step function no. ISTEP.

Sampling of the birth-point of a trajectory is sometimes done from such piecewise constant functions (EIRENE function STEP(. . . ) ), e.g., if one digit of the flag SORLIM has been set to the value 4. This is the case if, for example, a recycling ion flux distribution is given in discretized form on a grid along a target surface (from an external **CFD!** plasma solver).
The random sampling of a coordinate $u$ is from the step-function FLSTEP(RRSTEP). There is storage for up to NSTEP (see PARMUSR) such step-functions.
$u_i = RRSTEP(i, ISTEP), i = 1, \ldots, NSMAX(ISTEP) - 1$, is a discrete ordered (either increasing or decreasing) set of abscissae $u_i$, at which the piecewise constant step-function has a jump to $FLSTEP(u_i)$.
The final interval ends at RRSTEP(NSMAX(ISTEP),ISTEP)
Internally the piecewise constant function $FLSTEP$ is translated into a (normalized) cumulative distribution $VF(u)$ for random number generation. FLSTEP itself is not necessarily normalized. Let RNF be a uniform random number on the interval

$$0.0 = \text{VF(RRSTEP(1,ISTEP))} \rightarrow \text{VF(RRSTEP(NSMAX),ISTEP)} = 1.0,$$

or on a sub-interval thereof.
By calls to the entry STEP1(IINDEX,ISTEP,RNF,ISPEZ) of Function STEP the randomly sampled index $i$=IINDEX is returned, as well as a coordinate $u$ sampled from an uniform distribution on the $i^{th}$ increment $\Delta u_i$ = RRSTEP(i+1, ...) − RRSTEP(i, ...) : $u$=STEP1
In case of LEVGEO=1 or LEVGEO=2 $u$ has the simple meaning of one of the spatial coordinates in an EIRENE run.
However, $u$ need not necessarily be one of the 3 spatial coordinates. RRSTEP can also stand, e.g., for an arc-length along a polygon (LEVGEO=3), or for the accumulated length of an element in a selected list of sides of triangles (LEVGEO=4) or for the accumulated area of a surface element in a selected list of surfaces of tetrahedrons (LEVGEO=5). In these latter cases only the index IINDEX is used, and a point from an uniform distribution on the line-, or surface element, to which IINDEX points, must still be sampled.
The probability density functions FLSTEP for sampling a position (or cell index) RR are set in function STEP from a piecewise constant distribution function (e.g., from a surface flux density distribution). In the same call to STEP also their corresponding cumulative functions VF are obtained by integration of FLSTEP over RRSTEP, and normalization.
These step-functions have to be set in the initialization phase (by calls to function STEP from SAMUSR, INFUSR, etc.).
The initialization call to function STEP reads:
FLUX = STEP(NSPZI,NSPZE,NSMAX(ISTEP),ISTEP)
This call will initialize the step-function no. ISTEP, for a range of species ISPZ=NSPZI, NSPZE. At this point the grid RRSTEP and the distribution density FLSTEP on that grid must be defined (module CSTEP). RRSTEP is an ordered set of numbers, the abscissae of the points when FLSTEP jumps from one value to another. RRSTEP may have dimension length, area, volume or any other dimension of the independent variable for the piecewise constant distribution FLSTEP. In the initialization call to function STEP the distribution density FLSTEP (Flux per unit length or area or volume, or whatever the dimension of RRSTEP is.) is then converted into a (set of)

cumulative distribution $VF$ by

$$VF(ISPZ, ISTEP, i) = \sum_{j=1}^{j=i} \Delta u_j \cdot FLSTEP(ISPZ, ISTEP, j)$$

with, again, $\Delta u_J = RRSTEP(J+1, ISTEP) - RRSTEP(J, ISTEP)$,
one for each species index separately. This cumulative flux-distribution is then in units of a flux (source strength), typically in EIRENE: in [amp].
After this the total flux (normalization constant) FLTOT(ISPZ, ISTEP) = VF(ISPZ, ISTEP, NSMAX) is stored, and then the cumulative distributions $VF$ are normalized to

$$VF(\ldots, i) = VF(\ldots, i)/FLTOT(\ldots), \quad i = 1, NSMAX(\ldots).$$

hence enforcing $VF(ISPZ, ISTEP, NSMAX) = 1$ for random sampling.
The pdf FLSTEP, the cumulative distributions VF and normalization factors FLTOT themselves are stored in module *CSTEP.f*, as well as the normalized inverse functions (quantile functions) of the cumulative distributions, which are needed for efficient random sampling.
Default initializations of functions STEP are carried out in the initialization of surface source sampling routine SAMSRF, entry SMSRF0.
Random sampling using such step-functions STEP is done, e.g., from surface source sampling routine SAMSRF, entry SMSRF1.
In addition to the value of the distribution FLSTEP(i), the normalization constants FLTOT and the normalized cumulative VF(i) in each interval some further quantities may be defined as function of the interval index "*i*":

**TESTEP** electron temperature at the surface segment "*i*"

**TISTEP** ion temperature at the surface segment "*i*" (array of length NPLS )

**DISTEP** ion density at the surface segment "*i*" (array of length NPLS)

**IRSTEP** cell number in 1st (x or radial) grid of the surface segment "*i*".

**IPSTEP** cell number in 2nd (y or poloidal) grid of the surface segment "*i*".

**ITSTEP** cell number in 3rd (z or toroidal) grid of the surface segment "*i*".

**IBSTEP** block number, see section 2e

**IASTEP** additional cell number of the surface segment "*i*"

**ELSTEP** ion energy flux at the surface segment "*i*" (array of length NPLS)

**SHSTEP** electrostatic sheath potential at the surface segment "*i*"

**Default step functions**
Step functions for sampling the radial coordinate (at a given or sampled poloidal and toroidal position) (INDIM = 1, SORLIM-digit N = 4) are defined internally during the initialisation step of surface sampling routine SAMSRF if the selected step functions SORIND = ISTEP has not

been defined externally already. Also in case of unstructured grids (LEVGEO = 4, 5) default step functions can be defined for non-default surfaces.
In such cases the particle flux spatial sampling distribution FLSTEP of background species $ip$ on a radial grid segment $i$ or triangle side $i$ or tetrahedron side $i$ is computed as

$$FLSTEP(ip, i) = n_i(ip) \times cs(ip) \tag{2.11}$$

with $n_i(ip)$ the local ion density of species $ip$ (=DISTEP(ip,i)) and $cs(ip)$ is the isothermal acoustic speed of ion species $ip$, i.e.

$$cs(ip) = \sqrt{[Z_i T_e + T_i(ip)]/m_i(ip)}. \tag{2.12}$$

Here $Z_i$ denotes the ion electric charge.
The mean incident ion energy $ELSTEP = E_i$ corresponds to that resulting from the NEMOD = 2, 3 options, with $E_i = 3.0T_i + 0.5Z_iT_e$ and the sheath potential $SHSTEP = Sh$ is set to the sheath potential flag for the non-default surface: FSHEAT, see input block 6.
Note that by setting SHSTEP and hence parameter $Sh$, the evaluation of the full sheath potential function $\Delta\Phi$ in energy sampling options NEMODS, digit N=3,5,7 and 9 is overruled by using $\Delta\Phi = Sh$ instead.

### 2.7.2 Electrostatic sheath acceleration

For surface sources, and ionic source particles (test ions or bulk ion, with charge $Z_i$) an acceleration of the sampled velocity vector $\mathbf{v}_0$ towards the surface, in the direction normal to the target surface, may be added. I.e. the velocity space sampling described above (NEMODS options, digit N) is regarded to provide the velocity at the entrance of the electrostatic sheath in front of a target surface.
Energy sampling options NEMODS N= 3,5,7,9 add this sheath contribution to velocities sampled from the corresponding options N= 2,4,6, and 8, respectively.
Let $\mathbf{v}_0 = \mathbf{V}_0$. This sheath acceleration is achieved by setting a new velocity $\mathbf{V}_1$:
$\mathbf{V}_1 = \mathbf{V}_0 + \mathbf{V}_S$, $\mathbf{V}_S = V_S \cdot \mathbf{C}$ and $V_S$ evaluated such that

$$m_i \cdot V_S\,(\mathbf{V}_0\mathbf{C}) + \frac{m_i}{2}V_S^2 = ESHEAT(n_i, \vec{v}_{\pi,i}, T_e), \tag{2.13}$$

i.e., the energy $E$ of the particle is increased from $E_0$ to $E_1$ by the amount ESHEAT (eV) as compared to the option without sheath acceleration, and this increased energy comes from an additional speed component $V_s$ normal towards the target surface.
The sheath acceleration $ESHEAT = Z_i \cdot \Delta\Phi \cdot T_e$ is computed from the sheath voltage $\Delta\Phi$, where $Z_i$ is the electric charge of the sampled particle at birth point, $Z_i = 0$ for neutral particles. The sheath potential drop $\Delta\Phi$ is given by (1.80) in chapter 1, unless overruled by parameter $Sh > 0.$, (see (2.10), in which latter case $ESHEAT = Z_i \cdot Sh \cdot T_e$.
If $Sh \leq 0$ and the sheath potential according to equation (1.80) cannot be found either (e.g. because $\mathbf{V}_{\pi,i}(ip) \equiv 0.$), then $Sh = FSHEAT(MSURF)$ (the input sheath parameter for source surface no. MSURF, input blocks 3 and 6) is used.

## 2.8 Additional Data for some Specific Zones

**General remarks**

Input data in this block permit explicit specification of plasma parameters $T_e$, $T_i$, $D_i$, $V_x$, $V_y$, $V_z$ and of zone volumes VOL in selected cells ICELL. Furthermore, information may be given to the geometrical block of EIRENE that some "additional surfaces" are "invisible" for a particle located in cell ICELL and, therefore, possible crossings need not be checked for advancing this particle at the next step. (Intelligent use of this option can lead to a considerable speeding up of the code, less intelligent use will lead to dramatic errors.) All this information will be included in the EIRENE arrays after the initialization phase (Subroutines INPUT, PLASMA, GRID, VOLUME) and before setting the derived profiles ($D_e$, flux-surface labelling grids, . . . )

**The Input Block**

```
*** 8. Data for some specific zones
      NZADD
      DO 81 IZADD=1,NZADD
*     comment (optional card)
        INI INE
```

then read an arbitrary number of cards each starting with either T, D, V, M, VL or CH3 in any order

```
   81 CONTINUE
```

The following specifications are applied to each cell with cell number ICELL in the range $INI \leq ICELL \leq INE$.

**"CH3-cards"** (format: 3A,69A)

arbitrary number of strings ± n/m, separated by blanks. n and m must be integer variables $1 \leq$ n,m $\leq$ NLIMI. By default EIRENE assumes that any additional surface is "visible" from any cell of the computational box. A string $n/m$ (or $+n/m$) has the effect that during particle tracing possible crossings of additional surfaces number $n$ to number $m$ are not checked whenever a track starts in cell ICELL. By $-n/m$ these surfaces are activated again (in principle this option is not needed due to default setting).

A particle history can be forced to stop and then to restart in a cell e.g. by making appropriate use of the ILIIN = -1 option for additional- or non default standard surfaces.

**"T-cards"** T, IDION, TEADD, TIADD(IDION)

format: 1A,1I6,6X,2E12.4

plasma temperatures $T_e$ and $T_i$ in cell ICELL are reset to:

TEIN(ICELL)=TEADD

TIIN(IDION,ICELL)=TIADD(IDION)

**"D-cards"** D, IDPLS, DIADD(IDPLS)

format: 1A,1I6,6X,1E12.4

plasma ion density $D_i$ for species IDPLS in cell ICELL is reset to:

DIIN(IDPLS,ICELL)=DIADD(IDPLS)

**"V-cards"** V, IDPLS, VXADD(IDPLS), VYADD(IDPLS), VZADD(IDPLS)

format: 1A,1I6,6X,3E12.4

plasma drift velocity in x direction $V_x$ for species IDPLS in cell ICELL is reset to:

VXIN(IDPLS,ICELL)=VXADD(IDPLS)

VYIN,VZIN: likewise, for the drift velocities in y and z direction respectively.

**"M-cards"** M, IDPLS, MXADD(IDPLS), MYADD(IDPLS), MZADD(IDPLS)

same as "V-cards", but drift velocity in cell ICELL is given in Mach number units:

$VXIN = MXADD \cdot \frac{\sqrt{T_e+T_i}}{m_i}$,

$m_i$ being the mass of bulk ions of species IDPLS.

VYIN,VZIN are computed in the same way from MYADD and MZADD respectively.

**"VL-cards"** VL, VLADD

format: 2A,1E12.4

The zone volume in cell ICELL is explicitly set to:

VOL(ICELL)=VLADD

## 2.9 Data for Statistics and non-analog Methods

### General remarks

The statistical performance (**FOM!**, see (1.24) in section 1.3.3) of an EIRENE run (as for any Monte Carlo application in general) is very sensitive to the non-analog methods used by EIRENE. Therefore the input parameters in this block, which define the non-analog setting of an EIRENE run, should only be used, if the user has carefully worked through the code and has a detailed knowledge of the Monte Carlo techniques activated by setting the flags in this block.
Only the "Cards for Standard Deviation", which allow to obtain graphical and printed output regarding statistical standard deviation for all tallies estimated by EIRENE, can be used with no risk, except that evaluations of variances and covariances generally tend to slow down the code a bit (few percent) in case of large meshes.
Hence, as a rule, one should generally try to find the optimal setting of the non-analog sampling flags in this block and estimate the **FOM!** from test runs, and then use this setting, but without evaluation of standard deviations, in production runs.
The input flag NLANA in block 1 (section 2.1) de-activates all non-analog sampling distributions. This option should, for example, be used to derive an intuitive but physically correct picture from plots of selected particle trajectories (input block 11, section 2.11), which otherwise, e.g. in case of suppression of absorption, may be grossly misleading.

### The Input Block

```
*** 9. Data for Statistics and non-analog Methods
       (NLPRCA(IATM) IATM=1,NATM)
      .   (NLPRCM(IMOL)   IMOL =1,NMOL)
      .   (NLPRCI(IION) IION=1,NION)
      .   (NLPRCPH(IPHOT) IPHOT=1,NPHOT)
       NPRCSF
C next: read NPRCSF integers (FORMAT 12I6)
C         IPRSF(J) J=1,NPRCSF
       MAXLEV  MAXRAD  MAXPOL  MAXTOR  MAXADD
       DO IN=1,MAXRAD
         READ (IUNIN,66665) ID,NSSPL(IN),PRMSPL(IN)
       ENDDO
       DO IN=1,MAXPOL
         READ (IUNIN,66665) ID,NSSPL(N1ST+IN),PRMSPL(N1ST+IN)
       ENDDO
       DO IN=1,MAXTOR
         READ (IUNIN,66665) ID,NSSPL(N1ST+N2ND+IN),
      .                     PRMSPL(N1ST+N2ND+IN)
       ENDDO
       DO IN=1,MAXADD
         READ (IUNIN,66665) ID,NSSPL(N1ST+N2ND+N3RD+IN),
      .                     PRMSPL(N1ST+N2ND+N3RD+IN)
       ENDDO
       WMINV  WMINS  WMINC WMINL
       SPLPAR
```

** Cards for Standard Deviation*

```
NSIGVI   NSIGSI   NSIGCI   NSIGI_BGK   NSIGI_COP   NSIGI_SPC
DO  93   J=1,NSIGVI
   IGH   IIH
93 CONTINUE
DO  94   J=1,NSIGSI
   IGHW  IIHW
94 CONTINUE
DO  95   J=1,NSIGCI
   IGHC(1,J)  IIHC(1,J)  IGHC(2,J)  IIHC(2,J)
95 CONTINUE
```

**Meaning of the Input Variables for Statistics and non-analog Methods**

**NLPRCA** conditional expectation estimator (1.21) is used for atom species IATM

**NLPRCM** conditional expectation estimator is used for molecule species IMOL

**NLPRCI** conditional expectation estimator is used for test ion species IION (not ready to use)

**NLPRCPH** conditional expectation estimator is used for photon species IPHOT (in versions 2004 and younger)

**IPRSF** conditional expectation estimator is used, if trajectory points towards additional surface IPRSF. IPRSF ≤ NLIMI, the total number of additional surfaces read in input block 3B.

NPRCSF surfaces have that property of "attracting trajectories".

The next lines define so called Splitting and Russian Roulette surfaces (S&R-surfaces). Any test-particle crossing a S&R-surface in the negative direction is split into SPLPAR identical but independent particles with properly reduced weight. If a particle crosses a S&R-surface in the positive direction, then it is either killed or continues it's flight with properly increased weight. The surface orientation can be reversed by using a negative value for the the splitting parameter SPLPAR.

**MAXLEV** Maximum number of levels for splitting (≤ 15)

**MAXRAD** Total number of radial splitting surfaces (-NR1ST ≤ MAXRAD ≤ NR1ST)

< 0 -MAXRAD is used, the position of the radial splitting surfaces is automatically defined, and a constant splitting parameter (SPLPAR, see below) is used for radial splitting and RR.

> 0 radial surfaces with numbers NSSPL(IN), IN=1,MAXRAD are S&R-surfaces. The splitting parameter for surface NSSPL(IN) is PRMSPL(IN).

**MAXPOL** Total number of poloidal splitting surfaces (0 ≤ MAXPOL ≤ NP2ND). The S&R-surfaces and splitting parameters are selected as in the case of radial surfaces, see above.

**MAXTOR** Total number of toroidal splitting surfaces (0 ≤ MAXTOR ≤ NT3RD). The S&R-surfaces and splitting parameters are selected as in the case of radial surfaces, see above.

**MAXADD** Total number of additional splitting surfaces ($0 \leq$ MAXADD $\leq$ NLIMI). The S&R-surfaces and splitting parameters are selected as in the case of radial surfaces, see above.

**WMINV** minimum weight used for suppression of absorption at collisions ("survival biasing"). If a particle goes into a collision with weight less than WMINV, then suppression of absorption or any other non-analog weight correction is abandoned, and the analog game is played. WMINV acts only for events in the volume including volume source birth events, but not for events at surfaces.

**WMINS** Same as WMINV, but for surface events (including surface source birth events).

**WMINC** minimum acceptable weight for conditional expectation estimators (by abuse of language). More precisely, WMINC is the minimal acceptable probability for a test flight to reach a particular cell without collision. If this probability is smaller than WMINC, the particle track is stopped and restarted. E.g. for WMINC $\geq$ 1, the estimator used in the NIMBUS code results (ref. [**kn:Cupini**]), whereas for WMINC = 0 each particle path is integrated according to (1.21), until the nearest non transparent surface along the track is reached, regardless of any collisions. Periodicity surfaces are regarded as "transparent" in this context.

Note: strictly speaking this is not a non-analog method, but rather a particular choice of an unbiased estimator. Hence: the flag NLANA in input block 1 does not affect flags for conditional expectation estimators.

**WMINL** to be written

**SPLPAR** splitting parameter for default radial splitting option (MAXRAD $< 0$)

**NSIGVI** number of standard deviation profiles for volume averaged tallies to be estimated. NSIGVI must be less than or equal to the parameter NSD in PARMUSR, section 3.1.

**NSIGSI** number of standard deviation profiles for surface averaged tallies to be estimated. NSIGSI must be less than or equal to the parameter NSDW in PARMUSR, section 3.1.

**NSIGCI** number of correlation coefficients between volume tallies to be estimated. NSIGCI must be less than or equal to the parameter NCV in PARMUSR, section 3.1.

**NSIGI_BGK** if NSIGI_BGK $> 0$, then the standard deviations are evaluated for all tallies needed for the iteration procedure for the non-linear **BGK!** collision terms. See subroutine STATIS_BGK in code segment BGK.F

**NSIGI_COP** if NSIGI_COP $> 0$, then the standard deviations are evaluated for all tallies needed for the iteration procedure for the coupling to a plasma fluid model. The relevant tallies are selected in subroutine STATIS_COP in code segment *COUPLE_....F*, section 4.2

**NSIGI_SPC** if NSIGI_SPC $> 0$, then the standard deviations are evaluated for all surface flux spectra defined in sub-block 10F below.

**IGH** Index of species for selected volume averaged tally

**IIH** Index of volume averaged tally for which empirical standard deviation is to be calculated (see table 6.3).

**IGHW** Index of species for selected surface averaged tally

**IIHW** Index of surface averaged tally for which empirical standard deviation is to be calculated (see table 6.4)

**IGHC1 IIHC1 IGHC2 IIHC2** Species and tally index for first and second tally, respectively, between which the correlation coefficient is evaluated

Note: On printout the NSIGVI and NSIGSI standard deviations are printed as relative standard deviations, in percent. The standard deviations for the two tallies involved in the NSIGCI arrays are printed as absolute tallies, i.e., with the same units as the two tallies themselves.

## 2.10 Data for additional volume and surface averaged tallies

### General remarks

There is a large number of preprogrammed "default" volume or surface averaged tallies, which have been selected mainly to allow assessment of global particle and energy balances for all test particle species, as well as coupling of neutral gas and plasma transport equations. These tallies are estimated at each EIRENE run unless they are explicitly abandoned by the surface crossing switches ILSWCH (section 2.3.2), or are turned off in input block 11 (section 2.11).
There is, however, a very general option to allow for estimation of many more moments of the test particle $\mu$ -space distribution function by resorting to appropriately prepared problem specific routines UPTUSR, UPCUSR and UPSUSR. These are described in more detail in section 3.2. Furthermore there are options to form algebraic expressions from tallies, and, since Eirene$_{2003}$ also for particle and energy flux spectra evaluated from the fluxes on selected surfaces or in selected cells of the computational grid. The input data in block 10 described here are flags for activating and controlling these routines.
Default: If the first card in this block: NADVI, ..., contains only zeros, then the rest of this block can/must be omitted.

### The Input Block

```
*** 10. Data for additional tallies
      NADVI  NCLVI  NALVI  NADSI  NALSI NADSPC
* 10A. Data for additional track-length estimated volumetric tallies
      DO 101 IADVI=1,NADVI
        IADVE(IADVI) IADVS(IADVI) IADVT(IADVI) IADVR(IADVI)
        TXTTAL(IADVI,NTALA)
        TXTSPC(IADVI,NTALA) TXTUNT(IADVI,NTALA)
  101 CONTINUE
* 10B. Data for additional collision estimated volumetric tallies
      DO 102 ICLVI=1,NCLVI
        ICLVE(ICLVI) ICLVS(ICLVI) ICLVT(ICLVI) ICLVR(ICLVI)
        TXTTAL(ICLVI,NTALC)
        TXTSPC(ICLVI,NTALC) TXTUNT(ICLVI,NTALC)
  102 CONTINUE
* 10C. Data for algebraic function of volumetric tallies
      DO 103 IALVI=1,NALVI
        ALSTRNG
        TXTTAL(IALVI,NTALR)
        TXTSPC(IALVI,NTALR) TXTUNT(IALVI,NTALR)
  103 CONTINUE
* 10D. Data for additional surface crossing tallies
      DO 104 IADSI=1,NADSI
        IADSE(IADSI) IADSS(IADSI) IADST(IADSI) IADSR(IADSI)
        TXTTLS(IADSI,NTLSA)
        TXTSPS(IADSI,NTLSA) TXTUNS(IADSI,NTLSA)
  104 CONTINUE
* 10E. Data for algebraic function of surface crossing tallies
```

```
      DO 105 IALSI=1,NALSI
        ALSTRNG
        TXTTLS(IALSI,NTLSR)
        TXTSPS(IALSI,NTLSR) TXTUNS(IALSI,NTLSR)
  105 CONTINUE
```

optional, in versions Eirene$_{2003}$ and younger:

```
* 10F. Data for energy spectra
      DO 106 IADSP=1,NADSPC
        ISPSRF IPTYP IPSPZ ISPTYP NSPS ISRFCLL IDIREC
        SPCMN SPCMX SPC_SHIFT SPCPLT_X SPCPLT_Y SPCPLT_SAME
        if (idirec.ne.0) SPCVX SPCVY SPCVZ
  106 CONTINUE
```

**Meaning of the input variables for additional volume and surface averaged Tallies**

**NADVI** Total number of additional volume averaged, track-length estimated tallies

**NCLVI** Total number of additional volume averaged, collision estimated tallies

**NALVI** Total number of tallies defined as algebraic expressions of other volume averaged tallies

**NADSI** Total number of additional surface averaged tallies

**NALSI** Total number of tallies defined as algebraic expressions of other surface averaged tallies

**NADSPC** Total number of surface or cell averaged energy spectra

### 2.10.1 Additional volume averaged tallies, track-length estimator

**IADVE** flag for scaling factor for this tally (carried out in subroutine MCARLO)

= 1 scale tally per unit volume [cm$^{-3}$]. The tally is printed and plotted in the units

[cm$^{-3}$] · [units of $g^*$] · [cm] · [source strength FLUX]

however, with FLUX converted to units [s$^{-1}$] (rather than input units [$Ampere$]). For the definition of the detector functions $g$ and $g^*$ see section 3.2, the variable FLUX is explained in input block no. 7 (section 2.7). This scaling is default for "density tallies" of particles, momentum and energy.

= 2 scale tally per unit cell. Same units as above, however not per cm$^3$ but per cell instead.

= 3 same as IADVE = 1, but with FLUX in Ampere, rather than s$^{-1}$. This scaling is default for "source rate tallies", e.g. for particle, momentum and energy sources.

= 4 same as IADVE = 2, but with FLUX in Ampere, rather than s$^{-1}$.

**else** no re-scaling done, units as chosen in subroutine UPTUSR.

Note: scaling of EIRENE default tallies 1 to 8 (particle and energy densities, respectively) and tallies 85 to 96 (momentum density) is as for IADVE=1 option. Scaling of all default source and sink terms is as for IADVE=3 option.
If instead also for particle- or energy densities tallies 1 to 8 the scaling IADVE=3 would have been used, then, for example, particle densities would have been in units [amp s $cm^{-3}$] ("atomic charge density") rather than [$cm^{-3}$]. Energy densities would have been in [watt $\cdot$ s $cm^{-3}$] rather than [eV $cm^{-3}$].

**IADVS** species index of default volume averaged tally, which is to be replaced by this additional track-length estimated tally. In case of tallies with no species index, IADVS must be set equal to 1.

**IADVT** number of default volume averaged tally, which is to be replaced by this track-length estimated tally.

If either IADVS or IADVT is out of range, no default tally is replaced by this additional tally.

**IADVR** If automatic re-scaling of volumetric tallies is performed (i.e. if NLSCL = TRUE), then re-scale this tally with EIRENE recommended factor FATM (IADVR = 1), FMOL (IADVR = 2) or FION (IADVR = 3) (see block 1 for the flag NLSCL and the variables FATM,FMOL,FION)

**TXTTAL** Text to label tally on numerical or graphical output.

**TXTSPC** Text describing the species of particles contributing for this tally in output routines.

**TXTUNT** Text describing the units of this tally in output routines.

### 2.10.2 Additional volume averaged tallies, collision estimator

**ICLVE** as IADVE above, for collision estimated tally (subroutine UPCUSR).

**ICLVS** species index of default volume averaged tally, which is to be replaced by this collision estimated tally. In case of tallies with no species index, ICLVS must be set equal to 1.

**ICLVT** number of default volume averaged tally, which is to be replaced by this collision estimated tally.

If either ICLVS or ICLVT is out of range, no default tally is replaced.

**ICLVR** as IADVR above, for collision estimated tally (subroutine UPCUSR).

**TXTTAL, TXTSPC, TXTUNT** : as above

### 2.10.3 Algebraic expression in volume averaged tallies

**ALSTRNG** character string which is interpreted as an algebraic expression in some volume averaged tallies. An operand $\langle i, j \rangle$ stands for tally number $j$, first (species) index $i$, in table 6.2 (input tallies) and table 6.3 (output tallies) . Note that the first index for tallies with no species index must read 1, and the tally number of input tallies must be negative. Example: $\langle 1, -1 \rangle$ for the electron temperature tally, $\langle 2, 3 \rangle$ for the particle density of test ion species no. IION=2 . Expressions $\langle c \rangle$ with an integer or real constant $c$ are interpreted as scalars. The string may contain an arbitrary (but $\leq 20$) number of operands, and of operators +, -, *, /, **, and of properly nested parentheses (. . . ).

With Eirene version Eirene_revised_input_tallies evaluation of some mathematical functions have been introduced. Available functions consist of "ABS" (absolute value), "EXP" (exponential function), "LN" (natural logarithm), "LOG" (decimal logarithm). Additionally functions "DX", "DY" and "DZ" for calculation of partial derivatives in the respective carthesian coordinate are available. For the moment the code is only able to store two intermediate result for further calculations. Thus the expression for calculating the divergence of the transport flux of atom species 1 which in this case consists of D atoms

$$(DX(\langle 1, 85 \rangle) + DY(\langle 1, 89 \rangle) + DZ(\langle 1, 93 \rangle))/\langle 1.66D - 24 \rangle/\langle 2.E0 \rangle$$

will result in a warning messages. Nevertheless this problem can be overcome by introducing an additional pair of parenthesis. Thus

$$((DX(\langle 1, 85 \rangle) + DY(\langle 1, 89 \rangle)) + DZ(\langle 1, 93 \rangle))/\langle 1.66D - 24 \rangle/\langle 2.E0 \rangle$$

will produce the correct result.

Standard deviations are not available, generally, for algebraic tallies. For linear combinations of tallies it is, in principle, possible to obtain also the standard deviations, this evaluation of error estimates for such combinations of tally is, however, currently carried out only in a proprietary code segment (available from the author).

E.g., the total electron particle source due to test particle - plasma interaction in units: $\#/s/m^3$ can be obtained by the line:

$(\langle 1, 7 \rangle + \langle 1, 12 \rangle + \langle 1, 17 \rangle) * \langle 1.e6 \rangle/\langle 1.6022e - 19 \rangle$

in code versions older than 2002 (see tables in section 6.1.2), and the same expressions in versions 2002 and younger, i.e. after implementation of photons as further species type (and the related default tallies, tables in section 6.1.1):

$(\langle 1, 9 \rangle + \langle 1, 15 \rangle + \langle 1, 21 \rangle) * \langle 1.e6 \rangle/\langle 1.6022e - 19 \rangle$

and it would be stored on the tally: "ALGV" with the first (labelling) index IALVI.

Note that the cell volume array VOL is regarded as a volume averaged tally, by abuse of language (tally number = -14, see table 6.2).

**TXTTAL, TXTSPC, TXTUNT** as above.

#### 2.10.3.1 frequently used algebraic tallies

- mean energy

  The mean energy (eV) of a test particle species is the ratio of its total energy density (= total pressure) and particle density. The algebraic tally $\langle i,5\rangle/\langle i,1\rangle$ is that ratio estimate, here exemplarily for atomic species no. IATM=$i$. Similarly, for molecular species IMOL=$k$, this expression would read: $\langle k,6\rangle/\langle k,2\rangle$

- kinetic temperature

  The kinetic temperature $T_t$ (eV) is defined such that $3/2\ T_t$ is the mean thermal energy. The kinetic temperature is obtained by first subtracting the dynamic pressure (contribution from drift motion) from the total energy density, and then by rescaling.

$$(\langle i,5\rangle - \langle 1.8792e+35\rangle/\langle 2.0\rangle \\ * (\langle i,85\rangle * \langle i,85\rangle + \langle i,89\rangle * \langle i,89\rangle + \langle i,93\rangle * \langle i,93\rangle) \quad (2.14)$$

$$/\langle i,1\rangle)/\langle i,1\rangle/\langle 1.5\rangle \quad (2.15)$$

  Here the subtracted part $\langle 1.8792e+35\rangle/\langle 2.0\rangle * (\langle i,85\rangle * \langle i,85\rangle + \langle i,89\rangle * \langle i,89\rangle + \langle i,93\rangle * \langle i,93\rangle)/\langle i,1\rangle)$ is the dynamic pressure, scaled to eV cm$^3$ units of atoms, with atomic weight $\langle 2.0\rangle$ in atomic mass units in this example. The numerical factor $1.8792e+35$ is, in EIRENE unit conversion variables, is given by CVELI2/AMUA/AMUA, (defined in SETCON.F), and the momentum density tallies 85, 89 and 93 are in g cm s$^{-1}$ cm$^{-3}$.

### 2.10.4 Additional surface averaged tallies

as in sub-block 10A or 10B, but for surface averaged tallies rather than volume averaged tallies. Note: for surface averaged tallies collision and track-length estimators are identical, see section 3.2.4.

### 2.10.5 Algebraic expression in surface averaged tallies

as in sub-block 10C except for the calculation of partial derivatives, but for surface averaged tallies rather than volume averaged tallies. The corresponding tally numbers are listed in table 6.4.
There is one generalization available here compared to the options for volume tallies, which permits algebraic expressions not only between different complete surface tallies (expression evaluated for all NLIMPS surfaces), but also between different individual surfaces.
If the first index I is out of range, i.e., larger than the first dimension N1 of the tally J as listed in table 6.4, then the operand $\langle I,J\rangle$ identifies the following estimate:
surface index $ILIMPS = (I - N1 - 1)/N1 + 1$
species index $ISPZ = I - ILIMPS \cdot N1$
The other way round: in order to identify species ISPZ and surface no. ILIMPS for a particular tally J, on must set:
first operand $I = ILIMPS \cdot N1 + ISPZ$
second operand J
The algebraic expression is stored on the ALGS tally for all surfaces ILIMPS, which occurred in at least one of the operands.

Example: let there be up to 3 atom species (N1=NATM=3), and assume, that additional surfaces no. 7,8,9,10 and 11 comprise the pump. Let furthermore IATM=2 stand for helium atoms. The total pumped flux of helium atoms from these surfaces is the difference between incident and re-emitted He particle fluxes, summed over these 5 surfaces. This is evaluated by the string:
$\langle 23,1\rangle - \langle 23,2\rangle + \langle 26,1\rangle - \langle 26,2\rangle + \langle 29,1\rangle -$
$\langle 29,2\rangle + \langle 32,1\rangle - \langle 32,2\rangle + \langle 35,1\rangle - \langle 35,2\rangle$
The value found for this expression is stored on ALGS (. . . , ISURF), and is the same for ISURF=7, =8, =9, =10 or =11, and zero for all other surfaces.

### 2.10.6 Energy spectra in selected cells or surfaces

Additional, spectrally resolved, tallies, either surface averaged or cell based. Printout of these spectra is activated in input block 11a, flag NSPCPR.

**ISRFCLL** flag to choose between surface and cell averages.

=0: surface averaged spectrum

=1: volume (cell) averaged spectrum

**ISPSRF** number of surface or cell, for which spectra are to be evaluated. Positive for additional surfaces, negative for non-default standard surfaces.

**IPTYP** Type of particle (0 for photons, 1 for atoms, 2 for molecules, 3 for test ions, 4 for bulk ions)

**IPSPZ** species index of particle, within the specified type-category. If IPSPZ=0, then sum over species, for this type of particles.

**ISPTYP** type of spectrum:

**for surface averaged spectra (i.e. ISRFCLL=0):**

flux-spectrum (ISPTYP=1), [amp $eV^{-1}$]

or energy-flux spectrum (ISPTYP=2) [watt $eV^{-1}$]

Currently only surface "incident fluxes". Spectra for surfaces "emitted fluxes" (outgoing) are not yet available.

**for cell averaged spectra (i.e. ISRFCLL=1):**

**case a:** no direction specified: IDIREC=0

energy distribution (ISPTYP=1), [$cm^{-3}$ $eV^{-1}$]

energy weighted energy distribution (ISPTYP=2) [eV $cm^{-3}$ $eV^{-1}$]

velocity weighted energy distribution (ISPTYP=3) [cm $s^{-1}$ $cm^{-3}$ $eV^{-1}$]

**case b:** direction specified: IDIREC=1

velocity distribution along specified direction, binned in energy units (ISPTYP=1), [$cm^{-3}$ $eV^{-1}$]

energy weighted velocity distribution along specified direction, put into bins in energy units (ISPTYP=2) [eV $cm^{-3}$ $eV^{-1}$]

velocity weighted velocity distribution along specified direction, put into bins in energy units (ISPTYP=3) [cm $s^{-1}$ $cm^{-3}$ $eV^{-1}$]

**IDIREC, SPCVX, SPCVY, SPCVZ** These flags are only for cell averaged spectra (otherwise: not in use):

=0: no direction specified

=1: cell averaged spectra are along direction SPCVX, SPCVY, SPCVZ. The vector defining this direction is normalized internally.

**NSPS, SPCMN, SPCMX** Number of equidistant energy bins (eV) for spectra. The contributions with energies below the minimum energy SPCMN are stored in bin no. 0, those with energies above the maximum energy SPCMX are stored in bin NSPS+1.

NSPS > 0: The bins are set equidistant in linear energy scale

NSPS < 0: |NSPS| is used for the number of bins, but the bins are set equidistant in a logarithmic energy scale (option added by X.B., 2017). Same as for NCHENI parameter in input block 12 for (post processed) line of sight integrated and spectrally resolved data.

**SPC_SHIFT** shift spectrum by SPC_SHIFT (eV) in printout and plots.

**SPCPLT_X, SPCPLT_Y** SPCPLT_X is a flag foreseen to control the X axis (independent variable) in a plot of the spectrum. (X: e.g. the no. of an energy bin). This flag is currently not in use.

SPCPLT_Y is a flag to control the Y axis (dependent variable) in a plot of the spectrum. (Y: e.g. particle or energy flux in an energy bin). IF SPCPLT_Y > 0: logarithmic scale for Y axis, otherwise: linear scale for Y axis. .

**SPCPLT_SAME** plot this spectrum into previous frame

The surface flux spectra are printed together with the other surface averaged tallies, for those surfaces, which are selected for printout in sub-block 11A. They are all automatically plotted as soon as at least one further volume tally plot (input sub-block 11B.2) is selected.

## 2.11 Data for numerical and graphical Output

### General remarks

The input flags in this block control all the numerical and graphical output of an EIRENE run. This comprises diagnostics during the initialization phase (e.g. 2d and 3d geometry plots) as well as selected test particle histories printed and plotted during their generation. All numerical output of a run is arranged in so called "tallies". There are volume averaged tallies, surface averaged tallies ("surface crossing tallies") and global tallies. These latter tallies are derived from the former ones by integration over the total computational volume, or over all non-transparent surfaces, respectively.
Each volume averaged tally is an array TALLYV(I) of NSBOX data, comprising the responses for the respective detector function $g_V$ in each zone of the computational mesh (see section 2.2, "General Remarks").
Each surface averaged tally is an array TALLYS(I) of NLIMPS data, comprising the responses for the respective detector function $g_S$ integrated over each (non default standard or additional) surface.
Each volume or surface tally is characterized by its index ITALV or ITALS respectively, augmented by a first (species) index in some cases. The first index, if there is one, is always referred to as "species index" (by abuse of language), even for additional tallies ADDV or ALGV in which cases it might have a different meaning.
As for the volume tallies there are NTALV preprogrammed detector functions, up to NADV user supplied detector functions for track-length estimators (tally no. ITALV) and NCLV user supplied detector functions for collision estimators (tally no. ITALV). Furthermore there are NSNV snapshot tallies (tally no. ITALV), NCPV tallies for coupling with plasma background codes (tally no. ITALV) and NBGV tallies for the (non-linear) **BGK!** collision integrals, needed for the iteration procedure (tally no. ITALV). Up to NALV algebraic expressions in these tallies (tally no. ITALV) can be stored.
There is also one tally no. 0, which is evaluated in subroutine TALUSR (section 3.7). This tally is printed immediately after evaluation, i.e., it is not put onto any storage (unless explicitly done so in subroutine TALUSR).
The input volume tallies for the background medium are listed in table 6.2. They are selected for printout or graphical output by the **negative** value of their tally number. E.g., the choice ITALV = 5 selects tally no. 5 from table 6.3, and the choice ITALV = -5 selects tally no. 5 from table 6.2.
Furthermore, there are NTALS preprogrammed detector functions for surface averaged tallies, and there may be up to NADS additional user supplied surface averaged tallies (tally no. ITALS).
The preprogrammed tallies and their default units are given in table 6.3 (volume averaged tallies) and in table 6.4 (surface averaged tallies).
table 6.2 lists the NTALI (volume-) tallies which are computed from the input data in block 5 and which describe the background medium (plasma). By an abuse of language the cell volumes ($cm^{-3}$) stored in the array VOL(I) are also referred to as an input volumetric tally (no. ITALI=13 in table 6.2).

### The Input Block

```
* 11 Data for numerical and graphical output
* 11A Block for numerical output written on unit 6 (IUNOUT)
```

```
      TRCPLT   TRCHST   TRCNAL   TRCREA   TRCSIG
      TRCGRD   TRCSUR   TRCREF   TRCFLE   TRCAMD
      TRCINT   TRCLST   TRCSOU   TRCREC   TRCTIM
      TRCBLA   TRCBLM   TRCBLI   TRCBLP   TRCBLE
      TRCBLPH  TRCTAL   TRCOCT   TRCCEN   TRCRNF
      TRCHKTIM TRCSCL
      TRCSRC(J), J= 0, NSTRAI
      NVOLPR, NSPCPR
      DO 101 J=1,NVOLPR
        NPRTLV(J) NFLAGV(J) NSPEZV(J,1) NSPEZV(J,2) NTLVFL(J)
  101 CONTINUE
      NSURPR
      DO 102 J=1,NSURPR
        NSRF NPRTLS(J) NFLAGS(J) NSPEZS(J,1) NSPEZS(J,2)
     .    NTLSFL(J)
  102 CONTINUE
```

optional: unless block 11B starts:

```
      NLTVOUT
C (12 values per card, FORMAT 12I6)
      NUMTAL(J), J=1,NLTVOUT
      NLTSOUT
C (12 values per card, FORMAT 12I6)
      NUMTAL(J), J=1,NLTSOUT

* 11B Data for graphical output
* 11B.1 Data for 2D and 3D Geometry plot
      PL1ST     PL2ND     PL3RD     PLADD  PLHST
      PLCUT(1)  PLCUT(2)  PLCUT(3)  PLBOX  PLSTOR
      PLNUMV    PLNUMS    PLARR
      NPLINR NPLOTR NPLDLR NPLINP NPLOTP NPLDLP NPLINT NPLOTT
     .  NPLDLT
      DO 1140 J=1,5
        PL3A(J) TEXTLA(J) IPLTA(J)
     .    (IPLAA(J,I) IPLEA(J,I) I=1,IPLTA(J))
 1140 CONTINUE
      DO 1141 J=1,3
        PL3S(J) TEXTLS(J) IPLTS(J)
     .    (IPLAS(J,I) IPLES(J,I) I=1,IPLTS(J))
 1141 CONTINUE
      CH2MX  CH2MY           CH2X0  CH2Y0  CH2Z0
      CH3MX  CH3MY  CH3MZ  CH3X0  CH3Y0  CH3Z0
      ANGLE1  ANGLE2  ANGLE3
      I1TRC  I2TRC  (ISYPLT(J)  J=1,8)  ILINIE
* 11B.2 Data for plots of volume tallies
      NVOLPL
```

```
      PLTSRC(J), J= 0, NSTRAI
      DO 116 N=1,NVOLPL
C optional: one text card starting with *... to label the
C           plot
        NSPTAL(N)
        PLTL2D(N) PLTL3D(N) PLTLLG(N) PLTLER(N)
        TALZMI(N) TALZMA(N) TALXMI(N) TALXMA(N) TALYMI(N)
     .    TALYMA(N)
        IF (PLTL2D(NTL)) THEN
          LHIST2(N) LSMOT2(N)
          DO 118 I=1,NSPTAL(N)
            ISPTAL(N,I) NPTALI(N,I) NPLIN2(N,I) NPLOT2(N,I)
            NPLDL2(N,I)
 118      CONTINUE
        ENDIF
        IF (PLTL3D(N)) THEN
          LHIST3 LCNTR3 LSMOT3 LRAPS3 LVECT3 LRPVC3 LRAPS3D
     .      LR3DCON
          LPRAD LPPOL LPTOR
          DO 119 I=1,NSPTAL(N)
            ISPTAL(N,I)  NPTALI(N,I) IPROJ3(N,J)
            NPLI13(N,I)  NPLO13(N,I)
            NPLI23(N,I)  NPLO23(N,I) IPLANE(N,I)
 119      CONTINUE
          TALW1(N)  TALW2(N)  FCABS1(N)  FCABS2(N) RAPSDEL(N)
        ENDIF
 116  CONTINUE
```

**Meaning of the Input Variables for numerical and graphical Output**

* 11*A. Block for numerical output written on unit* 6

**TRCPLT** Trace-back from plot routines

**TRCHST** Printout of trajectories of selected test particle histories into geometry plots. These histories are selected by the flags I1TRC and I2TRC, see below, sub-block 11B.2.

**TRCNAL** Trace-back for non-analog methods, i.e. splitting surfaces, suppression of absorption, weighted post-collision species sampling etc.

**TRCMOD** Trace-back from routines for iterative mode (MOD_TMSTEP, MODBGK, and problem specific routines called from MODUSR).

**TRCSIG** Trace-back from post-processing line integral diagnostics block DIAGNO, line of sight integration routines.

**TRCGRD** Printout of "standard mesh surface data"

**TRCSUR** Printout of data for "additional surfaces"

**TRCREF** Printout of reflection model related data. In particular a list of all non-perfect recycling surfaces is printed, i.e., a list of surfaces for which there is some absorption at least for one incident particle species.

**TRCFLE** Trace-back from subroutines WRSTRT, WRGEOM, WRPLAS (writing on and reading from the dump files FT10, FT11, FT12, FT13 etc.

**TRCAMD** Trace-back from atomic and molecular data routines

XSECTA, XSECTM, XSECTI

**TRCINT** Trace back from user specified interfacing routine INFCOP, e.g. of data related to coupling of EIRENE to other codes.

**TRCLST** Printout of information during the last history of each stratum. E.g., sampling efficiencies, and other accumulated information, which is only available in the history generation routines during particle tracing.

**TRCSOU** Trace back from primary source sampling routines, e.g. from LOCATE, SAMPNT, SAMLNE, SAMSRF, SAMVOL and SAMUSR

**TRCREC** Printout of EIRENE recommendations for next run on same case:

A) stratified source sampling, at present: for proportional allocation of weights.

B) weight windows, at present: to be written

**TRCTIM** Printout CPU-time information for each history

**TRCBLA** Global particle and energy balance for atoms is printed.

**TRCBLM** Global particle and energy balance for molecules is printed.

**TRCBLI** Global particle and energy balance for test ions is printed.

**TRCBLP** Global particle and energy balance for bulk ions is printed.

**TRCBLE** Global particle and energy balance for electrons is printed.

**TRCBLPH** Global particle and energy balance for photons is printed.

**TRCTAL** Print list of activated and de-activated tallies. The default settings eliminate some tallies from storage and estimators which are likely to be irrelevant in a particular run, e.g. all photon related tallies are deactivated automatically if no radiation transfer calculation is included in a run. These default settings are overruled by the flags NTLVOUT and NTLSOUT at the end of this sub-block 11A, see below.

**TRCOCT** to be written: printout from octree geometry optimization procedure

**TRCCEN** printout from census array stored in time dependent mode: species and stratum resolved census fluxes

**TRCRNF** printout from random number generation routines: setting seed, etc., mostly for testing "correlated sampling" options (flag: NLCRR).

**TRCHKTIM** printout of CPU usage during particle tracing and scoring (excluding overhead), separated by particle type, species and strata.

**TRCSCL** test output of quantities entering the species specific scaling procedure

**TRCSRC(ISTRA), ISTRA = 0, NSTRAI**

The selected global, volumetric and surface crossing tallies are printed for those strata for which TRCSTR(ISTRA) = .TRUE.. In case TRCSTR(0) = .TRUE., the results after summation over all strata is printed for these tallies.

Note: printout for individual strata is only possible if the data for strata have been saved on file (NFILE-N=1,2 option, input block 1). Otherwise only the last stratum currently on storage arrays (mostly: sum over strata) is available.

**NVOLPR** Total number of volume averaged tallies to be printed.

**NSPCPR** Flag for printout of surface or cell based spectra (defined in input block 10F) If NSPCPR > 0: spectra are printed on output stream fort.(20+ioff), name: spectra.out

**NPRTLV** Index of the tally to be printed (first column in table 6.2 or table 6.3). If the tally has a species index, it is printed for all species, for which the integral of the tally over the computational area is nonzero. Otherwise the statement

ZERO INTEGRAL FOR SPECIES ISPZ = . . .

is printed below the header for this tally. The term "species index" is used here in a more general sense for the first index (if any) for any given tally. In case of some "additional tallies" or other tallies, in which the first index does not label a particle species but something else, this term "species index" then is to be understood in a more general sense.

If NPRTLV = 0, then the user defined post-processing routine TALUSR is called (section 3.7).

**NFLAGV** Flag to specify the level of printout.

= −1 print only header

= 0 additionally: print global quantities (total and block averages)

= 1 additionally: print 1D profiles (if any), averaged over all (if any) higher dimensions

= 2 additionally: print 2D profiles (if any), averaged over all (if any) higher dimensions

= 3 additionally: print 3D profiles (if any), averaged over all (if any) higher dimensions

= 4 print 3D profiles, but no lower dimensional averages

**NSPEZV** If NSPEZV(. . . , 1) ≠ 0, then this tally is printed only for the species index range:

$I1 = NSPEZV(\ldots, 1)$ to $I2 = MAX(I1, NSPEZV(\ldots, 2))$,

rather than for all species relevant for the specified tally.

**NTLVFL** In addition to the standard output stream fort.IUNOUT, this tally is also printed onto output stream fort.NTLVFL, for all species selected, and also the corresponding standard deviations are printed, if available. The format for this extra output stream is specified in subroutine PRTTAL, in code segment EIRASS.

The next paragraph describes printout for surface averaged tallies. Distinct from the volume averaged tallies, where selected tallies (NPRTLV) are printed for all the computational domain, no particular surface tallies are selected, but always all kinds of surface fluxes (tallies) are printed (particle fluxes, incident, emitted, energy fluxes, sputtered fluxes, . . . ). A choice is made now instead with respect to the spatial region: the particular surfaces (internal EIRENE surface numbers) are selected here by input flag NSRF.

**NSURPR** Total number of surfaces, for which surface averaged tallies are to be printed.

**NSRF** Index of the surface to be printed. By default all tallies listed in tables 6.4, 6.5 and 6.8 (depending on code version) are printed for this surface.

**NPRTLS(J) NFLAGS(J) NSPEZS(J,1) NSPEZS(J,2)** These next flags are only needed for those surfaces, for which a further spatial resolution within one surface is provided (this is enabled by providing storage through setting the flag NGSTAL = 1 in input block 1). Their meaning then corresponds to the meaning of flags NPRTLV(J), NFLAGV(J), NSPEZV(J,1), NSPEZV(J,2) for volume tallies, respectively. If the NPRTLS, . . . flags are not specified (i.e.: default=0), then no spatially resolved surface tallies are printed.

**NTLSFL(J)** The spatially resolved tallies (if any) and/or the flux energy spectra (if any, see input block 10F) are printed on additional output stream fort.NTLSFL(J) for this surface.

The next, optional input cards can be used to overrule the default choices of tallies which are available in a run. The logical flag TRCTAL described above can be used to print a list of all activated and deactivated tallies.

**NTLVOUT** total number of volume averaged tallies to be explicitly abandoned or enabled

**NUMTAL(J)** Number of a tally from table 6.3, e.g. NUMTAL(J)=1 for neutral atom density tally PDENA. The tally with this number NUMTAL(J) is explicitly enabled. With an additional negative sign this tally is removed from the run (and the balances), e.g. NUMTAL(J)=-2 would remove the evaluation (and storage) of tally PDENM from this run.

**NTLSOUT** to be written

* 11*B*.1 *Data for* 2*D plot of geometry*

**PL1ST** plot the x- (radial) standard grid surfaces into a 2D geometry plot

**PL2ND** plot the y- (poloidal) standard grid surfaces into a 2D geometry plot

**PL3RD** plot the z- (toroidal) standard grid surfaces into a 2D geometry plot

**PLADD** plot the additional surfaces into a 2D geometry plot

**PLHST** Plot the track of some selected test particle histories into the geometry plot (2D or 3D).

**PLCUT** Flag for the choice of a plane, in which the 3 dimensional geometrical configuration is plotted in case of a 2D geometry plot. This plane may be defined by either x = CONST, y = CONST or by z = CONST.

**PLCUT(1)** = .TRUE.

x = CONST ; plotting plane is the yz-plane.

y is the ordinate, z is the abscissa

**PLCUT(2)** = .TRUE.

y = CONST ; plotting plane is the xz plane.

z is the ordinate, x is the abscissa

**PLCUT(3)** = .TRUE.

z = CONST ; plotting plane is the xy plane.

y is the ordinate, x is the abscissa

**PLBOX** plot box defined by surface inequalities in addition to valid part of surface

**PLSTOR** produce file of coordinates along 2D projections of "additional surfaces" for later use in some graphics ("PATRAN format", also for RAPS-graphics, see below, sub-block: 11B3). These coordinates are stored on arrays XPL2D, YPL2D in subroutine STCOOR called from subroutine PLTADD.

**PLNUMV** print cell number into standard mesh cells

**PLNUMS** print numbers near additional surfaces

**PLARR** plot arrows to indicate the outer surface normal vector (only in LEVGEO=3, partly in LEVGEO=2, for non-default standard surfaces (input block 3a). Not functional (any more?) for additional surfaces.

**PLIDL** print output files for further processing in external tools (e.g. for FZJ proprietary IDL graphics tool). The grid information as well as all input and output tallies (excluding the de-activated ones) are printed on files, one file per tally. See calls to routines OUTIDL. . . from main program EIRENE.

**CH2MX** horizontal half width of 2D plot window (cm)

**CH2MY** vertical half width of 2D plot window (cm)

**CH2X0** horizontal co-ordinate of midpoint of 2D plot window (cm).

**CH2Y0** vertical co-ordinate of midpoint of 2D plot window (cm).

**CH2Z0** distance CONST of plotting plane to origin.

**NPLINR NPLOTR NPLDLR**

radial standard surfaces with labels

IR1ST = NPLINR, NPLOTR, NPLDLR

are plotted.

**NPLINP NPLOTP NPLDLP**

poloidal standard surfaces with labels

IP2ND= NPLINP, NPLOTP, NPLDLP

are plotted.

**NPLINT NPLOTT NPLDLT**

toroidal standard surfaces with labels

IT3RD= NPLINT, NPLOTT, NPLDLT

are plotted.

* 11*B*.2 *Data for* 3*D plot of geometry*

**CH3MX** half width of plot chamber in x- direction, used for 3D geometry plot

**CH3MY** half width of plot chamber in y- direction, used for 3D geometry plot

**CH3MZ** half width of plot chamber in z- direction, used for 3D geometry plot

**CH3X0** x-co-ordinate of midpoint of plot-chamber in user co-ordinates, 3D geometry plot only

**CH3Y0** y-co-ordinate of midpoint of plot-chamber in user co-ordinates, 3D geometry plot only

**CH3Z0** z-co-ordinate of midpoint of plot-chamber in user co-ordinates, 3D geometry plot only

**ANGLE1, ANGLE2**

First and second viewing angle for 3D geometry plot

The following 5 cards specify up to 5 groups of additional surfaces to be plotted on the 3D geometry plot. Each group may consist of subgroups, and each subgroup is defined by an interval of additional surface labeling indices ILIMI, from the full range of all (i.e NLIMI) additional surfaces.

**PL3A** logical flag, indicating if the additional surfaces specified in this card are to be plotted or not. If PL3A=.FALSE., the rest of this card is irrelevant

**TEXTLA** text written onto plot, characterizing this group of additional surfaces

**IPLTA** number of different subgroups of additional surfaces comprising this group

**IPLAA, IPLEA**

each subgroup consists of additional surfaces ranging from no. IPLAA(J) to IPLEA(J), J=1,IPLTA

The following 3 cards reading PL3S, . . . specify a group of standard mesh surfaces to be plotted on the 3D geometry plot. The meaning of the flags in each card is the same as above for the additional surfaces. The first card refers to the first (x or radial) standard mesh, the second card refers to the second (y or poloidal) standard mesh, the third card refers to the third (z or toroidal) standard mesh. Each group may consist of subgroups, and each subgroup is defined by an interval of standard surface labeling indices ISURF, from the full range of all (i.e NR1ST, NP2ND or NT3RD, resp.) standard surfaces.

**I1TRC** Number of first particle history to be traced in printout and/or 2D or 3D plot

**I2TRC** Number of last particle history to be traced in printout and/or 2D or 3D plot

**ISYPLT(J), J = 1, 8**

Indices to plot symbols along the particle tracks for different events. Up to 8 different events can be picked in any order in the array ISYPLT.

**0** no symbol

**1** symbol at particle's birth point (at primary source)

**2** symbol at the point of an electron impact collision event

**3** symbol at the point of a hard elastic collision event

**4** symbol at the point of a charge-exchange event

**5** symbol at the point of a soft elastic collision event

**6** symbol at an intersection with a "non-default" or "additional" surface

**7** symbol at a non-analog particle splitting point

**8** symbol at a non-analog particle killing point

("Russian Roulette")

**9** symbol at a periodicity surface intersection point

**10** symbol at restart after splitting

**11** symbol at collision point saved for conditional exp. estimator

**12** symbol at a continuation of track for conditional exp. estimator

**13** symbol at a particle stopped at time limit (t-dep. mode)

**14** symbol at a particle stopped at generation limit (t-dep. mode)

**15** If an error is detected, by default a symbol is always plotted. Plotting this symbol cannot be abandoned.

**ILINIE**

$\neq 0$ connect two successive events by a straight line. If a continuation of a track is computed for the conditional expectation estimators, this part is represented by a dotted line, see (1.21).

$= 0$ only symbols (if any selected) will be plotted along the history

* 11*B*3 *Graphical output of volume averaged tallies*

**NVOLPL** Total number of pictures from volume averaged tallies.

**PLTSRC(ISTRA), (ISTRA=0,NSTRAI)** Profiles of the selected volumetric tallies are plotted for those strata for which PLTSRC(ISTRA) = .TRUE. In case PLTSRC(0) = .TRUE., the summation over all strata is plotted for these tallies.

Note: graphical output for individual strata is only possible if the data for strata have been saved on file (NFILE-N=1,2 option, input block 1). Otherwise only the last stratum currently on storage arrays (mostly: sum over strata) is available.

**NSPTAL** Number of different "species" (quantities) to be plotted for this tally into one picture.

For 3D plots, NSPTAL = 1, except for the LVECT3 or LRPVC3 vector field options, in which case NSPTAL = 2. See below.

**PLTL2D** Switch to activate a 2D plot

**PLTL3D** Switch to activate a 3D plot

**PLTLLG** The tally is plotted on a logarithmic scale.

**PLTLER** Standard deviation is plotted if available. In case of 2D plot error bars are plotted along the curves. In case of 3D plot a full standard deviation profile is plotted on a separate picture.

**TALZMI** Minimum ordinate value for the plot.

If TALZMI = 666.0 then the minimum is searched for in the data array.

Even if PLTLLG (log. scale), the true minimum ordinate value (not the logarithm thereof) must be specified.

If TALZMI = 666.0 and PLTLLG, ZMIN=1.E-48 is used as cut-off value.

**TALZMA** like TALZMI, but maximum ordinate value for the plot.

If TALZMA = 666.0 then the maximum is searched for in the data array.

Even if PLTLLG (log. scale), the true maximum ordinate value (not the logarithm thereof) must be specified.

The following input variables are needed only if PLTL2D = TRUE

**TALXMI = TALXMA = 0.0** the radial (x-) grid is used as abscissa for plotting (default).

The next options allow to change the x axis, e.g. for cases when a meaningful radial (x-) grid does not exist for the curve to be plotted. No interpolation to a "true" radial (x-) grid is involved, so the resulting curves may change shape on the plot depending on options chosen here.

**TALXMI ≠ TALXMA**

if TALXMI < TALXMA, then a grid equidistant on a linear scale is used, TALXMI is then the minimum abscissa value for the plot, TALXMA is then the maximum abscissa value for the plot.

if TALXMI > TALXMA and both are positive, then a grid equidistant on a log. scale is used. TALXMI is then the minimum abscissa value for the plot, TALXMA is then the maximum abscissa value for the plot.

if TALXMI > TALXMA and at least one of them is negative, then a user defined grid XXP2D_USR is used. This must have been defined in one of the user-routines for this figure no. N=IBLD, i.e., the array XXP2D(J,IBLD) must have been set, J the grid index.

**ISPZTL** Index of the species of the tally that will be plotted.

If ISPZTL = 0, the tally obtained by summation over its species index is plotted.

**NPTALI** Index of tally to be plotted.

**NPLIN2** Index of the first cell that is displayed on the plot, for 2D plot only.

**NPLOT2** Index of the last cell that is displayed on the plot, for 2D plot only.

**NPLDL2** Increment for the cell indices, for 2D plot only.

The following input variables are needed only if PLTL3D = TRUE

**LHIST3** make a 3D histogram plot.

**LCNTR3** make a contour plot.

**LSMOT3** make a surface plot in a 3D cube.

**LRAPS3** produce files for RAPS graphics system. If additional surfaces are to be indicated in these blocks, PLSTOR =.TRUE., see sub-block 11B1.

**LVECT3** make a vector field plot.

NSPTAL must be 2 in this case. The first of these 2 tallies is taken as field of x - coordinates of the vector field to to be plotted, and the second tally is the y - coordinate field.

**LRPVC3** produce files for RAPS graphics system for vector field plot. Co-ordinates of vectors as in LVECT3 option. If additional surfaces are to be indicated in these blocks, PLSTOR=TRUE, see sub-block 11B1.

**LPRAD3** radial (x) co-ordinate is fixed. Plot tally versus poloidal (y) and toroidal (z) co-ordinate. See IPROJ3 flag below.

**LPPOL3** poloidal (y) co-ordinate is fixed. Plot tally versus radial (x) and toroidal (z) co-ordinate

**LPTOR3** toroidal (z) co-ordinate is fixed. Plot tally versus radial (x) and poloidal (y) co-ordinate

**ISPTAL** species index of tally to be plotted. ISPTAL = 0 means: sum over species index.

**NPTALI** Index of tally to be plotted.

**IPROJ3** in case of 3d calculation, IPROJ3 is the index of the mesh cell of the grid, for which the corresponding co-ordinate is fixed. (See LPRAD3, LPPLO3, LPTOR3 flags above).

**NPLI13**

**NPL013**

**NPLI23**

**NPLO23**

**TALW1** first viewing angle for 3D plot

**TALW2** second viewing angle for 3D plot

**FCABS1**

**FCABS2**

## 2.12 Data for Diagnostic Module

**General remarks**

The data in this block are used to define a line-of-sight (LOS) across the computational domain, along which line integrals $I$

$$I = \int_{LOS} g(l)dl = \int_{P_1}^{P_2} g(l)dl \tag{2.16}$$

are evaluated (by launching a "virtual" test particle along the LOS and scoring the response function $g(x)$ along its track, using the EIRENE particle tracing routines.
This is done in subroutine LININT, which is called from subroutine DIAGNO at the end (post processing phase) of an EIRENE run. The spatial dependence of the function $g(l)$ is defined as function of one or more of the estimated volume averaged tallies (e.g., atom density), and/or input tallies (e.g. plasma temperatures) and/or a small set of preprogrammed additional tallies, such as volumetric line emissivities for various H and He lines (e.g. Ba_alpha.f for the Balmer alpha emission rate per volume, etc.). At present there are two (version 2004 and older) or three preprogrammed functions $g(l)$, and the option to call a user supplied integrand.
Firstly there are a number of Lyman- and Balmer series volume source rates (emissivity), see Subr. SIGHA. Each hydrogenic line emissivity is available for coupling to up to 6 states ("donor states") and linear in the corresponding density (output tally), namely to the (ground) states: $H, H_2, H_2^+, H^+, H^-$ and $H_3^+$ (and their isotopes, isotopomeres. . . ).
Secondly the neutral atom charge-exchange source rate can be integrated along a line of sight for a given energy of the impacting plasma ion (Subr. SIGCX). This routine includes re-absorption along the line of sight. A LOS spectrum (also: "side-on spectrum"), energy resolved with up to NCHEN energies, may be obtained by this procedure.
Thirdly (version 2005 and younger) the side-on radiances of selected lines (photon test particle species), including reabsorption, reflection, line broadening, etc., can be obtained (Subr. SIGRAD). A number of different emission line shape profiles is available for these.
For testing and 3rd party defined further integrands $g(x)$ along lines of side there is a user defined option (Subr. SIGTST).

**The Input Block**

```
*** 12. Data for Diagnostic Module
      NCHORI  NCHENI
      IF (NCHORI.GT.0) THEN
        DO 121 ICHORI=1,NCHORI
          TXTSIG
          NSPTAL NSPSCL NSPNEW NSPCHR
          NSPSTR NSPSPZ NSPINI NSPEND NSPBLC NSPADD
          EMIN1  EMAX1 ESHIFT
          IPIVOT  XPIVOT  YPIVOT  ZPIVOT
          ICHORD  XCHORD  YCHORD  ZCHORD
  121   CONTINUE
        PLCHOR PLSPEC PRSPEC PLARGL PRARGL
      ENDIF
```

**Meaning of the Input Variables for Diagnostic Module**

**NCHORI** Total number of different line of sights

**NCHENI** |NCHENI| is the total number of energies, at which the spectrum is evaluated (irrelevant e.g. for total signals, such as Lyman and Balmer emissivity without line shape resolution).

The energy grid is equidistant on a linear scale, if NCHENI $\geq$ 0, and equidistant on a logarithmic scale otherwise. (Same as for NSPS parameter in input block 10F for spectrally resolved default tallies.)

**TXTSIG** Text in printout at the beginning of the data from this line of sight integral.

**NSPTAL** Flag for choice of preprogrammed function which is "line integrated".

= 1 charge-exchange source rate (SIGCX)

= 2 Hydrogen line emission source rate (SIGAL)

= 3 spectral radiance of photonic lines (SIGRAD) (new in versions 2005 and younger)

= 10 user supplied integrand (SIGUSR)

(this was option NSPTAL =3, in versions 2004 and older)

**NSPSCL** Flag for choice of linear or logarithmic axes in plots of spectra (vs. energy or wavelength, resp.) and of source term distribution along line of sight.

= 0 both axes linear

= 1 x axis linear, y axis logarithmic

= 2 x axis logarithmic, y axis linear

= 3 both axes logarithmic

**NSPNEW** Flag for the choice whether a spectrum is plotted on the same picture as the previous one (NSPNEW = 0) or onto a new graph (otherwise).

**NSPCHR** (Proprietary option, not for 3rd party application) If NSPCHR.gt.0, then automatically all grid cells of the computational volume along the chord are identified and marked in pre-processing. During the particle tracing phase then, energy resolved spectra (input block 10f) are computed (scored) in all these cells directly from Monte Carlo histories, by automatically augmenting input block 10f (additional energy resolved tallies: "spectra") correspondingly. These spectra are line-of-sight spectra in the direction of the chord specified here (direction SPCVX, SPCVY, SPCVZ in augmented input block 10f is taken to be a unit vector along this present line of sight).

**NSPSTR** Index for stratum, which is to be used for line integration (NSPSTR = 0: sum over strata).

**NSPSPZ** Index for species which is to be used for line integration

In case NSPTAL=1:

Index for atomic species IATM, with IATM ≤ NATM, for which charge-exchange spectrum is to be computed (NSPSPZ = 0: sum over atom species index)

In case NSPTAL=2:

(NSPSTR to be written, current default, hard wired: hydrogenic atoms, molecules, test ions, background ions).

Number of contribution to line intensity, as programmed in Subr. $Ba_{Alpha}$, $Ba_{Gamma}$, $Ly_{Beta}$, etc. (Currently up to 6 contributions for each H atom spectral line, and the total)

= 1 coupling to ground state atom: $H(1s)$

= 2 coupling to continuum, bulk ion: $H^+$

= 3 coupling to ground state molecule: $H_2$

= 4 coupling to ground state molecular ion : $H_2^+$

= 5 coupling to negative ion: $H^-$ (new in versions 1996 and younger)

= 6 coupling to: $H_3^+$ (new in versions 2012 and younger)

= 0 total, sum over all contributions to a particular line (was = 6, in versions 2011 and older)

In case NSPTAL=3:

Index for photon species (line), i.e. for IPHOT, with IPHOT ≤ NPHOT, for which side-on spectrum is to be computed (NSPSPZ = 0: sum over photon species index, not ready)

**NSPINI, NSPEND** only for NSPTAL=1:

Multipliers for the maximum ion temperature $Ti_{max}$ found along line of sight, for temperature fitting. The CX ion temperature is fitted from the CX line of side spectrum in the interval [NSPINI × $Ti_{max}$, NSPEND × $Ti_{max}$ ]

**NSPBLC** Standard mesh block number of 2nd point on line of sight.

**NSPADD** Additional cell number of 2nd point on line of sight.

If NSPADD = 0, then this 2nd point must lie in standard mesh block NSPBLC.

If NSPADD ≠ 0, then the block number NSPBLC must be NSPBLC = NBMLT+1, i.e. the second point on the line of sight is in the "additional cell region".

**EMIN1, EMAX1**

for NSPTAL=1,3,10:

minimum and maximum energy for spectral resolution, respectively.

for NSPTAL=2:

(DR, Oct. 2016 currently not available, use old option, see below)

Parameter to identify the particular hydrogen line. Emin1 and Emax1 are interpreted as integer principal quantum numbers $n, m$ of the lower and upper state for the transition.

EMIN1 = 1, EMAX1 = 3: Lyman-beta line

EMIN1 = 2, EMAX1 = 6: Balmer-delta line

EMIN1 = 2, EMAX1 = 5: Balmer-gamma line

EMIN1 = 2, EMAX1 = 4: Balmer-beta line

EMIN1 = 2, EMAX1 = 3: Balmer-alpha line

Old input version (still maintained for backward compatibility of input files: EMIN1 is an energy parameter to identify the particular hydrogen line and EMAX1 is not used. EMIN1 is given in eV, by $Ry \times \left(1/n^2 - 1/m^2\right)$, with $Ry = 13.6$ (eV).

EMIN1 = 12.089: Lyman-beta line

EMIN1 = 3.0222: Balmer-delta line

EMIN1 = 2.8560: Balmer-gamma line

EMIN1 = 2.5500: Balmer-beta line

EMIN1 = 1.8889: Balmer-alpha line

Other side on line emissivities can be obtained using the user supplied line of side integral SIGUSR (option NSPTAL=10) and analogy to the preprogrammed options, as well as the internal EIRENE hydrogen atom collisional radiative routine *H-COLRAD.F* to obtain the required reduced population coefficients for $H^*(n)$. For the preprogrammed options these latter coefficients for $n = 2, 3, 4, 5, 6$ are stored in AMJUEL, section H12, see this web page under: EIRENE AMS data files.

**ESHIFT** (for NSPTAL=1, 3, 10 options only)

energy shift for spectral resolution in printout, and plot

The line integration is carried out along a line defined by two points, a pivot point $P_{pivot}$ and a second point $P_{scnd}$. The second point must lie inside mesh block no. NSPBLC. Starting from this "second point", and moving in the direction towards the "first point" $P_{pivot}$, the first intersection $P_1$ with a non transparent additional or non default standard surface is computed. This is the starting point for line of sight integration. Then the line integration is carried out starting a "virtual particle" from this point $P_1$, into the computational area, until the next intersection $P_2$ with any non transparent additional or non default standard surface is found.

Note 1:
When specifying the coordinates of the points $P_{pivot}$ and $P_{scnd}$ to select a particular line of sight, the same rules must be followed as also for specifying birth point locations of "real" particles (input block 7). These starting points are assumed to be given **inside a cell, not exactly on a cell boundary**. For example coordinates z0=0.0 must be avoided if a z-grid surface coincides with this position. This latter is often the case e.g. already when default options for toroidal curvature (NLTRA option in block 2c) are used, even in otherwise only 1D (radial, x) or only 2D (radial and poloidal, x,y) resolved cases. The line of side points then must be specified at least with a small incremental distance to such coordinate surfaces, e.g. z0=1.0E-5 in the example.
Note 2:
The line integration (virtual particle flight) stops at the first non-transparent surface encountered along the line of sight. In cases when only the boundary plasma domain is simulated, the interface to the core plasma region is often such a "non-transparent surface" (mostly set to be purely absorbing). To allow line of sight **integration across this central core region** e.g. to

capture both top and bottom machine contributions along a line of sight, the following work-around can be used:
a) set the core boundary transparent, and switch (ILSWCH option block 3) into an unused additional cell, e.g. additional cell no. NC.
b) define a fixed "virtual core plasma" in this additional cell no. NC, e.g. by the options in input block 8, such that during "real" particle simulations the test particles get either absorbed in the core of reflected (by scattering) back into the boundary plasma domain.
For example in EMC3-EIRENE runs high plasma density and temperature are used in virtual inner core cells, which hence to not contribute significantly and artificially to line of sight integrals of atomic or molecular emission lines, which cross that core region.

The next card specifies the first ("pivot") point.

**IPIVOT** (only needed for NLTRA option, "toroidal approximation", sub-block 2c)

**1 ≤ IPIVOT ≤ NTTRA-1** (currently no available, error exit)

number of local toroidal co-ordinate system (NTTRA: see sub-block 2c), in which this pivot point is specified. The pivot point is then given in Cartesian coordinates is this local system

**IPIVOT = 0**

pivot point is given in global cylindrical coordinates,

XPIVOT,YPIVOT,ZPIVOT = $r, z, \phi$, with $\phi$ in degrees. The corresponding toroidal block numbers ITTRA are found automatically.

**XPIVOT** 1st co-ordinate of pivot point for line of sight, e.g. $x$

**YPIVOT** 2nd co-ordinate of pivot point for line of sight, e.g. $y$

**ZPIVOT** 3rd co-ordinate of pivot point for line of sight, e.g. $z$

The next card specifies the second ("inside") point.
The second point on the line of sight must either be inside the first (radial or x-) mesh, or the additional cell number of the mesh cell, to which this point belongs, must be specified by the NSPBLC and NSPADD flags.
(For otherwise the initial point for the line integration cannot be found automatically.)

**ICHORD** (only needed for NLTRA option, sub-block 2c)

Meaning analogous to that of first point flag IPIVOT:

**XCHORD** 1st co-ordinate of second point for line of sight

**YCHORD** 2nd co-ordinate of second point for line of sight

**ZCHORD** 3rd co-ordinate of second point for line of sight

**PLCHOR** the lines of sight are plotted into (2D or 3D) geometry plots. This, however, is automatically turned off if other plots are done between geometry plots (initialization phase) and line-of-sight integration (post processing phase).

**PLSPEC** the energy or wavelength resolved spectra, integrated along the lines of sight, are plotted (irrelevant in case of NSPTAL = 2).

**PRSPEC** the energy or wavelength resolved spectra, integrated along the lines of sight, are printed (from subr. OUTSIG.f) (irrelevant in case of NSPTAL = 2).

**PLARGL** the line of sight spectra, with their contributions spatially resolved along the lines of sight, are plotted.

**PRARGL** the line of sight spectra, with their contributions spatially resolved along the lines of sight, are printed. (directly from subr. LININT.f)

### 2.12.1 Line of sight: charge-exchange spectrum

### 2.12.2 Line of sight: line emissivity

### 2.12.3 Line of sight: line shape

### 2.12.4 Line of sight: user defined integral

## 2.13 Data for nonlinear and time dependent Options

### General remarks

The data in this block are used to define a discretization in time, and the test-particle self interaction effects. These latter options are based upon the "Bird's **DMCS!**" procedure, and are currently not in use (see [**kn:Behringer**] for a detailed description of its implementation in EIRENE). They can be activated by replacing the current dummy routine STOSS, which is called after each time-step from subroutine EIRENE, by a routine that reads the particle population from the "census arrays" described below, and which then carries out the binary self-collisions (modification of the individual particle velocity vectors). The rest of the code for time dependency and the **DMCS!** algorithm is the same and described in this section. Note that, currently, with the use of a dummy routine STOSS, non-linear self collision effects can still be simulated, in **BGK!** approximation, by using the iterative mode of operation (NITERI, NITERE, input block 1), see section 1.9.

As for the time-dependence options, it is important to note that there are two kinds of time-steps. One is the so called "time-cycle". Each time cycle is a more or less complete EIRENE run in its own, even if several (NTIME, see section 2.1) such time-cycles are carried out in one single job. In each time-cycle, a time-horizon is set and the trajectories are stopped (at latest) if they have survived until then. All relevant particle coordinates (position, velocity, type and species, cell indices, etc.) are stored at this time-horizon on the so called "census-arrays".

Each time cycle may consist of one or more than one (NTMSTP, see below) "internal time steps". At the end of each of these time-steps, the relevant snapshot tallies are updated (subroutine TIMCOL). These are the volumetric "snapshot tallies", the default "time-surface tallies" (particle and energy fluxes) and the "census arrays". Hence: these tallies are averaged over NTMSTP (internal) time-steps. For more details on snapshot tallies see section 3.2.3.

This distinction between complete "time cycles" and "internal time steps" is particularly relevant for obtaining truly steady state results on the census arrays, e.g., in order to continue a stationary case in the time-dependent mode (see below).

### The Input Block

```
*** 13. Data for nonlinear and time dependent Options
      NPRNLI NINITL_READ NPRMUL
      IF (NLERG.AND.NPRNLI.EQ.0) NPRNLI=100
      IF (NPRNLI.GT.0) THEN
        NPTST,NTMSTP
        DTIMV,TIME0
** 13A. DATA FOR SNAPSHOT TALLIES
        NSNVI
        IF (NSNVI.GT.NSNV)
          DO 1320 J=1,NSNVI
            ISNVE(J),ISNVS(J),ISNVT(J),ISNRC(J)
            TXTTAL(J,NTALT)
            TXTSPC(J,NTALT),TXTUNT(J,NTALT)
 1320     CONTINUE
        ENDIF
```

**Meaning of the Input Variables for this block**

**NPRNLI** Total number of test particles in time dependent arrays ("census arrays") (in old versions before 2001: NPRNLI must be ≤ NPRNL, see: PARMUSR). The scoring on census arrays stops at latest when NPRNLI scores are on the census array.

If NPRNLI > 0, but the rest of the data in this block are not specified, then a default time-horizon is defined. See default values specified below.

If NPRNLI = 0, no census arrays are scored

In case NLERG = TRUE (see input block 1), a time horizon is absolutely necessary in order to prevent infinite histories. Therefore, in case NLERG=TRUE and NPRNLI=0 an automatic correction to NPRNLI=100 is carried out.

**NINITL_READ** (new: 2013) Same as NINITL in block 7: provides random number seed for "time-stratum" (sampling from census array (default: =0, no fresh initialization of random number generator for this stratum)

**NPRMUL** (new: 2013) Multiplicative factor for NPRNLI, in order to increase size of census to more than 999999 particles, which otherwise would be the maximum due to I6 formatted integer input. Default: = 0: no multiplication carried out

**NPTST** Same as NPTS in block 7. This is the number of histories, which are continued from a previous time-cycle ("Time dependence stratum ISTRA=NSTRAI+1"). The initial coordinates are randomly sampled (with replacement) from the census-array data from an earlier time-cycle. The probability for sampling a particular particle from the census array is proportional to its weight stored on the census array as well. Due to "sampling with replacement" an individual particle, which is on census, may be sampled more than once, or not at all, with the likelihood for these events given by its weight ("warm restart"). This census array is either defined at the end of the previous time cycle in the same run (subr. TMSTEP), or it is read from an earlier run from stream 15 (via a call to subr. RSNAP from subr. INPUT) in the initial phase of the run, for the very first time-cycle (continuation of an earlier sequence of time-cycles).

If NPTST = 0, then NPTST is reset to IPRNL. IPRNL is the the number of scores on the census array in the previous time cycle.

If NPTST < 0, then NPTST is reset to IPRNL, and the random sampling from the census array is now replaced by a one-to-one re-launch of all particles from the census array without random sampling ("cold restart"). Until Aug. 2015 this option was available only in connection with the NLMOVIE option (movies of trajectories) and had led to other modifications of the run parameters as well. (Automatically then internally: NLMOVIE = TRUE). New: NLMOVIE AND NPTST < 0 options are now independent from each other. In both cases: one by one re-launch from old census is enforced, rather than random sampling from old census.

Default: NPTST=0

**NTMSTP** Total number of time-steps for particle tracing. Each trajectory can score on census up to NTMSTP times. Particle trajectories are stopped after NTMSTP time-steps.

Default: NTMSTP=1

For convenience and by abuse of language, we refer to the 3-dimensional hyper-surface $t = t_n$ of the four dimensional $(\mathbf{r}, t)$-space as "time-surface", and, hence, tallies evaluated at fixed time $t_n$ ("snapshot-tallies") are surface averaged tallies in this terminology.

Fluxes onto this surface are stored on the arrays for a surface no. NLIM+NSTSI+1, which is added automatically to the NLIM additional and NSTSI non-default standard surfaces.

The time-surface is transparent for NTMSTP-1 steps and absorbing afterwards, i.e., absorbing at time t = NTMSTP × DTIMV. Snapshot tallies are averages over NTMSTP time-steps.

Default: NTMSTP=1

If NTMSTP < 0, then each particle can score an unlimited number of times on census, i.e., the time-surface is always transparent. This option can be used for initialization of census arrays for time dependent runs. The census arrays then represent a stationary distribution corresponding to a certain constant (in time) influx of particles, rather than an estimate at a fixed time. Hence, for any fixed detector function (volume averaged tally), the snapshot estimator should give the same results (up to statistical precision) as the track-length estimator or the collision estimator. Note that in order to obtain this stationary estimate from snapshot estimates an additional multiplicative factor DTIMV (s) (see next) is applied to snapshot tallies in case NTMSTP < 0.

**DTIMV** Length of each individual internal time-step (s)

Default: DTIMV=1.D-02

**TIME0** Initial time $t_0$ of the first time-step. (irrelevant, only for printout and book-keeping)

Default: TIME0=0.

**NSNVI** Number of snapshot tallies computed from census arrays. In old EIRENE versions without dynamic allocation of storage NSNVI must be less or equal NSNV (see PARMUSR), and the detector functions are user supplied in subroutine UPNUSR, see Sub-section 3.2.3.

Default: NSNVI=0

The meaning of the next three cards is the same as for the corresponding cards in block 10A, 10B or 10D.

**Census Arrays**
All particle co-ordinates at time $t_i$ for history number IHIST, i.e., position, cell indices, velocity etc. are stored on arrays RPART(IHIST, ...) (Real) and IPART(IHIST, ...) (Integer) in subroutine TIMCOL. This subroutine TIMCOL is called if "a collision with the time surface $t = t_i$" has occurred.
The number of scores on the old census arrays (from a previous completed time-cycle in a current run or from an earlier run) is **IPRNL**.
The actual number of a score in subroutine TIMCOL is **IPRNLI**, and after the score IPRNLI is increased by one. IPRNLI is set to zero at the beginning of a run.

The actual number of a score in subroutine TIMCOL for the present stratum ISTRA is **IPRNLS**, and after the score IPRNLS is increased by one. IPRNLS is reset to zero at the beginning of sampling for each stratum.
Scaling of census array fluxes: to be written

## 2.14 Data for interfacing Subroutine “INFCOP” (example)

**General remarks**

Data in input block 14 control additional input for an EIRENE run.
In case NMODE = 0 (see input block 1), i.e., no call to interfacing subroutine INFCOP, only the additional input tallies ADIN (see table 6.2), if any, are specified here.
Otherwise, if NMODE ≠ 0 the data in this block are read from the code interfacing subroutine INFCOP (at entry IF0COP) rather than from subroutine INPUT. They may be used to modify or complete the model defined by the formatted input file so far. For example, by this option the entire geometry specification (blocks 2,3) can be modified or overwritten by a few geometry parameters without rewriting input blocks 2 and 3. This allows rapid geometry optimization (geometry parameter studies), which otherwise would require to generate a large set of different geometry input blocks. As this routine is problem specific, it must be written by the user and, therefore, input can be from any file and in any format chosen there.
Subroutine INFCOP is also used for interfacing with other codes, such as plasma transport codes. By use of the flags INDGRD (block 2), INDPRO (block 5) and INDSRC (block 7) data may be transferred from subroutine INFCOP into EIRENE geometry arrays, background medium arrays and source distribution arrays, respectively, in a preprogrammed format. This is described in chapter 4 in general terms. Here we give examples for one frequently used option, namely the coupling of EIRENE to the 2 dimensional plasma transport code B2 [**kn:Braams**] (the “B2-EIRENE” code system, [**kn:Reiter91b**],[**kn:Reiter92b**]). A corresponding version of INFCOP is available from FZ Jülich.

**The Input Block (in “stand alone” case NMODE = 0)**

```
***14. Data for additional input tallies, in stand alone mode
      IF (NMODE.EQ. 0) THEN

        NAINI, ...
        DO IAIN=1,NAINI
          NAINS(IAIN) NAINT(IAIN)
          TXTPLS(IAIN,11)
          TXTPSP(IAIN,11) TXTPUN(IAIN,11)
        ENDDO
      ELSEIF (NMODE.NE. 0) ... see paragraphs below
```

**Meaning of the Input Variables of block 14 in stand alone mode**

**NAINI** Number of additional input tallies (module COMUSR, printout: OUTPLA). Storage is provided for NAINI additional input tallies. These tallies may be filled e.g. in the initialization phase of a run, e.g. calls to PLAUSR, PROUSR, etc.

**NAINS** species index of additional input tally (case specific, some special preprogrammed cases: see below)

**NAINT** number index of additional input tally (case specific, some special preprogrammed cases: see below)

**TXTPLS(IAIN,11),TXTPSP(IAIN,11) TXTPUN(IAIN,11)** text (name of tally), species and units used for printout and plotting of this tally

Currently the following options have been preprogrammed:
$20 \leq$ NAINT $\leq 29$ :
normalized EIRENE atomic/molecular data profiles (rate coefficients in atomic units), as evaluated on computational grid, using input atomic and molecular data. These can be selected also in the NMODE=0 (subroutine INFCOP not called) option. These options allow verification of selected atomic/molecular data on the computational grid after evaluating the fits or data tables, using the plasma background data of a particular run. This option is in particular relevant for checking for extrapolation errors in A&M data fits or tables)

**NAINT**

= 20 electron impact collision rate coefficient $\langle \sigma v \rangle_{ei}$ [a.u.], with the rate coefficient profile taken from TABEI3(IREI, ...) arrays. $IREI : 1 \leq NREI$ : number of electron impact process. See TRCAMD output from the initialization phase of a run to identify, which particular electron impact process IREI is assigned to which species IATM, IMOL, etc.

= 21 electron cooling rate coefficient, EELEI3 array (not ready) [eV · (rate-coeff in a.u.)]

= 22 charge-exchange rate coefficient, TABCX3 arrays

= 23 charge-exchange ion energy weighted rate coefficient, EPLCX3 arrays (not ready) [eV · (rate-coeff in a.u.)]

= 24 elastic collision rate coefficient, TABEL3 arrays

= 25 elastic collision ion energy weighted rate coefficient, EPLEL3 arrays (not ready) [eV · (rate-coeff in a.u.)]

= 26 general ion impact collision rate coefficient, TABPI3 arrays

= 27 general ion impact collision ion energy weighted rate coefficient, EPLPI3 arrays (not ready) [eV · (rate-coeff in a.u.)]

= 28 recombination rate coefficient $\langle < \sigma v \rangle_{rc}$ [a.u.], with the rate coefficient profile taken from TABRC1(IREC, ...) arrays. $IREC : 1 \leq NREC$ : number of recombination process. See TRCAMD output to identify, which recombination process is assigned to which species IPLS.

= 29 electron energy loss weighted recombination rate coefficient [eV · (rate-coeff in a.u.)]

### 2.14.1 Version B2-EIRENE-1999 and older

B2 solves a set of continuum equations for electrons and ions, and EIRENE solves a set of kinetic transport equations for other species, not in the continuum description (e.g. neutrals, radiation, trace ions, ...) In this hybrid fluid-kinetic formulation the EIRENE code provides volumetric particle, parallel momentum and electron and ion energy sources $S_n, S_{m\|}, S_{Ei}, S_{Ee}$, respectively, from those "kinetic species", for the B2 continuum equations for the orthogonal set of "species in fluid description". Formally, these volumetric source rates arise in the particle, parallel

momentum and energy balances. For the B2 code these balances are discretized in conservative form (finite volume (FV) schemes) and solved (in the laboratory frame of reference):

$$\frac{D(n_{ipls})}{Dt} = S_n(ipls) + o.c. \quad ipls = 1, \ldots, NFLA \tag{2.17}$$

$$\frac{D(m_{ipls}\, n_{ipls}\, V_{\parallel,ipls})}{Dt} = S_{m\parallel}(ipls) + o.c. \quad ipls = 1, \ldots, NFLA \tag{2.18}$$

$$\frac{D(\frac{3}{2}\, n\, T_i + \sum_{ipls} \frac{m_{ipls}}{2} V^2_{\parallel,ipls})}{Dt} = S_{Ei} + o.c. \tag{2.19}$$

$$\frac{D(\frac{3}{2}\, n\, T_e)}{Dt} = S_{Ee} + o.c. \tag{2.20}$$

$D/Dt$ denotes the derivative (total, or convective derivative, or partial derivative), and $o.c.$ stands for other contributions such as e.g. friction forces, pressure gradient forces, collisional heating, etc.
All source rates $S$ are resolved per stratum $is$ and per cell of the computational grid $icell$. The particle and parallel momentum sources are also resolved with respect to receiving plasma fluid ion species $ipls$. The source rates are given as linear functions of default EIRENE tallies. Statistical error estimates (the standard deviations) of all sources transferred to B2 are evaluated in case specific routine STATS1_COP. All printout of these sources and their statistical errors is case specific and done in interfacing module EIRCOP.
**Particle source: [particles/time/volume], amp cm$^{-3}$**

$$S_n(ipls, icell) = \sum_{is} [PAPL_{is}(ipls, icell) + PMPL_{is}(ipls, icell) + PIPL_{is}(ipls, icell)] \tag{2.21}$$

The sum is over all strata (see "stratified sampling" in chapter 1). PAPL, PMPL, PIPL are default EIRENE tallies (table 6.7 in section 6.1.2):

- PAPL is the particle source for background plasma species $ipls$ resulting from "atom"-plasma interactions (summed over all EIRENE species of type "atom"), tally no. 11.
- PMPL is the particle source for background plasma species $ipls$ resulting from "molecule"-plasma interactions (summed over all EIRENE species of type "molecule"), tally no. 16.
- PIPL is the particle source for background plasma species $ipls$ resulting from "test ion"-plasma interactions (summed over all EIRENE species of type "test ion"), tally no. 21.

**Momentum source: [mass velocity/time/volume], g cm s$^{-1}$ amp cm$^{-3}$**

$$S_{m\parallel}(ipls, icell) = \sum_{is,icop} [COPV_{is}(icop, icell)] \tag{2.22}$$

The sum is over all strata (see "stratified sampling" in chapter 1). $COPV$ is the problem specific tally for code interfacing, tally no. 40, and it is scored in the routine UPDCOP (different versions exist for different plasma fluid codes), table 6.7 in section 6.1.2):
The sign of these sources in EIRENE is such that a gain in momentum for plasma species $ipls$ is taken positive. In the interfacing routine IF3COP of module EIRCOP the sign convention

is altered to that of the plasma fluid code, i.e. it then involves the flow direction relative to the magnetic field vector. Note that in more recent versions of EIRENE, the momentum sources parallel to a magnetic field have become default tallies, see below in section 2.14.2.
**Ion energy source: [Energy/time/Volume], watt cm$^{-3}$** (for total ion energy balance, in laboratory frame)

$$S_{Ei}(icell) = \sum_{is} [EAPL_{is}(icell) + EMPL_{is}(icell) + EIPL_{is}(icell)] \tag{2.23}$$

The sum is over all strata (see "stratified sampling" in chapter 1). EAPL, EMPL, EIPL are default EIRENE tallies (table 6.7 in section 6.1.2):

- EAPL is the ion energy source resulting from "atom"-plasma interactions (summed over all EIRENE species of type "atom"), tally no. 26.
- EMPL is the ion energy source resulting from "molecule"-plasma interactions (summed over all EIRENE species of type "molecule"), tally no. 31.
- EIPL is the ion energy source resulting from "test ion"-plasma interactions (summed over all EIRENE species of type "test ion"), tally no. 36.

**Electron energy source: [Energy/time/volume], watt cm$^{-3}$**

$$S_{Ee}(icell) = \sum_{is} [EAEL_{is}(icell) + EMEL_{is}(icell) + EIEL_{is}(icell)] \tag{2.24}$$

The sum is over all strata (see "stratified sampling" in chapter 1). EAEL, EMEL, EIEL are default EIRENE tallies (table 6.7 in section 6.1.2):

- EAEL is the electron energy source resulting from "atom"-plasma interactions (summed over all EIRENE species of type "atom"), tally no. 22.
- EMEL is the electron energy source resulting from "molecule"-plasma interactions (summed over all EIRENE species of type "molecule"), tally no. 27.
- EIEL is the electron energy source resulting from "test ion"-plasma interactions (summed over all EIRENE species of type "test ion"), tally no. 32.

The data transferred from B2 into EIRENE are either from Common BRAEIR, (a module common to both B2 and EIRENE) or from the stream FORT.31. Due to the "staggered grid" discretization in the B2 code, there as cell centred as well as surface centred quantities. In the 2D r-z plane (poloidal cross section of a torus) there are so called "x-surfaces", running from west to east, which are perpendicular to the poloidal magnetic field (or flux surfaces), and "y-surfaces", running from south to north, which are aligned with the poloidal magnetic field direction. They are read in the following sequence,

= 1 DI :plasma ion density [m$^{-3}$], cell centered

= 2 UU :poloidal velocity [m s$^{-1}$], x-surface centered

= 3 VV :radial velocity [m s$^{-1}$], y-surface centered

= 4 TE :electron temperature [J], cell centered

= 5 TI :ion temperature [J], cell centered

= 6 PR :plasma pressure [N $m^{-2}$], cell centered

= 7 UP :parallel velocity [m $s^{-1}$], x-surface centered

= 8 RR :pitch angle [1], cell centered

= 9 FNIX: Particle fluxes along the field [$s^{-1}$], x-surface centered

= 10 FNIY Particle fluxes across the field [$s^{-1}$], y-surface centered

= 11 FEIX Ion energy fluxes along the field [watt], x-surface centered

= 12 FEIY Ion energy fluxes across the field [watt], y-surface centered

= 13 FEEX Electron energy fluxes along the field [watt], x-surface centered

= 14 FEEY Electron energy fluxes across the field [watt], y-surface centered

= 15 VOL: cell volume [$m^3$], cell centered

= 16 BFELD: magnitude of magnetic field [T], cell centered

Thereafter index mapping (accounting for grid cuts), and units conversion are carried out, to turn these fields into EIRENE bulk ion tallies (table 6.6) as well as to derive recycling target fluxes and boundary conditions from them.

**The Input Block**

```
***14. Data for interfacing Subroutine "INFCOP"
      IF (NMODE.EQ. 0) THEN
...
...
      ELSEIF (NMODE.NE. 0) THEN

        LSYMET LBALAN LCHKQUD
        NFLA NCUTB NCUTL MSHFRM NTRFRM NFULL IBRAD IBPOL IBTOR
        DO 20 IPLS=1,NPLSI
          I IFLB(IPLS) FCTE(IPLS) BMASS(IPLS)
        20 CONTINUE
        NDXA NDYA
        NTARGI
        (NTGPRT(IT),IT=1,NTARGI)
        DO 30 IT=1,NTARGI
          DO 33 IPRT=1,NTGPRT(IT)
            I NDT(IT,IPRT) NINCT(..,...) NIXY(..,...)
.             NTIN(..,...) NTEN(..,...)
            NIFLG(..,...) NPTC(..,...) NSPZI(..,...) NSPZE(..,...)
```

```
   33 CONTINUE
 30 CONTINUE
CHGP CHGEE CHGEI CHGMOM
NAINB
DO 40 IAIN=1,NAINB
   I NAINS(IAIN) NAINT(IAIN)
   TXTPLS(IAIN,11)
   TXTPSP(IAIN,11) TXTPUN(IAIN,11)
 40 CONTINUE
NAOTB
DO 50 IAOT=1,NAOTB
   I NAOTS(IAOT) NAOTT(IAOT)
 50 CONTINUE
```

**Meaning of the Input Variables for interfacing Subroutine "INFCOP"**

**LSYMET =.TRUE.** Upside-down symmetry of all tallies transferred to external code via common block EIRBRA) is enforced.

Symmetry plane is the PSURF/2. surface, i.e., the poloidal (or y) co-ordinate surface PSURF(NP2ND/2) at the center of the computational interval in this co-ordinate.

This option has historical reasons. The very first B2-EIRENE runs ever have been performed for "upside-down symmetric" ITER double null configurations (Reiter et al., 1991, [**kn:Reiter91b**]). In more recent versions this option may not be available anymore.

**=.FALSE.** no such symmetry is enforced

**LBALAN =.TRUE.** Global particle and energy flux balance is performed and printed at the end of an EIRENE stand alone run (steady state or single time-step) and at the end of a short cycle between a plasma transport code and EIRENE. These balances compare plasma particle and energy fluxes at the boundaries, volume sources in plasma balance equations and neutral-plasma interaction sources and sinks.

**=.FALSE.** no such balances are computed

**NFLA** Number of different (bulk) ion species in plasma code.

**NCUTB** Number of mesh cells in each grid cut in plasma code (specific to grid generator used in connection with B2 fluid code).

**NCUTL** Number of mesh cells in each grid cut in EIRENE

**Note:** if NCUT ≠ NCUTL, the index mapping routines INDMAP and INDMPI are called at each call to the interfacing routines.

**MSHFRM, NTRFRM, NFULL** new flags, available only since 2013, see section 2.14.3

**IBRAD, IBPOL, IBTOR** new flags for magnetic field orientation, available only since March 2015, see section 2.14.3

**I** irrelevant, labelling index for EIRENE bulk ion species IPLS, runs from 1 to NPLS

**IFLB** $> 0$ labelling index of bulk ion species in plasma code. EIRENE bulk ion species IPLS corresponds to plasma code species IFLB(IPLS). Hence: IFLB $\leq$ NFLA, otherwise: error exit.

The drift velocity vector for all EIRENE species IPLS is set identical to the drift velocity of the plasma code species IFLB(IPLS).

Only one common ion temperature is available from B2, B2.5 code runs, even for multi-species applications. The bulk ion temperature TIIN(IPLS, . . . ) for all EIRENE species IPLS, which are read from the plasma code data files, i.e., for all species with IFLB(IPLS) $> 0$

$< 0$ The plasma density, flow field and temperature field for EIRENE species IPLS is read from the input stream fort.II, with II=-IFLB. (Currently only II=13). This file fort.13 must be available, e.g., from a previous EIRENE run, in which NFILEL = 1 or NFILEL = 3 options have been used to produce that file. (see input block 1). This option permits iteration on some species, which are not treated in the plasma code, e.g.: for neutral–neutral interactions.

$= 0$ The corresponding bulk ion species IPLS in EIRENE has zero density, i.e. bulk ions of this species are not present in this EIRENE run. Note the issue re. electron density and quasi-neutrality mentioned above.

**FCTE** bulk ion density (and flux) multiplication factor. The EIRENE bulk ion density (and fluxes) for species IPLS are obtained by multiplying the corresponding plasma code profiles for species IFLB(IPLS) with the factor FCTE. This option is needed, e.g., if the plasma code treats one ion species of mass 2.5 AMU, while EIRENE treats D ions and T ions separately.

Note: in an iterative mode all plasma species in the plasma code should also be in EIRENE. There may be more in EIRENE, with charge state zero (see introduction to section 2.4), but not less, because otherwise the electron density computed in EIRENE from quasi-neutrality (and used, e.g., for ionization mean free paths), may be inconsistent with the electron density in the plasma code.

**BMASS** Mass of plasma code species IFLB(IPLS) is BMASS(IPLS) in AMU.

**NDXA,NDYA**

grid size in 2D plasma code. NDXA is the size of the grid tangential to the flux surfaces, and NDYA is the size of the grid normal to the flux surfaces.

These grid sizes have to be equal to the grid sizes ($1^{st}$ and $2^{nd}$ grid RSURF and PSURF) of the mesh, on which plasma code data are specified for EIRENE.

I.e., NDYA=NR1ST-1 and NDXA=NP2ND-1 must be obeyed.

**NTARGI** Number of different surface recycling sources defined from the plasma code surface effluxes at specified boundaries.

**NTGPRT** Number of different surface segments (radially or poloidally), by which this target recycling stratum no. ITARGI is composed.

The following NTGPRT(ITARG) cards are used to specify these selected recycling boundaries and to set up the appropriate EIRENE surface source flags. These flags partially overrule those specified in block 7 (section 2.7) (surface sources), see below. If even INDSRC = 6 for that particular stratum in block 7 (section 2.7), then no further data must be read for this stratum in input block 7 (section 2.7) and the full specification of that recycling source is done in subroutine INFCOP automatically from the plasma code data and the next NTGPRT input cards for this stratum.

**IT** irrelevant, surface source labelling index

**NDT** number of surface in plasma code mesh, either in tangential or in normal direction. If the plasma code uses cell centered indexing, then the north and the east surfaces of a cell are labeled by the indices of the cell. In EIRENE cell indexing, the south and the west surface have the same index as the corresponding cell in the first (radial) and second (poloidal) mesh, respectively. Therefore NDT may be different from the surface labelling index in EIRENE input block 3a (section 2.3.1). Note also that NDT refers to cell numbers after index mapping, if NCUTL is not equal to NCUTB.

**NINCT** = 1 positive (i.e., outer) surface normal is in the positive co-ordinate direction (as it is the case by default for EIRENE standard co-ordinate surfaces).

= −1 positive surfaces normal is in the negative co-ordinate direction.

**NIXY** = 1 surface in the direction normal to the flux surface, i.e., it belongs to the 2$^{nd}$ EIRENE mesh PSURF (usually: divertor target)

= 2 surface is in the tangential direction, i.e., it belongs to the 1$^{st}$ EIRENE mesh RSURF (usually: vessel, liner, interface to vacuum region)

**NTIN,NTEN**

The surface source is restricted to the cells ranging from cell number NTIN to cell number NTEN-1, along the co-ordinate surface.

Note that NTIN and NTEN label cell boundaries, hence the NTEN-1 above.

Note further: the total number of surface cells in one surface recycling source ITARG (i.e., summed over NTGPRT(ITARG)) must not be larger then NR1ST+NP2ND (see definition of parameter NGITT in Common Deck PARMMOD).

**NIFLG** This corresponds to the SORIFL flag in input block 7, and is needed only if INDSRC=6, i.e., if the source is specified by data from block 14 alone.

**NPTC** This corresponds to the NPTS flag in input block 7, and is needed only if INDSRC=6, i.e., if the source is specified by data from block 14 alone.

**NSPZI,NSPZE**

Species range for this stratum. Only the EIRENE fluids IPLS corresponding to plasma code fluids IFL with NSPZI < IFL < NSPZE are sampled from this stratum. See index map ILFB(IPLS) specified above in this same input block.

One (geometrical) target may appear several times, with different species ranges. This permits to apply stratified sampling within the species distribution, and hence to remove statistical noise resulting from species source sampling.

The following automatic adjustments to input flags from block 7 are then carried out for strata ISTRA = 1, NTARGI (species and birth point sampling)

- The source strength FLUX(ISTRA) is reset to the value as computed from the plasma code data (fluxes) at the corresponding surface and the species range specified.
- Initial species distribution is sampled from bulk ion population: NLPLS(ISTRA) = TRUE
- Stratum ISTRA is a surface source: NLSRF(ISTRA) = TRUE
- Number of sub-strata NSRFSI(ISTRA)=1
- Mixed radial/poloidal (or: x/y) surface type: INDIM(ISTRA,1)=4
- Step-function sampling with SORLIM(ISTRA, 1) = 104, and the step function data FLSTEP, ... for the flux distribution entering the sheath region of the target are automatically defined.
- Number of step-function SORIND(ISTRA,1)=ITARG (=ISTRA)

The other input flags for birth-point sampling in input block 7 have then become irrelevant by this new settings of the modified parameters listed above.
The sampling of the velocity of the incident ions is as specified in input block 7 for that stratum.

**CHGP** The short cycle between EIRENE and the plasma code (i.e., only recomputing of source term profiles in subroutine EIRSRT at each time-step, but no new random walks) is stopped, if the total volume integrated particle source rate has changed by more than CHGP per cent as compared to the previous full EIRENE run.

**CHGEE** as above, but for total electron energy source rate.

**CHGEI** as above, but for total ion energy source rate.

**CHGMOM** as above, but for total ion parallel momentum source rate. (to be written, May 1995)

**NAINB** total number (=NAINI) of additional plasma code tallies transferred onto EIRENE input tally ADIN (e.g. in order to utilize EIRENE output facilities for plasma code data, or for the options described in sub block 10c.

**I** irrelevant, labelling index

**NAINS** species index of tally in B2 arrays.

**NAINT** flag to determine which particular quantity is put onto input tally ADIN(I, . . . ).

In the current version of subroutine INFCOP for interfacing to B2 the following "B2-quantities" can be selected.

$1 \leq NAINT \leq 16$ :

= 1 DI :plasma ion density [$m^{-3}$], also on DIIN, by IPLS species

= 2 UU :poloidal velocity [$m\,s^{-1}$]

= 3 VV :radial velocity [$m\,s^{-1}$]

= 6 PR :plasma pressure [$N\,m^{-2}$]

= 7 UP :parallel velocity [$m\,s^{-1}$]

= 8 RR :pitch angle [1]

= 9 FNIX: Particle fluxes along the field [$s^{-1}$]

= 10 FNIY Particle fluxes across the field [$s^{-1}$]

= 11 FEIX Ion energy fluxes along the field [watt]

= 12 FEIY Ion energy fluxes across the field [watt]

= 13 FEEX Electron energy fluxes along the field [watt]

= 14 FEEY Electron energy fluxes across the field [watt]

= 15 VOL: cell volume [$m^3$]

= 16 BFELD: magnitude of magnetic field [T]

$21 \leq NAINT \leq 30$ : normalized EIRENE atomic/molecular data profiles (rate coefficients in atomic units). These can be selected also in the NMODE=0 (subroutine INFCOP not called) option, see previous paragraph for their description. These options allow verification of selected atomic/molecular data on the computational grid and evaluated using the plasma background data of a particular run (in particular: check for extrapolation errors in A&M data)

**TXTPLS, TXTPSP, TXTPUN**

text for printout and plotting, same as described for additional output tallies in bock 10.

**NAOTB** total number of additional EIRENE surface tallies transferred to B2 code (e.g. in order to allow re-scaling in B2 such that total number of particles (neutrals and ions) is conserved.

**NAOTS** not in use

**NAOTT** not in use

### 2.14.2 Version B2-EIRENE-2000 and younger

The first thing to note is that due to introduction of photons as a forth type of test particle (for radiation transfer simulations) the numbering of tallies (source rates) to be transferred from EIRENE into B2 has changed. For example the particle source rates PAPL, PMPL, PIPL are now default EIRENE tallies listed in table 6.3 in section 6.1.1 rather than in table 6.7 in section 6.1.2:

- PAPL is the particle source for background plasma species $ipls$ resulting from "atom"-plasma interactions (summed over all EIRENE species of type "atom"), tally no. 14.
- PMPL is the particle source for background plasma species $ipls$ resulting from "molecule"-plasma interactions (summed over all EIRENE species of type "molecule"), tally no. 20.
- PIPL is the particle source for background plasma species $ipls$ resulting from "test ion"-plasma interactions (summed over all EIRENE species of type "test ion"), tally no. 26.

Default tally no. 32 might be added here as well, once photo-ionization processes are activated in an EIRENE run. Analogous statements, i.e. modifications of volume averaged tally numbers, apply for the other source rates (for electron and ion energy balance), as compared to those described above in section 2.14.1.
One further difference is that momentum source terms (for parallel momentum balance equations of plasma fluid codes) have become default tallies (MAPL, MMPL, MIPL), rather than problem specific tallies (COPV). These are now scored in routine UPDATE, together will all other tallies listed in table 6.3, rather than in the coupling-specific routine UPTCOP. I.e. now we have (compare to (2.22)):

$$S_{m\|}(ipls, icell) = \sum_{is} \left[ MAPL_{is}(ipls, icell) + MMPL_{is}(ipls, icell) + MIPL_{is}(ipls, icell) \right] \tag{2.25}$$

The sum is over all strata (see "stratified sampling" in chapter 1). MAPL, MMPL, MIPL are default EIRENE tallies (table 6.3 in section 6.1.1):

- MAPL is the parallel momentum source for background plasma species $ipls$ resulting from "atom"-plasma interactions (summed over all EIRENE species of type "atom"), tally no. 97.
- MMPL is the parallel momentum source for background plasma species $ipls$ resulting from "molecule"-plasma interactions (summed over all EIRENE species of type "molecule"), tally no. 98.
- MIPL is the parallel momentum source for background plasma species $ipls$ resulting from "test ion"-plasma interactions (summed over all EIRENE species of type "test ion"), tally no. 99.

The sign of these sources in EIRENE is such that a gain in momentum (speeding up the plasma flow) for plasma species $ipls$ is taken positive. In the interfacing routine IF3COP of module EIRCOP the sign convention is altered to that of the plasma fluid code, i.e. it then involves the flow direction relative to the magnetic field vector. Note that in all B2-EIRENE versions until March 2015 the poloidal unit vector is always taken to be given by the poloidal magnetic field component, i.e., the poloidal magnetic field component is always taken to point in the positive poloidal grid coordinate direction.
**Meaning of the Input Variables for interfacing Subroutine "INFCOP"**

input block 14: unchanged, as compared to section 2.14.1

Additional data to those described in previous sub-section 2.14.1 are read, basically to account for non-orthogonal grids and for direct transfer of surface recycling boundary conditions into EIRENE.
From input stream FORT.29 two angles are read for each cell:

= 1 ALPHXB: angle of B-field against R-coordinate (x-coordinate in EIRENE, [*radians*], evaluated at B2-x-surface (East, West) surface-centered

= 2 ALPHYB: angle of B field against R-coordinate (x-coordinate in EIRENE, [*radians*], evaluated at B2-y-surface (North, South) surface-centered

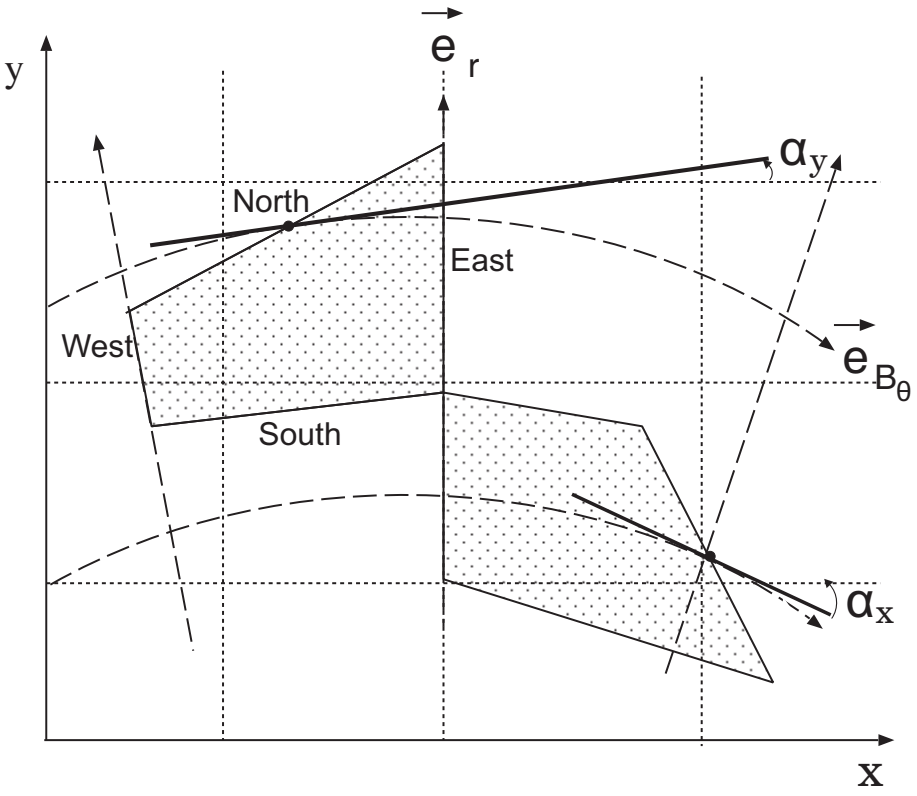



Figure 2.1: Definition of angles $\alpha_x$=ALPHXB and $\alpha_y$=ALPHYB, respectively, for inclined (non-orthogonal) B2-grids to be used in EIRENE code.

Then there are 14 arrays for definitions of boundary conditions (stream FORT.31 or Common BRAEIR), in order to achieve full symmetry of boundary condition options between east-west and north-south cell faces. This is in particular useful for cases with wide grids (up to the real vacuum vessel) in which boundary conditions on strongly inclined surfaces are defined.

**1** v_par,x: parallel velocity [m s$^{-1}$], x-surface (across field, East) centered

**2** v_par,y: parallel velocity [m s$^{-1}$], y-surface (along field, North) centered

**3** v_rad,x: plasma ion density [m s$^{-1}$], x-surface (across field, East) centered

**4** v_rad,y: poloidal velocity [m s$^{-1}$], y-surface (along field, North) centered

**5** $\delta_e$_par,x: x-surface (across field, East) centered

**6** $\delta_e$_par,y: y-surface (along field, North) centered

**7** $\delta_e$_rad,x: x-surface (across field, East) centered

**8** $\delta_e$_rad,y: y-surface (along field, North) centered

**9** $\delta_i$_par,x: x-surface (across field, East) centered

**10** $\delta_i$_par,y: y-surface (along field, North) centered

**11** $\delta_i$_rad,x: x-surface (across field, East) centered

**12** $\delta_i$_rad,y: y-surface (along field, North) centered

**13** sheath_x: x-surface (across field, East) centered

**14** sheath_y: y-surface (along field, North) centered

On any particular surface element only either the x or the y values are set, depending upon whether this surface is a x or y surface. One cell, however, may have either one or two recycling surfaces, one north- and one east surface.

### 2.14.3 Version B2-EIRENE-wide-grid (2011 and later)

Further adaptations in case of "wide grid" option, to accommodate mixed fluxes (e.g. west and north) at a single boundary element. Also additional: flags for magnetic field orientation, e.g. to allow poloidal and/or toroidal magnetic field components to be reversed, opposed to B2 radial and poloidal grid orientation.
To be written
Meaning of additional flags from input block 14 (format: see section 2.14):

**MSHFRM** three different options for reading of B2 grid:

MSHFRM = 0: "Linda" format, historically oldest format, default, e.g. used in SOLPS. Originates from LINDA grid generating code, G. Maddison, mid eighties last century.

MSHFRM = 1: "Sonnet" format

MSHFRM = 2: "Carre" format

**NTRFRM** two different formats for triangular grid files are supported:

NTRFRM = 0: old format, (first all x coordinates, then all y coordinates)

NTRFRM = 1: new format, (x,y coordinate per point, one point per input line)

**NFULL** Maximum number of short cycling steps, before a full EIRENE run is enforced for all strata

**Magnetic field conventions in B2, B2.5 interfaces**

The next three flags (introduced in March 2015) allow to change the orientation of the magnetic field as reconstructed form B2 or B2.5 parameters. The magnetic field itself is typically not used in coupled B2-EIRENE cases, however the "parallel momentum exchange" tallies transferred from EIRENE to B2 (MAPL, MMPL, MIPL, or in older versions, certain COPV tallies) are based on a projection of the vectorial momentum exchange rate onto the parallel to **B** component. Introducing the radial, poloidal and toroidal magnetic field components ($B_r, B_\theta, B_\phi$) the default (and exclusive, until March 2015) option **in all B2-EIRENE versions ever, until then**, was to assume that

$B_r = (\mathbf{B} \cdot \mathbf{e}_r)$

the radial magnetic field $B_r$ is zero: $B_r$ =0 always. But note that the B2, B2.5 radial grid index is typically IY, whereas the EIRENE x-or radial grid index is IX, IR.

$B_\theta = (\mathbf{B} \cdot \mathbf{e}_\theta)$

the poloidal magnetic field is in the direction of increasing coordinate of the poloidal B2 (B2.5) grid index IX, but EIRENE y- or poloidal grid index IY, or IP. Expressed in unit vectors: $\mathbf{e}_{B_\theta} = \mathbf{e}_\theta$, by default. Its magnitude (relative to the toroidal magnetic field, is determined by the B2, B2.5 pitch angle array RR(icell) and the absolute magnetic field strength $|\mathbf{B}|$ (the latter often being defaulted to $|\mathbf{B}| = 1$, because irrelevant on the EIRENE side for neutral particles, but notably not for photons (Zeeman splitting) nor for the kinetic (guiding center) trace ion options.

$B_\theta = |\mathbf{B}| \cdot$ pitch.

$B_\phi = (\mathbf{B} \cdot \mathbf{e}_\phi)$

the toroidal magnetic field is normal to the poloidal (x-y) plane in the EIRENE z (periodic cylinder, NLTRZ-option ) or PHI (approximated torus, NLTRA option) coordinate. Its strength, relative to the poloidal field, is given by the B2, B2.5 pitch array RR(icell): $B_\phi$ = $= |\mathbf{B}| \cdot \sqrt{1 - \text{pitch}^2}$. Its orientation is strictly irrelevant due to assumed toroidal symmetry, except in case of kinetic (guiding center) ion tracing by EIRENE. In the latter case it would matter indeed, e.g. to fix the $\nabla\mathbf{B}$ drift direction.

*Note: The confusing notational convention of x and y coordinates in B2-EIRENE still exists because B2 originally started as 1D parallel code, with the only coordinate being along the field, whereas EIRENE also started as 1D code, but with the first coordinate being the radial direction. (Both codes originated separately, in the early eighties of the last century, before they have been coupled together in the late eighties.)*

**IBRAD** not in use

**IBPOL** if $< 0$, then the poloidal field orientation on the EIRENE side is changed compared to the default: now $\mathbf{e}_{B_\theta} = -\mathbf{e}_\theta$, i.e. (BXIN,BYIN) $\rightarrow$ −(BXIN,BYIN) in module eirmod_infcop.f

**IBTOR** if $< 0$, then the toroidal field orientation on the EIRENE side is changed compared to the default: now $\mathbf{e}_{B_\phi} = -\mathbf{e}_\phi$, i.e. (BZIN) $\rightarrow$ −(BZIN) in module eirmod_infcop.f .

These additional B-field options now also introduce an ambiguity in the definition of parallel (to **B**) momentum exchange terms between plasma ions (in continuum approximation) and kinetic neutrals, see definition of momentum source tallies MAPL, MMPL and MIPL in section 6.1.1. Since prior to introducing the IBPOL, IPTOR flags there was $\mathbf{e}_{\|} = \mathbf{e_B}$ always, it made no difference which "parallel to **B**" unit vector was used for projections. With the additional freedom in the choice of B-field signs, i.e. with $\mathbf{e}_{\|} = \pm\mathbf{e_B}$, the proper choice of using either $\mathbf{e}_{\|}$ or $\mathbf{e_B}$ used for projecting the vectorial momentum exchange rates onto a "parallel" direction must depend on conventions used in the plasma continuum code. Current default in EIRENE is to use $+\mathbf{e_B}$ for this projection for the mentioned momentum source tallies. The same applies for momentum source terms in diamagnetic or radial directions, when plasma drifts and electrical currents are explicitly accounted for in B2 or B2.5 applications.

### 2.14.4 Version B2.5-EIRENE (2012 and later)

One distinction to the earlier versions of interfaces to the B2 plasma fluid code is that in B2.5 the energy conservation equation is solved for the internal energy only (i.e. the thermal energy, rather than for the total energy, as in B2.)
The ion energy source term $S_{Ei-internal}(icell)$ transferred from EIRENE to B2.5 in grid cell $icell$ reads:

$$S_{Ei-internal}(icell) = S_{Ei-total}(icell) - UA \times S_m(icell) + EKIN \times S_n(icell) \tag{2.26}$$

with $S_{Ei-total}, S_m(icell), S_n(icell)$ being the total ion energy source rate, the momentum source rate and the particle source rate, respectively, as defined in section 2.14.2 for the interface to the B2 plasma solver. The coefficients $UA$ and $EKIN$ are the parallel plasma flow velocity $V_{\|}$ and the kinetic energy carried by the parallel plasma flow: $EKIN = (m_{ipls}/2)V_{\|}^2$. This equation applies for a single plasma fluid case (one single plasma species $ipls$). Relation (2.26) for the internal energy source follows from either substituting the lower order moment equations into the total energy moment equation, or by forming the energy moment equation directly by averaging the kinetic equation with velocities in the plasma frame: $w = v - V$, i.e. by averaging the kinetic equation with $(m/2)w^2$ rather than $(m/2)v^2$ with $V$ the plasma flow velocity [**kn:Mueller**].
In case of multi-fluid plasmas we use, without proof (to be checked !) simply the same formula, but with the second and third term replaced by the sum over all (receiving) plasma ion species:

$$\sum_{ipls} UA(ipls)S_m(ipls, icell)\ , \quad \sum_{ipls} EKIN(ipls)S_n(ipls, icell) \tag{2.27}$$

In any case we see that the internal energy source term can be written as linear function of default tallies listed in sections 2.14.2 and 2.14.3. Hence they can be scored *per history* in newly introduced routine UPFCOP (see fig. 1.11), directly on tally COPV. Therefore the statistical error estimates for code interfacing tallies ($S_n, S_m, S_{Ei}, S_{Ee}, \ldots$) are directly available, using the standard printout and graphical output procedures of EIRENE, see discussion in section 1.3.3 on evaluation of statistical variances for linear functions of tallies.
Prior to implementation of routine UPFCOP such variances have been evaluated in case specific codes STATS1_COP.
Other differences between earlier versions of EIRCOP and the one for B2.5 (2012) are related to different interpolation schemes, the cell centered vs. surface centred fluxes, and cell indexing, in B2.5 as compared to B2.

# Chapter 3

# Problem specific Routines

**General remarks**

As mentioned earlier, already the hard wired options in EIRENE allow a large variety of linear and also of non-linear transport problems to be studied. In older versions of EIRENE (2001 and older) in most cases of linear neutral gas transport in a prescribed background plasma the only problem specific FORTRAN that is required consisted of a few parameter statements (i.e., the COMMON deck PARMUSR described below in section 3.1).
In it one had defined the storage requirements for this particular case.
**In versions EIRENE$_{2002}$ and younger this pre-assignment of storage has been removed, by a dynamical allocation of storage throughout.**
Some controllable storage options (e.g.: trading CPU - vs. storage - optimization), formerly under: "parameters for storage reductions" in PARMUSR, can now be set in an additional, but optional input line at the beginning of input block 1 (as new second line in this block), see section 2.1.

The names of all other routines in the user specified block end with . . . USR.

The description of the "additional tally" routines UPTUSR, UPCUSR, UPSUSR and UPNUSR given below (section 3.2) is meant as a guide for the more experienced users of the code only. These routines allow to extend the number of responses estimated by EIRENE almost arbitrarily.
The number of options for fast particle surface interaction models can be increased by adding subroutines REFUSR and SPTUSR, which have to provide the particle data (after reflection or sputtering, respectively) (velocity vector, weight, type, species, etc.) after a test particle has collided with a non-transparent surface and neither absorption, nor specular reflection nor the "thermal particle re-emission" is identified as surface event by the flags and the random sampling procedure. The use of this routine REFUSR is described in section 3.3.
By including a birth point sampling routine SAMUSR, one can easily model any spatial distribution of the primary source, in addition to the preprogrammed options described in section 2.7. This is explained in section 3.4.
Some geometrical variables, such as cell volumes, surface coefficients, etc., may be re-defined or modified after reading the input data. This is done by a call to routine GEOUSR, called from EIRENE subroutine INPUT. See section 3.5.1 below. Similarly, any background data (plasma profiles) may explicitly be modified in routine PLAUSR (see section 3.5.2).
The routine for additional background profiles PROUSR (called for those background tallies, for which INDPRO=5, see block 5, is described in section 3.6. It is recommended to use this option, if a particular closed form expression for the background profile is needed. It is then more convenient than the external data file options INDPRO=6, or INDPRO=7, see chapter 4.

The various routines needed for the general user supplied geometry option (NLGEN, LEVGEO=10, see block 2a) are described in section 3.8. Simple examples are also given there.

## 3.1 Parameter Statements (for EIRENE-2001 or older)

The deck "PARMUSR" is the first part of the user supplied Fortran file. It contains parameter statements used in an EIRENE run and it determines the storage required to run EIRENE on a particular problem.
In EIRENE versions after 2001 dynamic allocation of storage is implemented, the module PARMUSR has been removed. Rather than from fixed "parameter statements", the storage (such as no. of species, grid size, no. of atomic processes, etc.) is now allocated directly as identified from input flags, usually with the same name as previously in module PARMUSR. The actual "physical" numbers used in a particular run have an additional letter "I" at the end of the variable name. E.g. NATM (storage for number of different atomic species) then is identical with input flag NATMI (input block 4a), etc. Only very few exceptions exist, in which the storage parameters may be larger than the actual physical parameters in a run, such as: NSTRA and NSTRAI, the number of strata.
Such deviations may happen if a parameter is altered in the code after the automatic storage parameter evaluation in routine FIND_PARAM.f was carried out, e.g. in the above example, if the census stratum is turned off later in the run due to other parameter settings.
The module PARMUSR reads: (... stands for an integer not less than 1)

```
*COMDECK PARMUSR
C
C   Geometry
      PARAMETER (N1ST=...,  N2ND=...,  N3RD=...)
      PARAMETER (NADD=...,  NTOR=...)
      PARAMETER (NLIM=...,  NSTS=...)
      PARAMETER (NPLG=...,  NPPART=...)
      PARAMETER (NKNOT=...,  NTRI=...)
      PARAMETER (NCOORD=...,  NTETRA=...)
C   Primary Source
      PARAMETER (NSTRA=...,  NSRFS=...)
      PARAMETER (NSTEP=...)
C   Species and Tallies
      PARAMETER (NATM=...,  NMOL=...,  NION=...,  NPLS=...,
     .           NADV=...,  NADS=...)
      PARAMETER (NCLV=...,  NSNV=...,  NALV=...,  NALS=...,
     .           NAIN=...)
      PARAMETER (NCOP=...,  NBGK=...)
C   Statistics
      PARAMETER (NSD=...,  NSDW=...,  NCV=...)
C   Atomic Data
      PARAMETER (NREAC=...,  NRRC=...,  NREI=...)
      PARAMETER (          NRCX=...,  NREL=...,  NRPI=...)
C   Surface Reflection Data
      PARAMETER (NHD1=...,  NHD2=...,  NHD3=...,  NHD4=...,
     .           NHD5=...,  NHD6=...)
C   e.g.: TRIM Database
```

```
C       PARAMETER (NHD1=12, NHD2=7, NHD3=5, NHD4=5, NHD5=5,
C     .                 NHD6=...)
C   Diagnostic Chords Data
        PARAMETER (NCHOR=..., NCHEN=...)
C   Interfacing routines:
        PARAMETER (NDX=..., NDY=..., NFL=...)
C   Census Arrays
        PARAMETER (NPRNL=...)
C
C   PARAMETERS FOR STORAGE REDUCTIONS
C
C   1.) IGJUM-FLAGS FOR SPEEDUP OF GEOMETRICAL CALCULATIONS
C
C   NOPTIM: REDUCES IGJUM3-ARRAY: IGJUM3(NOPTIM,NLIMPS)
C           NOPTIM=N1ST*N2ND*N3RD+NADD:  NO STORAGE
C                                        OPTIMIZATION, SPEEDUP
C                                        OF GEOMETRICAL
C                                        CALCULATIONS IS
C                                        POSSIBLE BY CH3 FLAGS
C           NOPTIM=1 MINIMAL STORAGE, NO SPEEDUP OF
C                                     GEOMETRICAL CALCULATION
C   DEFAULT:
        PARAMETER (NOPTIM=...)
C
C   NOPTM1: BIT-ARITHMETIC/INTEGER, E.G. 32 FOR IBM RISK, 46
C           FOR CRAY
C           DEFAULT: NOPTM1=1: NO STORAGE OPTIMIZATION
C   DEFAULT:
        PARAMETER (NOPTM1=...)
C
C   2.) EXTERNAL GEOMETRY?
C   NGEOM_USR = 1 ==> EXTERNAL GEOMETRY ROUTINES (...USR),
C                     LEVGEO=10
C                     NO STORAGE FOR GEOMETRY DATA IN EIRENE
C   NGEOM_USR = 0 ==> ELSE, NO STORAGE OPTIMIZATION
C   DEFAULT:          PARAMETER (NGEOM_USR=...)
C
C   3.) SUM OVER STRATA
C   NSMSTRA = 0 ==> SUM OVER STRATA IS NOT PERFORMED
C   NSMSTRA = 1 ==> SUM OVER STRATA IS PERFORMED
C   DEFAULT:
        PARAMETER (NSMSTRA=...)
C
C   4.) CALCULATION OR STORAGE OF ATOMIC DATA
C
C   NSTORAM=0,1,2...,8,9.      =0: MINIMUM STORAGE, MAXIMUM
```

```
C                                        CALCULATION
C                                    =9: MAXIMUM STORAGE, MINIMUM
C                                        CALCULATION
C   DEFAULT:
      PARAMETER (NSTORAM = . . . )
C
C   5.) SPATIALLY RESOLVED SURFACE TALLIES?
C
C   NGSTAL = 0 ==> NO STORAGE FOR SPATIALLY RESOLVED SURFACE
C                  TALLIES
C   NGSTAL = 1 ==> SPATIALLY RESOLVED SURFACE TALLIES ARE
C                  COMPUTED
C   DEFAULT:
      PARAMETER (NGSTAL = . . . )
```

**Meaning of the Parameter Variables (and Defaults)**

**N1ST** Maximum number of standard mesh points in x- (radial) direction (block 2a)

**N2ND** Maximum number of standard mesh points in y- (poloidal) direction (block 2b)

**N3RD** Maximum number of standard mesh points in z- (toroidal) direction (block 2c)

**NADD** Maximum number of additional zones defined by the additional surfaces (block 2d)

**NTOR** Maximum number of toroidal segments for approximation of torus by cylindrical segments (block 2c)

**NLIM** Maximum number of "additional surfaces" (block 3b)

**NSTS** Maximum number of non-default standard surface models (block 3a)

**NPLG** Maximum number of points in one polygon (block 2a)

**NPPART** Maximum number of valid parts in one polygon (block 2a)

**NKNOT** Maximum number of knots for mesh of triangles (block 2a)

**NTRI** Maximum number of triangles (block 2a)

**NCOORD** Maximum number of knots for mesh of tetrahedrons (block 2a)

**NTETRA** Maximum number of tetrahedrons (block 2a)

**NSTRA** Maximum number of strata (sources) (block 7)

**NSRFS** Maximum number of sub-strata in one stratum (block 7)

**NSTEP** Maximum number of different step functions used for primary source sampling (block 7)

**NATM** Maximum number of different atom species (block 4a)

**NMOL** Maximum number of different molecule species (block 4b)

**NION** Maximum number of different test ion species (block 4c)

**NPLS** Maximum number of different bulk ion species (block 5)

**NADV** Maximum number of additional volume averaged tallies, track-length estimated (block 10a)

**NADS** Maximum number of additional surface averaged tallies (block 10d)

**NCLV** Maximum number of additional volume averaged tallies, collision estimated (block 10b)

**NSNV** Maximum number of additional volume averaged tallies, snapshot estimated (in time dependent mode only) (block 13)

**NALV** Maximum number of algebraic tallies obtained from volume averaged tallies (block 10c)

**NALS** Maximum number of algebraic tallies obtained from surface averaged tallies (block 10e)

**NAIN** Maximum number of additional input tallies, not needed by EIRENE, but to facilitate presentation (block 14)

**NCOP** Maximum number of tallies for coupling to external codes (block 14)

**NBGK** Maximum number of tallies for iterative **BGK!** scheme for neutral–neutral interactions, see MODUSR, below.

**NSD** Maximum number of standard deviation profiles for volume averaged tallies (block 9)

**NSDW** Maximum number of standard deviation profiles for surface averaged tallies (block 9)

**NCV** Maximum number of covariance estimates (block 9)

**NREAC** Maximum number of atomic reactions from external files (block 4)

**NRRC** Maximum number of re-combination processes (blocks 4,5)

**NREI** Maximum number of electron impact processes (blocks 4,5)

**NRCX** Maximum number of charge-exchange reactions (blocks 4,5)

**NREL** Maximum number of elastic reactions (blocks 4,5)

**NRPI** Maximum number of ion impact reactions (blocks 4,5)

Reflection database parameters: see example above. NHD6 is the maximum number of different target-projectile combinations, for which databases can be included.

**NCHOR** Maximum number of lines of sight for diagnostics module (block 12)

**NCHEN** Maximum number of energy values for spectra computed in diagnostics module (block 12)

**NDX, NDY, NFL**

Problem specific, in case of coupling to external code. See input block 14. Otherwise (stand alone mode): just set to 1.

**NPRNL** Maximum number of particles stored on census array, for time dependent mode (block 13)

In addition to these parameters, which depend upon the size of the particular problem under investigation, there are a few further parameters in PARMUSR, which can be used to optimize storage versus CPU performance. E.g., atomic data can either be stored in great detail, or they can be recomputed whenever they are actually needed. This is controlled by the parameter NSTORAM. The latter choice (e.g., NSTORAM=0, or NSTORAM=1) may be the better one on very large meshes (above, say, 20000 cells, as are commonly encountered in 3D stellarator applications).

## 3.2 The "Additional Tally" routines UPTUSR, UPCUSR, UPSUSR and UPNUSR

Let $f(\mathbf{r}, \mathbf{v}, j_S)$ be the distribution function in the 6 dimensional $\mu$ -space for particle species $j_S$, S = PH: Photons, S = A: Atoms, S = M: Molecules, S = I: Test Ions, S = P: Bulk Ions
e.g.:

$$\begin{aligned} j_A &= H, D, T, He, C, O, \ldots \\ j_M &= H_2, D_2, HD, CH_4, H_2O, \ldots \\ j_I &= H_2^+, CH_3^+, C^+, O^{++}, \ldots \\ j_P &= H^+, D^+, He^+, He^{++}, C^+, \ldots \end{aligned} \tag{3.1}$$

The distinction between "Test Ions" and "Bulk Ions" is problem specific, see input blocks 4 and 5. The "volume averaged tallies" estimated by EIRENE (see table 6.3) are profiles of "responses"

$$R_g =< f|g >= \sum_j \int d^3\mathbf{v} \int_{volume} d^3\mathbf{r}\; g(\mathbf{r}, \mathbf{v}, j) \cdot f(\mathbf{r}, \mathbf{v}, j)$$

for several preprogrammed and an arbitrary number of additional user supplied "detector functions" $g$. Here the summation over $j$ is only over test particle species, i.e., atoms, molecules or test ions, and not over background particles ("bulk ions", electrons). The default EIRENE estimators are not based upon moments of distribution function $f$ but upon moments of the flux

$$\Phi(j_S, \mathbf{r}, \mathbf{v}) = |\mathbf{v}| f(j_S, \mathbf{r}, \mathbf{v}),$$

hence on detector functions $g^* = g/|\mathbf{v}|$.
Occasionally also moments of the pre-collision density

$$\Psi(j_S, \mathbf{r}, \mathbf{v}) = \Phi(j_S, \mathbf{r}, \mathbf{v}) \cdot \Sigma(j_S, \mathbf{r}, \mathbf{v})$$

are utilized, then with detector functions $\hat{g} = g/(|\mathbf{v}|\ \Sigma)$. Here $\Sigma$ (dimension: 1/length) is the local total macroscopic cross section, i.e., the inverse of the local mean free path length. The estimators may, in general, be composed of a "track-length estimate"

$${}^k R_t = \sum_{i=0}^{k_{n-1}} {}^k w_i^*\, I_{l,i} = \sum_{i=0}^{k_{n-1}} {}^k w_i^* \int_{\mathbf{r}_i}^{\mathbf{r}_{i+1}} dl\; g^*(l), \tag{3.2}$$

and a "collision estimate" :

$${}^k R_c = \sum_{i=1}^{k_n} {}^k \hat{w}_i\; \hat{g}(\mathbf{r}_i). \tag{3.3}$$

Here ${}^k\hat{w}_i$ is the weight of history number $k$ before going into collision number $i$ and ${}^k w_i^*$ is the weight of history number $k$ when coming out of collision number $i$. In $R_t$ the sum is over line integrals along the track of a test flight from point of collision $\mathbf{r}_i$ to the next one at $\mathbf{r}_{i+1}, i = 0, 1, \ldots, {}^k n - 1$, whereas in $R_c$ the sum is over the points of collisions $\mathbf{r}_i, i = 1, 2, \ldots, {}^k n$.

Here $i = 0$ corresponds to the point of birth, and ${}^k n$ the number of the last collision along the track (i.e., the collision which leads into the final state "absorbed particle").
The tallies are estimated as arithmetic means

$$R = \frac{1}{N}\left(\sum_{k=1}^{N}({}^k R_t + {}^k R_c)\right) \tag{3.4}$$

of the contributions of each test particle history k.

### 3.2.1 Track-length estimated volume tallies, UPTUSR

The track-length estimators $R_t$ for detector function $g$ are updated in subroutine UPDATE (defaults) and UPTUSR (. . . , WV, . . . ) (additional user supplied tallies), with

$$WV = {}^k w_i^* / |\mathbf{v}| \quad .$$

In EIRENE variables:

$$WV = WEIGHT/VEL \quad .$$

Then, in order to estimate a response for a detector function $g$ on additional tally number ITALV, a statement in subroutine UPTUSR should read:

```
DO IC=1,NCOU
  ICELL=NRCELL+NUPC(IC)*NR1P2+NBLCKA
  ADDV(ITALV,ICELL)=ADDV(ITALV,ICELL)+WV*CLPC(IC)*g
ENDDO
```

Here $CLPC$ is the length $l$ of the flight in the cell ICELL, $g$ is assumed to be a constant in this cell in this example. In more general cases, when detector $g$ varies along the track, the product $l \cdot g$ is to be replaced by the line integral $\int g\, dl$ along this track.

### 3.2.2 Collision estimated volume tallies, UPCUSR

The default collision estimated contributions $R_c$ are updated in subroutine COLLIDE. Subroutine COLLIDE is called from the particle tracing subroutines FOLNEUT and FOLION (defaults). Non default collision tallies are updated in UPCUSR(. . . , WS, IND) (additional user supplied tallies, called from COLLIDE). Here we have

$$WS = {}^k \hat{w}_i / (|\mathbf{v}|\Sigma) \quad .$$

In EIRENE variables:

$$WS = WEIGHT/SIGTOT = WEIGHT/VEL \cdot ZMFP \quad .$$

$IND = 1$ indicates a call before sampling from the collision kernel, and $IND = 2$ indicates calls with the parameters (velocity, weight, . . . ) of the particle emerging from the collision. At each collision, the subroutine UPCUSR is called once before and once after the collision.
Then, in order to estimate a response for a detector function g on additional tally number ICOLV, a statement in subroutine UPCUSR should read:

```
COLV(ICOLV,ICELL) = COLV(ICOLV,ICELL) + WS * g
```

Here $g$ is evaluated at the point of a collision, which is known to have taken place in cell ICELL in this call to subroutine UPCUSR.
We note that track-length estimates ADDV reduce to collision estimates COLV, if ADDV is updated at the points of collision only and if the EIRENE variable ZMFP (local mean free path length) is used rather than CLPD. This fact is used by EIRENE for "condensed particle species" (NFOL$ flag in block 4). This is used, for example, if the motion of test ions along field lines is not followed explicitly but if, instead, the particles undergo a next collision immediately at their places of birth (i.e.: quasi steady state approximation "QSSA" for this species).

### 3.2.3 Snapshot estimated volume averaged tallies, UPNUSR

By default snapshot tallies are only scored in time dependent mode, see input flags in section 2.13. They can, however, also be used as a third alternative to track-length and collision estimators for any stationary response. In particular for diffusive processes, as e.g. ion transport with Fokker Planck type Coulomb collision operators, such "snapshot averaging" is the most frequent option used in Monte Carlo applications to estimate stationary moments. See paragraph B below.

#### 3.2.3.1 A: time dependent estimates

The default snapshot estimated tallies at a fixed time $t^*$, see section 2.13, are updated in subroutine TIMCOL. Currently these are only the census tallies and the particle- and energy fluxes at census. Subroutine TIMCOL is called from the particle tracing subroutines FOLNEUT and FOLION (defaults), in case of time dependent mode (see input block 13). Non default snapshot tallies can be scored in UPNUSR (additional user supplied tallies, called from TIMCOL). Here we have

$$WSNAP = {}^{k}\hat{w}_i \quad .$$

In EIRENE variables:

$$WSNAP = WEIGHT \quad .$$

Then, in order to estimate a response for a detector function $g$ on snapshot tally number ISNV, a statement in subroutine UPNUSR should read:

```
SNAPV(ISNV,ICELL) = SNAPV(ISNV,ICELL) + WSNAP * g(...)
```

E.g., with $g = 1$ this is an unbiased estimate of cell averaged particle density at time $t = t^*$, see section 3.2, i.e., just counting the particle weights at $t = t^*$ provides an unbiased estimate of the particle density. This is intuitively clear, but also mathematically correct since these snapshot estimators are formally derived as special cases of track-length- collision or surface flux estimators (interpreting the time-horizon sub-manifold as "time-surface", by abuse of language) by including in $g$ a delta function in time : $g(t, \mathbf{r}, \mathbf{v}) = \delta(t^*)\tilde{g}(\mathbf{r}, \mathbf{v})$.
Subroutine UPNUSR must at least include module COMPRT (for the particle weight WEIGHT). Here $g$ is evaluated at the point of time $t^*$ at which TIMCOL is called. Scaling of snapshot tallies is done by interpreting the linear source strength "FLUX" scaling factor (given in input block 7 (section 2.7) for each stratum) as a total number of particles, rather than as flux (particles $s^{-1}$), i.e. "FLUX is the integral of the source distribution not only over physical space, but also over time interval $DTIMV$ from initial time $t_0$ to time $t^*$ (section 2.13).

#### 3.2.3.2 B: stationary snapshot tallies

A stationary value of snapshot tallies, to be compared with other tallies in stationary mode, is obtained by setting the flag NTMSTP of block 13 (section 2.13) to negative values. Stationary results in this "time dependent mode" are obtained by launching all test flights at time $t_0$, but scoring snapshot tallies not only at $t^* = t_1 = t_0 + DTIMV$, but also at $t_2 = t_0 + 2\ DTIMV, t_3 = t_0 + 3\ DTIMV, \ldots$, i.e. at all times $t_n = t_0 + n\ DTIMV$ until the flight terminates. If the source distribution and the background medium parameters are constant in time, then the snapshot score contribution from time $t_i$ can be interpreted as score at $t^* = t_1$ resulting from the source at earlier time $t_0 - (i-1)DTIMV$. Hence this stationary snapshot score at $t^*$ is in fact an integration over all contributions from earlier times $t < t_0$.
When scaling the time- snapshot tallies at $t = t^*$ in this stationary mode to absolute units, then an extra multiplicative factor $DTIMV$ (i.e., the time interval between subsequent scores along a particle history) is applied for snapshot tallies, because the stationary (time-averaged) volume averaged tallies are scaled interpreting the stratum source strength $FLUX$ as flux: "atomic particles per second", whereas the snapshot tallies require this same scaling factor $FLUX$ to be the absolute number of (atomic) particles (flux integrated over one time interval from $t_0$ to $t_1 = t_0 + DTIMV$).
Further considerations regarding the scaling factors for volume averaged tallies in EIRENE, with respect to dimensionality of phase space of the problem (1D, 2D, 3D, stationary or time-dependent) are given in section 1.3.2.

### 3.2.4 Surface averaged tallies, UPSUSR

Finally we note that "surface averaged tallies" (table 6.4) may be considered just as special cases of the "volume averaged tallies" described above, if appropriate use is made of $\delta$ -functions. Let therefore $\rho$ be a co-ordinate normal to a surface S at the strike point $\underline{r}_S$ of a test flight from $\underline{r}_i$ to $\underline{r}_{i+1}$ with speed unit vector $\underline{\Omega} = \underline{v}/(|\underline{v}|)$. I.e., the surface is described locally by the equation $\rho = 0$ at the point $\mathbf{r}_S \in S$, at which we may define an orthonormal basis $\mathbf{e}_\rho, \mathbf{e}_{\rho'}, \mathbf{e}_{\rho''}$. Furthermore, $\mathbf{r}_S = \mathbf{r}_{i+1}$, which means that one may treat surface events exactly in the same way as collisions with the background medium in our terminologies. Then a "surface response function" $g_S$ defined as

$$g_S(j, \mathbf{r}, \mathbf{v}) = \delta(\rho) \cdot g(j, \mathbf{r}, \mathbf{v})$$

is the detector function for the same response $R_g$ as previously detector function $g$ was, but now surface averaged instead of volume averaged:

$$\int_{surface} d^2s\ g \cdot f = \int_{volume} d^3r\ g_S \cdot f$$

Therefore, the line integrals $I_{l,i}$ in the formula given above for track-length estimates, equation (3.2), now reduce to:

$$I_{l,i} = \int_{\mathbf{r}_i}^{\mathbf{r}_{i+1}} dl\ \delta(\rho) \cdot g^*(l) = g^*(\rho = 0)/\cos(\mathbf{e}_\rho, \mathbf{\Omega}) \tag{3.5}$$

if the surface is intersected during the test-flight from $\mathbf{r}_i$ to $\mathbf{r}_{i+1}$, and $I_{l,i} = 0$ else. This can easily be seen by writing

$$\mathbf{r}_{l,i} = \mathbf{r}_i + l \cdot \begin{pmatrix} \cos(\mathbf{\Omega}, \mathbf{e}_\rho) \\ \cos(\mathbf{\Omega}, \mathbf{e}_{\rho'}) \\ \cos(\mathbf{\Omega}, \mathbf{e}_{\rho''}) \end{pmatrix} \tag{3.6}$$

i.e., $\rho = r_{\rho,i} + l \cdot \cos(\mathbf{\Omega}, \mathbf{e}_\rho)$. In other words: track-length estimators become formally identical to collision estimators, for detector functions containing surface collision delta functions.
Consequently, a statement in subroutine UPSUSR(WEIGHT,IND) for this surface averaged tally must read:

$$ADDS(ITALS, IS) = ADDS(ITALS, IS) + WC \cdot g \quad ;$$

here $IS$ is the index of the surface which is being crossed, $ITALS$ is the labelling number of the surface tally. Furthermore,

$$WC = {}^k\omega_i^* / [|\mathbf{v}| \cdot \cos(\mathbf{\Omega}, \mathbf{e}_\rho)],$$

the detector function $g$ is evaluated at the strike point $\mathbf{r}_S$ and, as above, ${}^k\omega_i^*$ is the weight of the history k after event i and before event i+1. Note:

$$WEIGHT = {}^k w_i^* = {}^k \hat{w_{i+1}}$$

in our terminology.
The flag IND has the value 1, if the particle is incident onto the surface, and IND = 2 if the particle is emitted from the surface.
Some care is needed because the local surface normal unit vector $\mathbf{e}_\rho$ at the strike point $\mathbf{r}_S$ is not known in subroutine UPSUSR unless the surface input variable ILIIN (see input block 3) is positive (equal to 1,2,3 or 4) for the surface $S$. In all other cases, this vector (defined by its Cartesian components CRTX,CRTY,CRTZ in EIRENE) has to be computed in subroutine UPSUSR at each entry, if it is needed for updating the surface averaged tally.
Subroutine UPSUSR is called whenever the default surface tallies are updated. For one sided surface tallies (ILIIN=-2 or ILIIN=-4), therefore, UPSUSR is also only called for one sided tallies only.
In addition to these calls, UPSUSR is also called from surface source routines. For test particles emitted from source surfaces (or generated from incident bulk ions after reflection or sputtering) the call is UPSUSR(WEIGHT,2). For bulk ions (ITYP=4) incident onto a source surface the call is UPSUSR(-WEIGHT,1).
Therefore, the tallies ADDS include the direct source contribution, if the source is a surface source, whereas the default surface tallies don't.

## 3.3 The user surface reflection model REFUSR

**General remarks**:

This subroutine is called in the initialization phase of an EIRENE run, at entry RF0USR, from subroutine REFLEC, and at entry SP0USR, from subroutine SPUTER. At these entries the user supplied reflection models and sputtering models (if any), respectively, may be initialized.
Later, during the Monte Carlo sampling phase, it is called via entry RF1USR, whenever a test flight intersects a surface NLLI, for which the fast particle reflection model flag ILREF(NLLI) has been set equal to ILREF(NLLI)=3.
Furthermore, if for a certain surface NLLI a user specified "sputter model" is activated by the flag ILSPT(NLLI)=3, then REFUSR is called at the entry SP1USR, whenever this surface is intersected by a test flight.

**Format of subroutine**

```
      SUBROUTINE REFUSR
C USER SUPPLIED SURFACE INTERACTION ROUTINE
C INPUT : CO–ORDINATES OF INCIDENT PARTICLE
C OUTPUT: CO–ORDINATES OF EMITTED PARTICLE
* CALL PARMMOD
      :
      :
C   INITIALIZE USER SUPPLIED REFLECTION MODEL
      ENTRY RF0USR
      :
C   DEFINE SPECIES DEPENDENT RECYCLING COEFFICIENT HERE
C     RECYCT(ISPZ,MSURF)=.....
      :
      RETURN
      ENTRY RF1USR (XMW,XCW,XMP,XCP,IGASF,IGAST,ZCOS,ZSIN,
     .              EXPI,RPROB,E0TERM,*,*,*,*)
C   RETURN 1: EIRENE STANDARD ANGULAR DISTRIBUTION (DEP. ON
C             ''EXPI")
C   RETURN 2: THERMAL MOLECULE MODEL (DEP. ON ''IGAST",
C             ''E0TERM")
C   RETURN 3: THERMAL ATOM MODEL (DEP. ON ''IGAST", ''E0TERM")
C   RETURN 4: ABSORB PARTICLE AT THIS SURFACE
      :
      :
      RETURN
C
C   INITIALIZE USER SUPPLIED SPUTTERING MODEL
      ENTRY SP0USR
      :
C   DEFINE SPECIES DEPENDENT PHYSICAL SPUTTERING COEFFICIENT
C   HERE
```

```
C       RECYCS( ISPZ , MSURF ) = . . . . .
C   AND ALSO CHEMICAL SPUTTERING COEFFICIENT
C       RECYCC( ISPZ , MSURF ) = . . . . .
        :
        ENTRY SP1USR
        :
        :
        RETURN
        END
```

## 3.4 The user source sampling routine SAMUSR

**General remarks**:

This subroutine is called from the point source sampling routine SAMPNT, the surface source sampling routine SAMSRF, or from the volume sampling routine SAMVOL, if the flag SORLIM(N, ISTRA) (input block 7) for the spatial distribution of birth points on sub-stratum N for stratum ISTRA has a negative value.
This subroutine returns to EIRENE the (Cartesian) coordinates of the birth point
X0,Y0,Z0
cell index information
IRUSR,IPUSR,ITUSR,IAUSR,IBUSR
as well as local plasma background data at this place of birth:
TIWL,TEWL,DIWL,VXWL,VYWL,VZWL,EFWL,SHWL,WEISPZ
The input parameters (EIRENE input block 7 for primary sources)
SORAD1,SORAD2,SORAD3,SORAD4,SORAD5,SORAD6
can be used in this problem specific routine.
Dimensions, Precision:

**Format of subroutine**

```
      SUBROUTINE SAMUSR (ISR,X0,Y0,Z0,SORAD1,SORAD2,SORAD3,
     .                   SORAD4,SORAD5,SORAD6,IRUSR,IPUSR,
     .                   ITUSR,IAUSR,IBUSR,TIWL,TEWL,DIWL,
     .                   VXWL,VYWL,VZWL,EFWL,SHWL,WEISPZ)
      USE PRECISION
      USE PARMMOD
            :
C MORE MODULES, IF NEEDED
            :
      IMPLICIT NONE
      INTEGER, INTENT(IN) :: ISR, ISTR
      REAL(DP), INTENT(IN)  :: SORAD1,SORAD2,SORAD3,SORAD4,
     .                         SORAD5,SORAD6
      REAL(DP), INTENT(OUT) ::
     .           X0,Y0,Z0
      INTEGER,  INTENT(OUT) :: IRUSR,IPUSR,ITUSR,IAUSR,IBUSR
C     NPLS: NUMBER OF BULK ION SPECIES IN EIRENE INPUT BLOCK 5
C     NSPZ: TOTAL NUMBER OF PARTICLE SPECIES:
C           NSPZ=NPHOT+NATM+NMOL+NION+NPLS,
C           SEE INPUT BLOCKS 4 AND 5.
      REAL(DP), INTENT(OUT) ::
     .           TEWL,SHWL,
     .           TIWL(NPLS),DIWL(NPLS),
     .           VXWL(NPLS),VYWL(NPLS),VZWL(NPLS),
     .           EFWL(NPLS),WEISPZ(NSPZ)
```

```
C   INITIALIZE SAMPLING FOR SUBSTRATUM  ISR OF STRATUM ISTR
C   ISR:  NUMBER OF SUBSTRATUM (SEE INPUT BLOCK 7)
C   ISTR: NUMBER OF STRATUM
      ENTRY SM0USR (ISR,istr,sorad1,sorad2,sorad3,
     .                       sorad4,sorad5,sorad6)

      RETURN

      ENTRY SM1USR (ISR,X0,Y0,Z0,SORAD1,SORAD2,SORAD3,SORAD4,
     .              SORAD5,SORAD6,IRUSR,IPUSR,ITUSR,IAUSR,
     .              IBUSR,TIWL,TEWL,DIWL,VXWL,VYWL,VZWL,EFWL,
     .              SHWL,WEISPZ)
C   FIND BIRTH POINT COORDINATES. SUBSTRATUM ISR IF STRATUM
C   ISTRA HAS ALREADY BEEN IDENTIFIED. ISTRA (IF NEEDED) IS IN
C   MODULE COMPRT.
C   INITIALIZE THOSE VARIABLES WHICH ARE NOT SET BELOW:
      X0   =0._DP
      Y0   =0._DP
      Z0   =0._DP

      IRUSR =0
      IPUSR =0
      ITUSR =1
      IAUSR =0
      IBUSR =1

      TEWL=0._DP
      SHWL=0._DP
      TIWL=0._DP
      DIWL=0._DP
      VXWL=0._DP
      VYWL=0._DP
      VZWL=0._DP
      EFWL=0._DP
      WEISPZ=0._DP

C   HERE COMES THE PROBLEM SPECIFIC DEFINITION OF BIRTH POINT
C   SAMPLING


      RETURN

      END
```

Note: The parameters EFWL, SHWL have been introduced in 2005.

Note: Correction in May 2006: In previous versions of this manual the parameters TEWL and TIWL have been interchanged with respect to the calling statements in the EIRENE code. (Thanks to Jose Guasp, (CIEMAT, Spain) for pointing this out.)

## 3.5 The user routines to overrule input data

### 3.5.1 The user geometry data routine GEOUSR

to be written

### 3.5.2 User supplied background data routine PLAUSR

to be written

## 3.6 The user routines for profiles PROUSR

**General remarks**:

Purpose: set input tallies (background medium), on the grid (one number per tally per cell) For more general input tallies, evaluated "on the fly" at a particular position, see routine VECUSR below (section 3.6.1).
This subroutine is called from the subroutine PLASMA if the input flag INDPRO (input block 5) has the value INDPRO(IPRO)=5 for a particular background tally IPRO. Only the data for cells from the standard mesh (ICELL = 1, . . . , NSURF) are used. Default "vacuum data" are set for all cells in the "additional cell region" ICELL = NSURF + 1, . . . , NSBOX.
PROUSR is called, furthermore, from subroutine INPUT to provide cell volumes for all cells (ICELL = 1, . . . , NSBOX), if the input flag INDPRO(12) has the value INDPRO(12)=5

**Format of subroutine**

```
      SUBROUTINE PROUSR (RHO,INDEX,P0,P1,P2,P3,P4,P5,PVAC,
     .                   NDAT)
* CALL PARMMOD
      :
      :
      IF (INDEX.EQ.0) THEN
C   DATA FOR TE-PROFILE, NDAT=NSURF
        DO ICELL=1,NDAT
          RHO(ICELL) = ......
        ENDDO
      ELSEIF (INDEX.EQ.....) THEN
      :
      :
      ELSEIF (INDEX.EQ.5+5*NPLS) THEN
C   DATA FOR VOL, NDAT=NSBOX
      :
      :
      ELSE
        WRITE (6,*) 'INVALID INDEX IN PROUSR '
        CALL EXIT
      ENDIF
      RETURN
      END
```

Depending upon the value of INDEX, one has to specify a background tally on the array RHO, RHO(J),J=1,NDAT. These are:

**INDEX = 0** Electron temperature (eV)

**INDEX = 1** Ion temperature (eV), NPLSI calls, one for each ion species. I.e.:

```
DO IPLS=1,NPLSI
  CALL PROUSR(HELP,1+0*NPLSI,TI0(IPLS),...,NSURF)
  CALL RESETP(TIIN,HELP,IPLS,1,NPLS,NSURF)
ENDDO
```

**INDEX = 1+ NPLS** Ion density ($cm^{-3}$), NPLSI calls, one for each ion species

**INDEX = 1+2*NPLS** Ion drift velocity, x direction ($cm\,s^{-1}$), NPLSI calls, one for each ion species

**INDEX = 1+3*NPLS** Ion drift velocity, y direction ($cm\,s^{-1}$), NPLSI calls, one for each ion species

**INDEX = 1+4*NPLS** Ion drift velocity, z direction ($cm\,s^{-1}$), NPLSI calls, one for each ion species

**INDEX = 1+5*NPLS** Magnetic field, x component, (AU)

**INDEX = 2+5*NPLS** Magnetic field, y component, (AU)

**INDEX = 3+5*NPLS** Magnetic field, z component, (AU)

**INDEX = 4+5*NPLS** Magnetic field, magnitude, (T)

**INDEX = 5+5*NPLS** Cell volume, ($cm^3$)

**INDEX = 6+5*NPLS** Additional tally, NAINI calls, one for each additional quantity.

### 3.6.1 User supplied background data routine VECUSR

This routine is called at a particular position (X0, Y0, Z0) in grid cell NCELL, and returns a vector quantity. Position and cell number information are taken from module EIRMOD_COMPRT.F. It is used in particular if the magnetic field, or plasma flow field, are needed with a higher spatial resolution than the one provided by the standard input plasma (background) tallies, which are cell averaged, i.e. constant within a cell.

```
SUBROUTINE VECUSR(INDEX,VEC_X,VEC_Y,VEC_Z,IPLS)
```

This routine is called in case INDPRO(5)=8 for the B-field tally (e.g. from routine BFIELD.F), and in case (INDPRO(4)=8) for the background flow velocity tallies (from many places in the code), to provide either a local magnetic field (unit) vector, or a local plasma (background medium) flow velocity vector ($cm\,s^{-1}$), respectively, in Cartesian coordinates.
Current implementation:

**INDEX = 1** return the magnetic field unit vector, Cartesian coordinates. This local (at position X0, Y0, Z0, in cell NCELL) vector is used instead of cell averaged input tally no. 5:

[BXIN(ncell), BYIN(ncell), BZIN(ncell)].

VECUSR with INDEX=1 is called from routine BFIELD.F

**INDEX = 2** return plasma (background) flow velocity vector, Cartesian coordinates, for EIRENE background species IPLS. This local (at position X0, Y0, Z0, in cell NCELL) vector is used instead of cell averaged input tally no. 4:

[VXIN(IPLS,ncell), VYIN(IPLS,ncell), VZIN(IPLS,ncell)].

Note: species index IPLS is the EIRENE background species index, not to be confused with the mapped species index MPLSV(IPLS).

## 3.7 User supplied post-processed tally routine TALUSR

to be written

```
      SUBROUTINE TALUSR(ICOUNT,VECTOR,TALTOT,TALAV,
     .                  TXTTL,TXTSP,TXTUN,ILAST,*)
```

## 3.8 User supplied "general geometry block"

The following routines have to be provided for the general geometry options (LEVGEO=10 option, NLGEN=TRUE), in which no specific geometry data are available from EIRENE input, and all the geometrical parameters for particle tracing and scoring of tallies are transferred from outside.

- INIUSR (initialize user specified geometry block), see section 3.8.1
- LEAUSR(x) (return cell number, for any given position x), see section 3.8.2
- TIMUSR (flight time to next cell boundary, and next cell number or surface number), see section 3.8.3
- VOLUSR (volume of each grid cell), see section 3.8.4
- NORUSR (outer surface normal, for any given position on a surface), see section 3.8.5

In case NLGEN, only a reduced set of the standard EIRENE options is available, and the five user routines mentioned above and described below may have to be supplemented by some further options from the problem specific segment USER.F .

**Input block 5** only the INDPRO = 3, 5 and 6 options are available. INDPRO = 3 provides constant profiles, same value in each cell. Hence, in case of non-constant background parameters one has to resort to subroutine PROUSR (INDPRO=5), or to the code segment INFCOP (INDPRO=6) for coupling to another source (code) for the data of the background medium.

**Input block 7** only the point source option is available. In case of surface sources or volume sources, the subroutine SAMUSR has to be called, see section 3.4.

**Input block 11** only the printout options are available, no graphics output options are available. However, subroutine PLTUSR(LOG,J) is called from the 2D geometry plotting routine PLT2D (LOG=.TRUE., J = number of surface to be plotted) and from the 3D geometry plotting routine PLT3D (LOG=.FALSE., J = number of surface to be plotted)

### 3.8.1 Subroutine INIUSR

This subroutine is called from subroutine INPUT, after reading from the formatted input file and after (optional) calls to interfacing routines INFCOP. Any initialization (including definition of COMMON blocks) for the user geometry package may be done here.

### 3.8.2 Subroutine LEAUSR

Identify cell index from a point given by its three Cartesian coordinates.
The function LEAUSR is called from functions LEARC1 and LEARCA in case of LEVGEO = 10. The call is

NCELL=LEAUSR(X,Y,Z)

Here X,Y,Z are the Cartesian coordinates of a point inside the computational domain in centimeters. LEAUSR must then return the number of the cell to which this point belongs.
Note (see section 2.2): Any particular cell in EIRENE can be specified in one of two possible ways:
Either by giving the 5 cell numbers:
NRCELL, NPCELL, NTCELL, NBLOCK, NACELL
in the first,(radial) second (poloidal) third (toroidal) grid or additional cell region, or, alternatively, by giving the cell index in the 1-dimensional cell array
NCELL

The relation between these two options is:
NCELL = NRCELL + ((NPECLL-1)*NR1P2) + (NTCELL-1))*NP2T3 + NBLCKA
with
NBLCKA = (NBLOCK -1) * NSTRDT +NRADD
LEAUSR is expected to return the cell index NCELL.

### 3.8.3 Subroutine TIMUSR

The subroutine TIMUSR is called from TIMER (block GEO3D) in case of LEVGEO=10 (general external geometry option). Distinct from the other geometry levels in TIMER, the number of the nearest intersected surface (MRSURF) along the trajectory is not returned in general. Only if this nearest surface is one of the "non-default standard surfaces" defined in input block 3a, this information is returned (by a negative value of argument NEWCEL.
The call is

```
CALL TIMUSR (NRCELL,X0,Y0,Z0,VELX,VELY,VELZ,
.            NJUMP,NEWCEL,TIM,ICOS,IERR,NPANU,NLSRFX)
```

**Input**

**NRCELL** Actual cell number, for which the intersection to the cell boundary has to be found. If NJUMP=0, i.e., for the first call of a new trajectory (e.g., after a collision), the starting point X0,Y0,Z0 must be in that cell. In later calls for the same trajectory (NJUMP > 0), NRCELL has been automatically updated, i.e., it must not necessarily contain the starting point X0,Y0,Z0, and the flight time (distance) from the starting point is accumulated.

**X0,Y0,Z0** Cartesian coordinates of the starting point of a trajectory.

**VELX,VELY,VELZ**

unit (speed-) vector pointing in the direction of the flight.

**NJUMP** $= 0$ this is the first call for a particular trajectory.

$\neq 0$ this is a later call for a particular trajectory. The initial position and speed unit vector are the same as in the previous call. Hence: the geometrical parameters depending only on those need not be evaluated again.

**NPANU** number of EIRENE test flight (particle history). Only for diagnostic printout from TIMUSR.

**NLSRFX** TRUE: the starting position (X0,Y0,Z0) of the current trajectory is exactly on the cell boundary of cell NRCELL. This information may by used inside TIMUSR to detect and exclude flight paths of zero length (time)

FALSE: Particle trajectory does not start at the cell boundary of cell NRCELL, but inside this cell.

**Output**

**NEWCEL** $< 0$ trajectory has intersected one of the non-default surfaces (input block 3a). ABS(NEWCEL) is the number of that surface (corresponding to running index ISTSI in input block 3a).

$> 0$ number of next cell, in which the flight would continue, if no collision event stops the track already earlier. I.e., cell number of the neighbor cell in the direction of the flight.

**TIM** distance (cm) from [X0,Y0,Z0] to the nearest intersection of the trajectory with a boundary of cell NRCELL. Since the speed vector [VELX,VELY,VELZ] is normalized to one (cm $s^{-1}$), this distance is also the flight time (s)

**ICOS** only relevant, if the trajectory has intersected a non-default surface. In this case, ICOS is the sign (+1 or -1) of the cosine of the angle of incidence against the surface normal (this latter unit vector, [CRTX,CRTY,CRTZ] may, e.g., be found in TIMUSR by a call to NORUSR, see section 3.8.5).

**IERR** error flag. Presently any value different from 0 will lead to an immediate end of the run. See subroutine TIMER in code segment GEO3D.F

### 3.8.4 Subroutine VOLUSR

The subroutine VOLUSR is called from subroutine VOLUME in case of LEVGEO=10. The call is

**CALL** VOLUSR(N,A)

Here N is the total number of cells in this run, and A is an array of length N, containing the N cell volume values in $cm^3$.

### 3.8.5 Subroutine NORUSR

The subroutine NORUSR provides the outer surface normal unit vector CX,CY,CZ at surfaces, i.e. after a particle trajectory has intersected a “non default grid surface” at point X,Y,Z.
It is called in the LEVGEO=10 option from subroutine STDCOL, after all surface intersection settings and possible switches have been carried out, or directly via the entry point STDNOR of STDCOL. The call is

```
CALL NORUSR(M,X,Y,Z,CX,CY,CZ,SCOS,VELX,VELY,VELZ,
.                    NRCELL)
```

Here [X,Y,Z] are the Cartesian coordinates of an intersection point, located on non-default standard surface no. M. The routine must return the surface normal unit vector [CX,CY,CZ].
If surface M has a periodicity condition (i.e. surface flag ILIIN(M) >3 in input block 3a (section 2.3.1) then this routine must also carry out the periodicity transformation for the test particle (which may be different for neutral particles and for charged (test ion) particles).
Periodicity is carried out, in general, by moving particles to another surface and/or altering its speed unit vector.
In this cases the code expects on output in [X,Y,Z] the new coordinates, on integer variable M the new surface number, and, if applicable, on [VELX,VELY,VELZ] the new speed unit vector. NRCELL must contain the cell number, into which the new flight enters from surface M.
Currently NORUSR is called for all non default surfaces in the general (external) geometry level LEVGEO=10, i.e. for NRGEN = T.
It is also called (currently without warning or conditioning) for tetrahedral grids at periodicity surfaces.

# Chapter 4

# Routines for interfacing with other codes: EIRCOP

### General remarks

The "background" or "host" medium, usually the "plasma" is fixed in an EIRENE run, and usually either specified in input block 5 (section 2.5), or the background volume tallies are simply ready from data files.
However the mutual coupling between test-particle species treated by EIRENE and the background medium may be very strong, even if the test particle density is quite low compared to the background (plasma) density. This is because in-elastic collisions may lead to strong particle, momentum and/or energy sources for the balances governing the background medium, and these "chemical sources" may even be the dominant terms for the global flow problem, at least in some parts of the computational volume (e.g. typically in the divertor of fusion reactors). Original diffusion-advection equations for a background medium there may turn into diffusion-advection-reaction type equations and even change their mathematical characteristics.
For such type of "background medium models" (often "Navier Stokes like" or related) then special interfacing procedures between the kinetic Monte Carlo component (EIRENE) and the numerical background model are required.
For the EIRENE code such procedures have been developed in the late eighties, originally for the SOLXY 2D plasma code ([**kn:Gerhauser88a**]) but have then been transferred and extended for the rather well established (at that time) 2D multi-fluid plasma solver B2.
The EIRENE modules developed for code interfacing are described in this section, original references are: Reiter, [**kn:Net**], and first applications [**kn:Reiter91b**] (1991) and the most detailed discussion probably still is the FZ Jülich report JUEL-2872, Maddison, Reiter, [**kn:Maddison94**], (1994).
Special Monte Carlo options, such as correlated sampling (see flag NLCRR in input block 1, section 2.1) or the semi-implicit coupling schemes based on source term adaptations/rescalings to reduce Monte Carlo run-time in these iterative non-linear schemes (jargon: "short cycles"), have also had their origin in those early studies, loc cit., but have meanwhile been generalized significantly, e.g. to allow selective "short cycling" for individual strata, while for others (notably for volume recombination sources) full new Monte Carlo estimates are retained.
The names of all routines in the interfacing block end with . . . COP.

## 4.1 Routine for interfacing INFCOP

To write an interfacing routine INFCOP in order to couple EIRENE to another code is already a quite formidable task, a new scientific challenge and it is highly recommended that, in addition to the information given below, close interaction with scientists at FZ Jülich may be needed. One also might ask for a few sample routines INFCOP, which are available from the authors.
In the rest of this section plasma data and geometrical (grid) information, from interfacing with an external model/code are described with reference to their location and size on a single large, 1D storage array RWK. This array RWK which is then used in the initialization phase of an EIRENE run to bring this information into regular EIRENE storage places/names.
In Version from 2001 or later, rather than filling 1D array RWK, now directly input tallies TEINTF, TIINTF, DIINTF, etc. can be filled. These are pointers now to a vector (now named PLASMA_BACKGROUND), but which has the same structure as the old RWK array. Apart from that the rest remains the same and the material in this section is still valid.
Data are transferred from subroutine INFCOP into EIRENE via the EIRENE work array RWK (Module CSPEI). They are read onto the EIRENE arrays for input volume tallies (table 6.2) by calls of subroutine PROFR in the initialization phase individually for each tally, for which the flag INDPRO has the value 6 or 7 for a particular input tally (see section 2.5). The following addresses are foreseen on RWK for this data transferring procedure:
*Plasma data* (*INDPRO* = 6 *option*)

```
TEIN   :   (RWK ((0            ) * NRAD + J), J=1,NSBOX)
TIIN   :   (RWK ((1 + 0*NPLS) * NRAD + J), J=1,NSBOX, I=1,NPLS)
DIIN   :   (RWK ((1 + 1*NPLS) * NRAD + J), J=1,NSBOX, I=1,NPLS)
VXIN   :   (RWK ((1 + 2*NPLS) * NRAD + J), J=1,NSBOX, I=1,NPLS)
VYIN   :   (RWK ((1 + 3*NPLS) * NRAD + J), J=1,NSBOX, I=1,NPLS)
VZIN   :   (RWK ((1 + 4*NPLS) * NRAD + J), J=1,NSBOX, I=1,NPLS)
BXIN   :   (RWK ((1 + 5*NPLS) * NRAD + J), J=1,NSBOX)
BYIN   :   (RWK ((2 + 5*NPLS) * NRAD + J), J=1,NSBOX)
BZIN   :   (RWK ((3 + 5*NPLS) * NRAD + J), J=1,NSBOX)
BFIN   :   (RWK ((4 + 5*NPLS) * NRAD + J), J=1,NSBOX)
VLIN   :   (RWK ((5 + 5*NPLS) * NRAD + J), J=1,NSBOX)
ADIN   :   (RWK ((6 + 5*NPLS) * NRAD + J), J=1,NSBOX, I=1,NAIN)
```

*Geometrical data* (*INDGRD* = 6 *option*)

```
LEVGEO = 1 or LEVGEO = 2:
RSURF  :   (RWK ((6+5*NPLS+NAIN) * NRAD +                    J),
             J=1,N1ST)
EP     :   (RWK ((6+5*NPLS+NAIN) * NRAD +   N1ST +          J),
             J=1,N1ST)
EL     :   (RWK ((6+5*NPLS+NAIN) * NRAD + 2*N1ST +          J),
             J=1,N1ST)
TR     :   (RWK ((6+5*NPLS+NAIN) * NRAD + 3*N1ST +          J),
             J=1,N1ST)
PSURF  :   (RWK ((6+5*NPLS+NAIN) * NRAD + 4*N1ST +          J),
             J=1,N2ND)
TSURF  :   (RWK ((6+5*NPLS+NAIN) * NRAD + 4*N1ST + N2ND +  J),
```

```
                J=1,N3RD)
LEVGEO = 3
PGINTF:   (RWK ((6+5*NPLS+NAIN) * NRAD +                    J),
                J=1,NPMAX)
TSURF  :  (RWK ((6+5*NPLS+NAIN) * NRAD + NPMAX +            J),
                J=1,N3RD)
LEVGEO = 4
TRINTF:   (RWK ((6+5*NPLS+NAIN) * NRAD +                    J),
                J=1,NTMAX)
TSURF  :  (RWK ((6+5*NPLS+NAIN) * NRAD + NTMAX +            J),
                J=1,N3RD)
LEVGEO = 5
THINTF:   (RWK ((6+5*NPLS+NAIN) * NRAD +                    J),
                J=1,NHMAX)
```

### 4.1.1 entry IF0COP

*Geometry data* (*INDGRD* = 6 *option*)

In the LEVGEO = 3 option (2D grid of quadrangles) PGINTF is an array of length NPMAX, which contains all relevant information to generate a 2D mesh of polygons in the x-y plane. It is equivalenced to the common block CPOLYG via the statement:
EQUIVALENCE (PGINTF(1),XPLG(1,1)), . . .
In the LEVGEO = 4 option (2D grid of triangles) TRINTF is an array of length NTMAX, which contains all relevant information to generate a mesh of triangles in the x-y plane. It is equivalenced to the common block CTRIA via the statement:
EQUIVALENCE (TRINTF(1),XT(1)), . . .
In the LEVGEO = 5 option (3D grid of tetrahedrons) THINTF is an array of length NHMAX, which contains all relevant information to generate a mesh of tetrahedrons in the 3d computational domain. It is equivalenced to the common block CTETRA via the statement:
EQUIVALENCE (THINTF(1),XT(1)), . . .

### 4.1.2 entry IF1COP

*Plasma* (*background*) *data* (*INDPRO* = 6 *option*)

to be written

### 4.1.3 entry IF2COP(ISTRA)

*Primary source data* (*INDSRC* = 6 *option*)

In case of an EIRENE run, in which input information is obtained from an external code (e.g.: B2, EMC3, DIVIMP, . . . ), the stratification of the primary source should be such that the first NTARGI (see section 2.14) strata are determined by the plasma fluxes onto surfaces (“recycling surface sources”). In this case the surface source distribution $Q_S(\mathbf{r}, \mathbf{v}, i, t)$ (section 1.3) can be defined automatically from the interfacing routine, using the data specified in input block 14.

The remaining primary sources NTARGI+1, . . . , NSTRAI (e.g. gas puff, volume recombination sources, etc.) still have to defined in input block 7. Whether input block 7 or input block 14 is used for a particular surface source ISTRA ≤ NTARGI is controlled by the flag INDSRC. For sources ISTRA > NTARGI the input flag INDSRC is irrelevant.
If INDSRC(ISTRA) < 0 then interfacing routine IF2COP is not called. Input data are used from block 7, see section 2.7.

If INDSRC(ISTRA) = 0–5, then the input flags described in section 2.7 are used to define the spatial distribution of a recycling source on a grid boundary (function STEP(ISTEP), see section 2.7.1) as well as the distribution in velocity space. The step function itself and the total source strength FLUX(ISTRA), however, are defined in IF2COP, using data from input block 14, see section 2.14, for target recycling source ITARG. In this case the sampling range from the spatial surface distribution and the species distribution specified in input block 7 must be equal to or a subrange of the the step-function for target recycling source ITARG defined from the data in input block 14.

If INDSRC(ISTRA) = 6, then the input flags described in section 2.14 are used to define the entire surface recycling source automatically. The corresponding data in input block 7 for this stratum are not used and may be omitted. Stratum ISTRA corresponds to the target recycling source ITARG from block 14.

### 4.1.4 entry IF3COP(ISTRAA,ISTRAE)

*Return data at the end of an EIRENE stratum*

At this entry data are transferred back from EIRENE to the interfacing module (and from there, after possibly further processing, to the external code). This entry is called from the "strata-loop" in subr. MCARLO, after all trajectories for a particular stratum ISTRA have been sampled and after all volume and surface tallies have been scaled and processed to their final form.
The call to IF3COP is controlled by the flag NMODE (input block 1).
At entry IF3COP to module INFCOP data are expected for all strata ISTRA in the range ISTRA = ISTRAA,ISTRAE. There they may be further prepared (e.g. normalized, or scaled to other units) for transfer to the external code

### 4.1.5 entry IF4COP

*post-processing after one complete cycle*: *overall balances, statistics, etc.*

## 4.2 Routines for cycling of EIRENE with external codes: EIRSRT

In this subroutine the "cycling" between EIRENE and external codes is controlled. There are various options, such as "full time dependent (explicit)", "quasi-stationary (explicit)", and "quasi-stationary (implicit)".

## 4.3 Routines for special tallies needed for code coupling: UPTCOP

Updating of code interface tallies COPV, "along the fly". These tallies are specific to the particular background plasma **CFD!** code, hence module UPTCOP is part of the code interfacing module. Apart from scoring along Monte Carlo histories in this special routine (rather than in EIRENE routine UPDATE), the tally COPV is scaled, averaged, printed and plotted as any other EIRENE tally.
In older versions of EIRENE (before 2002) in particular momentum exchange rates (i.e. friction terms) in the direction parallel to the magnetic field have been programmed here in UPTCOP. Meanwhile these have become default tallies (i.e. scoring moved to UPDATE, see section 6.1. Still some particular coupling tallies, e.g. to render the coupling procedure more implicit, are scored in this routine.

## 4.4 Statistical noise in Monte Carlo terms for external code, noise-residuals: STATIS_COP

As pointed out in section 2.9: for all primary (not derived) EIRENE tallies the empirical standard deviation can also be obtained, if requested, by setting proper flags in input block 9.
Furthermore: (since Nov. 2013): the new option UPFCOP, to obtain statistical estimates of linear combination of standard tallies, by evaluating these composite scores after each trajectory from the individual tally scores. This option is restricted to **linear** combinations of tallies, because only then the sum over events and the sum over trajectories commute.
The original considerable CPU penalty for evaluating standard deviations along with the mean values (the tallies themselves) was avoided, at least for volume averaged tallies, in EIRENE versions 99 and younger, by major code optimization (indirect addressing) in these parts. In versions 2012 and younger the same optimization was carried out also for standard deviations of all surface averaged tallies.
In Versions Oct. 2013 and older some standard deviation tallies of special interest for code interfacing (such as total, summed over species, source rates) have been implemented in a special routine in the code interfacing module: STATIS_COP.
This routine STATIS_COP has now been made redundant, because all tallies needed for interfacing, including sums, differences of default tallies, are now available as COPV tallies, either scored in UPTCOP or, for linear combinations, in UPFCOP, and for those COPV tallies standard deviations are available by default. This allows direct and unbiased estimation of variances for source terms of internal (rather than total) energy balances, or, more generally, for plasma fluid equations formulated in the rest frame of the plasma rather than in the laboratory frame. For example the B2.5 version of the B2 2D plasma transport code contains the energy balance equations in the plasma frame, but the momentum balance equations in the laboratory frame. For all combinations direct statistical error estimates of EIRENE sources are now available.

Version 2012 and older:
Subroutine STATIS_COP provides these estimates for those tallies which are specific to a particular coupled case. Usually these will be source terms for particle, momentum, and energy balance equations, summed over all donor species.
At entry IF4COP (see section 4.1.5 above) these may be further processed. For example, in case of coupling to the B2 plasma fluid code these noise estimates are integrated into global quantities (dimension: 1/time) and represent the contribution of statistical noise to the overall residuals of a B2 run.
These are printed from IF4COP, together with the overall balances of a coupled run (TRCBAL = .**true**.)
Hence: if the B2 estimated residuals are of the same order as these "statistical noise residuals", then a further convergence of the combined code can only be achieved by either increasing the CPU-time for EIRENE.
Otherwise: The coupled B2-EIRENE run has failed to converge to the solution within the statistical noise.
This is also represented by the convergence measure of "saturated residuals".

# Chapter 5

# EIRENE Continuous Integration

The EIRENE Continuous Integration (CI) pipeline ensures developments to EIRENE do not adversely impact the existing EIRENE code base and allows developers to verify their code across a wide range of EIRENE use cases. The pipeline runs through a series of tests and requires that all the tests pass for successful completion. These tests run on a Gitlab Runner https://docs.gitlab.com/runner/ instance that is triggered whenever code is pushed to the repository, they can also be run manually at any time. The set of tests is defined and controlled by the .gitlab-ci.yml file in the EIRENE repository root directory

## 5.1 Overview

The CI pipeline is broken into the following stages. Each stage contains a number of jobs, the stages run in order, however the **needs** keyword is used in many of the jobs to allow jobs in later stages to start before the previous stage has finished as long as the **needs** are met.

1. preparation
2. build:executable
3. run
4. run_mpi
5. test
6. preparation:openmp
7. build openmp_executable
8. run_openmp
9. test_openmp
10. coverage
11. tag
12. deploy
13. on-schedule

### 5.1.1 Pipeline control

It is possible to exclude some stages by setting the EXCLUDE variables to true in order to save time during development. When pushing to either the develop or master branches the EXCLUDE variables are overidden in the workflow section so that all stages will run for these branches. The rules keyword is used in the workflow and also in the individual jobs to enforce this behaviour. At present the available variables are EXCLUDE_SERIAL, EXCLUDE_MPI, EXCLUDE_EMC3, EXCLUDE_OPENMP.
The **rules** keyword allows conditional control of pipeline jobs, each rule is evaluated in order and stops when the first true condition is reached. Detailed documentation of this and other CI commands can be found at https://docs.gitlab.com/ee/ci/yaml/gitlab_ci_yaml.html.

## 5.2 Continuous Integration Stages

### 5.2.1 preparation

The preparation stage contains 2 jobs, build_library and prepare_testcases. The build library stage creates a buildRelease directory then calls cmake and make to build the EIRENE library before uploading the buildRelease and libRelease directories as artefacts. The serial test cases then link to libRelease/libEIRENE.a.
The prepare testcases job retrieves the set of EIRENE test cases from the EIRENE-sample-cases repository, also on the jugit server. Rather than using git clone a new temporary git repository is created and the remote origin set to the url of the repository. git fetch with a depth of 1 is then called using the commit hash in the variable SAMPLE_CASE_VERSION. When changing the sample cases used in the CI it is this variable that must be changed. FETCH_HEAD is then checked out, resulting in a shallow checkout of depth one, avoiding the download of the complete repository. and saving significant time in the CI pipeline. The EIRENE-sample-cases directory is then uploaded as an artefact wit the .git directory excluded.

### 5.2.2 build:executable

In this stage executables are built for each test case. build_executable is an anchor used by the job specification for each sample case. The majority of cases build a standalone executable that is linked to libRelease/libEIRENE.a. However for the ITER cases the jobs use the build_iter_executable anchor and the executable is built directly from the EIRENE source. The build is started from within the makefiles of the ITER cases, which call cmake with the following switches -DEIRENE_INTERFACE=SOLPS-ITER, -DEIRENE_USER-ROUTINES=iter, and -DEIRENE_POSTFIX=_iter. An EIRENE executable is then created in binRelease and called from the ITER sample case directories.

### 5.2.3 run

The run stage executes all of the sample cases with the command make output, which ensures the case has been compiled and runs the executable to produce output. Output is generated in both

fixed format and json format and stored in the directories output_fixed and output_json. The run anchor is used for each of the sample cases.

### 5.2.4 run_mpi

The MPI enabled jobs run in a separate stage that runs each tests sequentially due to local resource limits although this will likely be reviewed. The MPI tests are all based on EMC3 coupled cases that are stored in the serial sample repository. This testing is very limited at present and will likely be extended in the future.

### 5.2.5 test

The test stage uses system diff to compare the output of all of the sample cases with reference output stored in each case with the command make test. This means that the output must match exactly, so that running the tests on differenct machines or operating systems will almost certainly cause the tests to fail. It is therefore necessary to update the sample cases section 5.3 when a new Docker image section 5.4.2 is used.

### 5.2.6 OpenMP stages

The OpenMP stages duplicate the serial stages with _openmp appended to the name. They run after the completion of the serial and MPI tests and compile the tests with OpenMP support using -DOPENMP=ON in the cmake command and use 4 OpenMP threads at execution time. Only a subset of testcases are duplicated. A separate branch of the EIRENE-sample-cases repository is used for the tests, at the time of writing this is eirene_unified_openmp_bullseye. This branch was created from the initial commit, so they can be merged into the original branch containing the serial tests at any time.
The input files for OpenMP are largely the same as for the serial cases apart from the use of the variables NLIDENT and NPTSDEL. NLIDENT is a logical in the first operating mode block of the input file described in section 2.1 and sets the random seed used by each process or thread to the same value, so that the output can be compared. NPTSDEL is not currently described in the input section of the EIRENE manual. It is the sixth entry in the second line of each stratum distribution definition, section 2.7 in the EIRENE manual, section 7 of the input file. This integer parameter sets the number of particles after which the random seed is reset to the value given by NINITL. This means that as long as $NPTS/NPTSDEL$ is an integer product of the number of threads used the serial and threaded cases will use the same initial seed for each block of particles, allowing comparison

#### 5.2.6.1 OpenMP comparison

When comparing output using OpenMP the non-associative nature of multi threaded floating point addition means that small differences are likely to occur in output when compared to serial runs. Because of this it is necessary to compare the relative difference of output and set thresholds on these differences
The compare_eirene.py script compares EIRENE output in 2 directories. table 5.1 shows the available command line options that allow control over the output data to be read as well as levels

| Option | Description |
|---|---|
| -h, –help | Show the help message and exit |
| -d DIR1, –dir1 DIR1 | First directory containing output files (default=output) |
| -r DIR2, –dir2 DIR2 | Second directory containing output files (default=output_ref) |
| -f FILE, –file FILE | If specified only this file will be checked |
| -q, –quiet | Only prints a message if differences exist |
| -b, –binary | Check binary files |
| -v, –verbose | Prints more detail about the comparisons being made |
| -t, –statwarn | Prints out values of all mismatches above machine precision |
| -w, –warnings | Prints out values of all mismatches above statistical error |
| -e, –errall | Prints out all mismatches |
| -s, –single_precision | Sets tolerance to single precision |
| -i, –ignore_small_values | Ignore values that are much smaller than the mean |
| -x, –write_texfile | Write differences to a file in tex tabular format |
| –ignore_factor IGNORE_FACTOR | Set the ratio to the mean at which small values are ignored (default=1000) |

Table 5.1: Command line options for the compare_eirene.py script.

of tolerance and reporting. When called with no arguments the script compares file in the the directories output and output_ref giving a report for each file where numerical differences exist. The source can be found in any of the openmp specific test case directories, see section 5.3.
The script compares the relative difference between all numeric values in the output files to a series of thresholds, see table 5.2. The number of differences at each of level is then reported. The tolerances cannot be changed on the command line and must be changed by hand in the script. The level chosen for Machine Precision is approximate and choice of levels above machine precision is arbitrary however provides a useful representation of differences between output.

| Reporting Level | Double Precision Threshold (Default) | Single Precision Threshold |
|---|---|---|
| Machine Precision | $1^{-15}$ | $1^{-7}$ |
| Statistical Difference | $1^{-14}$ | $1^{-6}$ |
| Warning | $1^{-12}$ | $1^{-5}$ |
| Error | $1^{-3}$ | $1^{-3}$ |

Table 5.2: Thresholds and labels for comparison of the relative difference of EIRENE output in the script compare_eirene.py

#### 5.2.6.2 Exclusion of selected data points

Some of the EIRENE tallies result in zero or very close to zero values after the addition and subtraction of values that are expected to be the identical due to particles changing species in the relevant cell. In practice machine precision limits mean that the 2 values can differ by a small amount and when one is subtracted from the other a remainder is left. When running serially this remainder is in general the same for repeated runs on the same system, as in the CI. However when OpenMP threading is enabled these values can vary due to the change in ordering of the operations, causing problems for the CI testing as the values can be different by up to a factor 2, causing the pipeline to fail when testing these values.
There is no simple solution available to this problem although in practice only data in the fort.10 output file in the ITER_1573_openmp and 2D-D_slab_openmp cases are affected. In order to address this the data points that are affected are excluded from testing by adding a list of points to exclude to the compare_eirene.py script in these directories. The points are selected using the errors generated by in previous runs. Not all of the excluded tallies have been confirmed to be due to machine precision, the tallies that have been identified are represented by the pointers PMML, PAML and PAAT.
It is likely that changes to the code will result in different cells being affected by this problem, in which case the cells will need to be identified and added to the lists for exclusions. However care must be taken to ensure that this process is in fact the cause of the new errors.

### 5.2.7 coverage

The coverage stage only runs when pushing to the develop and master branches and can only be used with GNU compilers. This stage clones EIRENE and the sample cases and compiles with the -ftest-coverage flag to produce a coverage report using the Makefile in the EIRENE-sample-cases root dorectory. The GNU tool gcov is used to asses the level of code coverage provided by the serial sample cases and this is reported through the CI. As of the latest release (MsV-2023Mar-PBoerner-beta) the coverage report fails due to a compiler error, this is being investigated.

### 5.2.8 tag, deploy and on-schedule

There are 3 stages that use the .tag_and_deploy.yml and .update_fork_branches.yml files in the EIRENE root directory. These stages do not have any impact on the normal running of the EIRENE CI so are not discussed here.

## 5.3 Sample cases

The EIRENE CI uses sample cases from the EIRENE_sample-cases repository at
https://jugit.fz-juelich.de/eirene/EIRENE-sample-cases
At present two different branches are used for the tests, one for the serial tests and one for the OpenMP tests. This allows development of either without impacting the other. The branch used for serial tests at the time of writing is eirene_unified_bullseye which contains 24 tests cases covering different geometries and cases. The eirene_unified_openmp_bullseye branch is used for

the OpenMP cases and contains a subset of 7 of the serial test cases. If either of the sample cases are changed it is necessary to change the relevant git reference in the .gitlab-ci.yml file in the root directory of the EIRENE repository.

## 5.4 Development of the Continuous Integration pipeline

### 5.4.1 Linting and validation

Changes to the made to the gitlab-ci.yml file can be validated on the EIRENE gitlab repository web pages. In the Continuous Integration section there is a button in the top right corner labelled CI lint, or alternatively the web page can be accessed directly at
https://jugit.fz-juelich.de/eirene/eirene/-/ci/lint
The contents of the gitlab-ci.yml file can be pasted into this page which will validates the syntax and report any errors. In the case of valid syntax it will produce a table of the commands launched by the CI. For developers with permission to write to the develop or master branch a full CI simulation can also be run by selecting the check box **Simulate pipeline creation for the default branch**.

### 5.4.2 Docker Images

The Docker images used by the EIRENE CI are stored in the repository
git@jugit.fz-juelich.de:eirene/eirene-ci-image.git
This image can be built locally and the test cases run in a docker environment for testing purposes.

#### 5.4.2.1 Using the Docker image

The use of the docker image on Juelich systems is described in the EIRENE wiki
https://jugit.fz-juelich.de/eirene/eirene/-/wikis/EIRENEContinuousIntegration
In order to run the tests cases on a local Linux machine follow the steps described here. It is assumed that docker is installed on the machine and that you have read access to the following repositories.

- Docker image git@jugit.fz-juelich.de:eirene/eirene-ci-image.git
- Sample cases git@jugit.fz-juelich.de:eirene/EIRENE-sample-cases.git
- EIRENE git@jugit.fz-juelich.de:eirene/eirene.git

Create a directory within which one can run the docker container
First build the docker image locally by first cloning the Docker image repository. This will create a directory eirene-ci-image, go into this directory and build the image.

```
$ git clone git@jugit.fz-juelich.de:eirene/eirene-ci-image.git
$ cd eirene-ci-image
$ sudo docker image build . -t debianbullseye:test
```

The debianbullseye:test tags can be changed to any tags you prefer. This may take some time. It is also necessary to clone the EIRENE and EIRENE-sample-cases repositories. This can be done from either the docker container or from your default environment, if you have ssh keys enabled then this may be easier, however there is the potential for conflict between the git version on your system and that used in the docker container. You can run the docker image from a directory of your choice, if cloning the EIRENE repository using your local system it should be cloned into the directory that you will run docker from. The sample cases repository should be cloned into the eirene directory containing the EIRENE source. Assuming you have already cloned the EIRENE and sample cases repositories you should run the following command under sudo.

```
$ sudo docker run --privileged=true -it -v `pwd`:`pwd` \
  -w `pwd` debianbullseye:test
```

If you have not yet cloned the EIRENE and the sample cases it will now need to be done in the running container. Once this is done you can change directory to the appropriate case and run the make command to build the appropriate executable. The make output and make test will run and test the code. It is unlikely that the tests will pass due to differences in the architecture of the local machine. In this case it is necessary to copy the output generated by a first run into the output_ref directory and then carry out the tests.

# Chapter 6

# Default EIRENE tallies, and selected Modules

## 6.1 Tables of EIRENE tallies

The following three tables comprise the EIRENE preprogrammed default "tallies". (Further tallies can be added via the problem specific interfacing or user routines).
The term "tally" is adopted from neutron transport applications, a more precise terminology would be "response", see section 3.2. By abuse of language we also refer to input data, such as plasma profiles defined on the computational grid, as tallies, because in post-processing these can printed, plotted and be algebraically mixed with the output tallies (responses) to form more general output quantities. The three tables given below list the (volumetric) input tallies, (volumetric, cell averaged) output tallies, and the surface averaged output tallies, respectively.
Because of the new type of particle introduced during 2002 (photon gas, ITYP=0) the numbering of tallies has changed significantly. Therefore the tables are given below once for the current EIRENE version (2002 and younger), and once for the versions older than 2001. A further change in tally numbering and naming conventions has been carried out in early 2014, and this affects the surface sputter tallies, (and those surface tallies in the list after the sputter tallies. These latter revisions are described in table 6.5, and only for that section of the surface tally list that has been affected.
Options for switching on and off of input tallies have been added to the code in 2019. At the same time the input data has been reviewed and additional input tallies have been made available. For this reason the table of input tallies is given twice once for the current EIRENE version (Eirene_revised_input_tallies) and once for previous versions.

Each **volume averaged response** is a spatial function, averaged over a grid cell, hence: a piecewise constant function, which is constant in each grid cell ICELL:
ICELL=1,NRAD.
Default volumetric particle, energy and momentum source tallies are split by the **type** (but not the individual species index of the incident test particle) involved in a collision process: **Pht!** (**Pht!**), **Ats!** (**Ats!**), **Mls!** (**Mls!**), **TI!** (**TI!**). This permits scaling of these sources with the factors FPHOT, FATM, FMOL, FION, to eliminate statistical errors from the particle balances (see NLSCL option, input block 1). They are further split by both: the type (also: electrons and the species index of the "receiving" component, for which the source/sink term would arise in a velocity space averaged balance equation (with the sign as specified). Any further tallies (e.g. by incident species, or particular collision process resolved, or energy (spectrally) resolved, etc.)

have to be implemented in the 3rd party (user) or the problem specific interfacing routines.
Tally naming convention:

**Particle source rates** name: P[A][BC], with

A: type of incident particle, A = A (atoms), M (molecules), I (test ions), PH (photons), P (bulk particle)

BC: type of produced particle, BC = AT (atoms), ML (molecules), IO (test ions), PHT (photons), EL (electrons), PL (bulk particles)

The tallies are resolved by species index of produced particle, and are to be scaled (NLSCL option) by type of incident test particle.

**Energy source rates** name: E[A][BC], analogue to particle source rates

but not resolved wrt. species index of secondary particles.

**Momentum source rates** name: M[A][BC], analogue to particle source rates

only available for [A] = A, M, I, PH and for [BC]=PL, i.e. currently default tallies are only available for momentum sources/sinks for background (plasma) particles caused by test particles, resolved by type of test particle (name of tally) and by species of bulk particle (1st index in array).

**Surface averaged responses** are also spatial functions, averaged over surface segments, hence: piecewise constant on each surface ISURF or surface segment:
ISURF=1,NLIMI.
and with some spatial resolution possible on standard mesh surfaces
ISURF=NLIM,NLIM+NSTSI
Emitted particle and energy surface averaged fluxes are split by the type of the incident particle: **Pht!** (**Pht!**), **Ats!** (**Ats!**), **Mls!** (**Mls!**), **TI!** (**TI!**) or **BI!** (**BI!**). This permits scaling of these fluxes with the factors FPHOT, FATM, FMOL, FION, to eliminate statistical errors from the particle balances (see NLSCL option, input block 1).

Naming conventions for emitted surface fluxes are analogue to those for volumetric source rates, i.e. PRF[A][BC], SPT[A][BC] with A and BC having the same meaning as described above for volume tallies.
PRF... are fluxes reflected or emitted due to incident particle fluxes, of type [A], SPT[A][BC] are sputtered fluxes (additional fluxes of surface material).
**Note:** The naming convention of **surface sputter tallies SPT...** differs in older versions of EIRENE. The full resolution wrt. to incident type and emitted species is only implemented in EIRENE revisions SVN 360 and later (Jan. 2014), see table 6.5.
In all older versions only much fewer sputter tallies SPT[BC] were available, i.e. only one type-identifier in the name of these "sputter tallies".
And [BC] (and the corresponding species index) could either mean:

i) resolution wrt. incident flux species (summing over all sputtered flux species and types) (same as in incident surface flux tallies), or

ii) resolution wrt. sputtered flux (summing over all incident species and types),

depending on code version. Type ii) convention was used in EIRENE versions operated at ITER-IO (e.g. in SOLPS4.3), all other EIRENE versions had convention i).
The extension of Jan. 2014 now combines i) and ii) by providing the resolution wrt. emitted type and species (as ii) did) and simultaneously keeping the resolution wrt. incident type (i), as needed for proper scaling of particle balances (see option NLSCL, input block 1).

If temporal resolution has been requested (input block 13), then NSTSI is increased by one: NSTSIP=NSTSI+1, and the last tally I2=NLIM+NSTSIP corresponds to the "Time-Surface" (the census array) then.

### 6.1.1 Current status, incl. photon gas tallies (Eirene$_{2002}$ and younger)

Table 6.1: **Input Tallies for Background, Input, Module: COMUSR, EIRENE version Eirene_revised_input_tallies.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. | switched |
|---|---|---|---|---|---|---|
| -1 | TEIN | Plasma Temperature, Electrons | 1 | eV | / | on |
| -2 | TIIN | Plasma Temperature, Bulk Ions | NPLSTI | eV | / | on |
| -3 | DEIN* | Plasma Density, Electrons | 1 | $cm^{-3}$ | / | on |
| -4 | DIIN | Plasma Density, Bulk Ions | NPLS | $cm^{-3}$ | / | on |
| -5 | VXIN | Plasma Drift Velocity, x-component, Bulk Ions | NPLSV | $cm\,s^{-1}$ | / | NPLSV ¿ 0on |
| -6 | VYIN | Plasma Drift Velocity, y-component, Bulk Ions | NPLSV | $cm\,s^{-1}$ | / | NPLSV ¿ 0 on |
| -7 | VZIN | Plasma Drift Velocity, z-component, Bulk Ions | NPLSV | $cm\,s^{-1}$ | / | on |
| -8 | BXIN | Magnetic field unit vector, x-component | 1 | / | / | on |
| -9 | BYIN | Magnetic field unit vector, y-component | 1 | / | / | on |
| -10 | BZIN | Magnetic field unit vector, z-component | 1 | / | / | on |
| -11 | BFIN | Magnetic field strength | 1 | T | / | on |
| -12 | ADIN | Additional input tally | NAIN | see INFCOP | / | NAIN ¿ 0 |
| -13 | EDRIFT* | Kinetic energy in drift motion, Bulk Ions | NPLS | eV | / | on |
| -14 | VOL | Zone Volume | 1 | $cm^3$ | / | on |
| -15 | WGHT ** | unused, importance fct. | NSPCMC | 1 | / | off |
| -16 | BXPERP * | PERP. MAGN. FIELD VECTOR, X DIRECTION | 1 | | / | off |
| -17 | BYPERP * | PERP. MAGN. FIELD VECTOR, Y DIRECTION | 1 | | / | off |
| -18 | EXIN | electric field unit vector, x component | 1 | | / | off |
| -19 | EYIN | electric field unit vector, y component | 1 | | / | off |
| -20 | EZIN | electric field unit vector, z component | 1 | | / | off |
| -21 | EFIN | electric field strength | 1 | $V\,cm^{-1}$ | / | off |
| -22 | POT | electric potential | 1 | V | / | off |
| -23 | BVIN * | flow velocity parallel to B field | NPLSV | $cm\,s^{-1}$ | / | NPLSV ¿ 0 |
| -24 | PARMOM * | parallel to B flow momentum | NPLS | $g\,cm\,s^{-1}$ | / | NPLS ¿ 0 |
| -25 | PSI | PSI fct | 1 | – | / | off |
| -31 | dTE/dX * | partial derivative of TE in X direction | 1 | $eV\,cm^{-1}$ | / | off |
| -32 | dTE/dY * | partial derivative of TE in Y direction | 1 | $eV\,cm^{-1}$ | / | off |

Table 6.1: **Input Tallies for Background, Input, Module: COMUSR, EIRENE version Eirene_revised_input_tallies.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. | switched |
|---|---|---|---|---|---|---|
| -33 | dTE/dZ * | partial derivative of TE in Z direction | 1 | eV $cm^{-1}$ | / | off |
| -34 | dTI/dX * | partial derivative of TI in X direction | 1 | eV $cm^{-1}$ | / | off |
| -35 | dTI/dY * | partial derivative of TI in Y direction | 1 | eV $cm^{-1}$ | / | off |
| -36 | dTI/dZ * | partial derivative of TI in Z direction | 1 | eV $cm^{-1}$ | / | off |
| -37 | dNE/dX * | partial derivative of NE in X direction | 1 | $cm^{-3}$ $cm^{-1}$ | / | off |
| -38 | dNE/dY * | partial derivative of NE in Y direction | 1 | $cm^{-3}$ $cm^{-1}$ | / | off |
| -39 | dNE/dZ * | partial derivative of NE in Z direction | 1 | $cm^{-3}$ $cm^{-1}$ | / | off |
| -40 | dNI/dX * | partial derivative of NI in X direction | 1 | $cm^{-3}$ $cm^{-1}$ | / | off |
| -41 | dNI/dY * | partial derivative of NI in Y direction | 1 | $cm^{-3}$ $cm^{-1}$ | / | off |
| -42 | dNI/dZ * | partial derivative of NI in Z direction | 1 | $cm^{-3}$ $cm^{-1}$ | / | off |
| -43 | dVX/dX * | partial derivative of VX in X direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -44 | dVX/dY * | partial derivative of VX in Y direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -45 | dVX/dZ * | partial derivative of VX in Z direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -46 | dVY/dX * | partial derivative of VY in X direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -47 | dVY/dY * | partial derivative of VY in Y direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -48 | dVY/dZ * | partial derivative of VY in Z direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -49 | dVZ/dX * | partial derivative of VZ in X direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -50 | dVZ/dY * | partial derivative of VZ in Y direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -51 | dVZ/dZ * | partial derivative of VZ in Z direction | 1 | cm $s^{-1}$ $cm^{-1}$ | / | off |
| -52 | dBX/dX * | partial derivative of BX in X direction | 1 | / | / | off |
| -53 | dBX/dY * | partial derivative of BX in Y direction | 1 | / | / | off |
| -54 | dBX/dZ * | partial derivative of BX in Z direction | 1 | / | / | off |
| -55 | dBY/dX * | partial derivative of BY in X direction | 1 | / | / | off |
| -56 | dBY/dY * | partial derivative of BY in Y direction | 1 | / | / | off |
| -57 | dBY/dZ * | partial derivative of BY in Z direction | 1 | / | / | off |
| -58 | dBZ/dX * | partial derivative of BZ in X direction | 1 | / | / | off |
| -59 | dBZ/dY * | partial derivative of BZ in Y direction | 1 | / | / | off |

Table 6.1: **Input Tallies for Background, Input, Module: COMUSR, EIRENE version Eirene_revised_input_tallies.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. | switched |
|---|---|---|---|---|---|---|
| -60 | dBZ/dZ * | partial derivative of BZ in Z direction | 1 | / | / | off |
| -61 | dBF/dX * | partial derivative of BF in X direction | 1 | T $cm^{-1}$ | / | off |
| -62 | dBF/dY * | partial derivative of BF in Y direction | 1 | T $cm^{-1}$ | / | off |
| -63 | dBF/dZ * | partial derivative of BF in Z direction | 1 | T $cm^{-1}$ | / | off |
| -64 | dADIN/dX * | partial derivative of ADIN in X direction | 1 | / | / | off |
| -65 | dADIN/dY * | partial derivative of ADIN in Y direction | 1 | / | / | off |
| -66 | dADIN/dZ * | partial derivative of ADIN in Z direction | 1 | / | / | off |
| -67 | dEDRIFT/dX * | partial derivative of EDRIFT in X direction | 1 | eV $cm^{-1}$ | / | off |
| -68 | dEDRIFT/dY * | partial derivative of EDRIFT in Y direction | 1 | eV $cm^{-1}$ | / | off |
| -69 | dEDRIFT/dZ * | partial derivative of EDRIFT in Z direction | 1 | eV $cm^{-1}$ | / | off |
| -70 | dVOL/dX * | partial derivative of VOL in X direction | 1 | $cm^3$ $cm^{-1}$ | / | off |
| -71 | dVOL/dY * | partial derivative of VOL in Y direction | 1 | $cm^3$ $cm^{-1}$ | / | off |
| -72 | dVOL/dZ * | partial derivative of VOL in Z direction | 1 | $cm^3$ $cm^{-1}$ | / | off |
| -73 | dWGHT/dX * | partial derivative of WGHT in X direction | 1 | $cm^{-1}$ | / | off |
| -74 | dWGHT/dY * | partial derivative of WGHT in Y direction | 1 | $cm^{-1}$ | / | off |
| -75 | dWGHT/dZ * | partial derivative of WGHT in Z direction | 1 | $cm^{-1}$ | / | off |
| -76 | dBXPERP/dX * | partial derivative of BXPERP in X direction | 1 | | / | off |
| -77 | dBXPERP/dY * | partial derivative of BXPERP in Y direction | 1 | | / | off |
| -78 | dBXPERP/dZ * | partial derivative of BXPERP in Z direction | 1 | | / | off |
| -79 | dBYPERP/dX * | partial derivative of BYPERP in X direction | 1 | | / | off |
| -80 | dBYPERP/dY * | partial derivative of BYPERP in Y direction | 1 | | / | off |
| -81 | dBYPERP/dZ * | partial derivative of BYPERP in Z direction | 1 | | / | off |
| -82 | dEX/dX * | partial derivative of EX in X direction | 1 | | / | off |
| -83 | dEX/dY * | partial derivative of EX in Y direction | 1 | | / | off |
| -84 | dEX/dZ * | partial derivative of EX in Z direction | 1 | | / | off |
| -85 | dEY/dX * | partial derivative of EY in X direction | 1 | | / | off |
| -86 | dEY/dY * | partial derivative of EY in Y direction | 1 | | / | off |

Table 6.1: **Input Tallies for Background, Input, Module: COMUSR, EIRENE version Eirene_revised_input_tallies.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. | switched |
|---|---|---|---|---|---|---|
| -87 | dEY/dZ * | partial derivative of EY in Z direction | 1 | | / | off |
| -88 | dEZ/dX * | partial derivative of EZ in X direction | 1 | | / | off |
| -89 | dEZ/dY * | partial derivative of EZ in Y direction | 1 | | / | off |
| -90 | dEZ/dZ * | partial derivative of EZ in Z direction | 1 | | / | off |
| -91 | dEF/dX * | partial derivative of EF in X direction | 1 | $V\,cm^{-1}\,cm^{-1}$ | / | off |
| -92 | dEF/dY * | partial derivative of EF in Y direction | 1 | $V\,cm^{-1}\,cm^{-1}$ | / | off |
| -93 | dEF/dZ * | partial derivative of EF in Z direction | 1 | $V\,cm^{-1}\,cm^{-1}$ | / | off |
| -94 | dPOT/dX * | partial derivative of POT in X direction | 1 | $V\,cm^{-1}$ | / | off |
| -95 | dPOT/dY * | partial derivative of POT in Y direction | 1 | $V\,cm^{-1}$ | / | off |
| -96 | dPOT/dZ * | partial derivative of POT in Z direction | 1 | $V\,cm^{-1}$ | / | off |
| -97 | dBV/dX * | partial derivative of BV in X direction | 1 | $cm\,s\,cm^{-1}$ | / | off |
| -98 | dBV/dY * | partial derivative of BV in Y direction | 1 | $cm\,s\,cm^{-1}$ | / | off |
| -99 | dBV/dZ * | partial derivative of BV in Z direction | 1 | $cm\,s\,cm^{-1}$ | / | off |
| -100 | dPARMOM/dX * | partial derivative of PARMOM in X direction | 1 | $g\,cm\,s^{-1}\,cm^{-1}$ | / | off |
| -101 | dPARMOM/dY * | partial derivative of PARMOM in Y direction | 1 | $g\,cm\,s^{-1}\,cm^{-1}$ | / | off |
| -102 | dPARMOM/dZ * | partial derivative of PARMOM in Z direction | 1 | $g\,cm\,s^{-1}\,cm^{-1}$ | / | off |
| -103 | dPSI/dX * | partial derivative of PSI in X direction | 1 | / | / | off |
| -104 | dPSI/dY * | partial derivative of PSI in Y direction | 1 | / | / | off |
| -105 | dPSI/dZ * | partial derivative of PSI in Z direction | 1 | / | / | off |

**Note:**
We distinguish between so called primary input tallies, and derived input tallies, the latter being indicated by *. The latter are automatically computed from the former by EIRENE.
Primary input tallies are set or transferred from external data sources via subr. PLASMA.f, derived input tallies are set via subroutine PLASMA_DERIV.f. Storage for both kinds of tallies is allocated in module COMUSR.f

- NPLSTI: number of different bulk particle (bulk ion) temperatures in a run. See input block 5. Default: NPLSTI=NPLS.
- NPLSV: number of different bulk particle (bulk ion) flow velocities in a run. See input block 5. Default: NPLSV=NPLS.

- Some input tallies such as electron and ion temperatures and densities as well as drift velocities and magnetic field are not allowed to be switched off.
- DEIN and EDRIFT are “derived” input tallies, internally computed from the ion densities (quasi-neutrality), and the flow field, respectively.
- Tallies -26 – -30 are intentionally left free for future use and are switched off per default.
- Tallies -31 – -120 are further “internal” derived quantities, providing derivatives in carthesian coordinates (gradient vector) of input tallies -1 – -25 and are switched off if not explicitely switched on in the input block 5 of the EIRENE input file (section 2.5).
- All input tallies are available for printing, plotting and for use in algebraic expressions in volume averaged tallies when switched on.
- B-field and E-field tallies contain redundancy. Storage can be reduced.
- WGHT space species weight window arrays. Currently not in use and switched off.
- Magnetic field input is needed only for photon transport with Zeeman split line shapes, or in case of charged particle transport (Subr. FOLION). Only quite rudimentary preprogrammed input options for magnetic field are available, usually it is expected to receive B-field information from external files, see INDPRO(5) options for magnetic field tallies in input block 5 (section 2.5).
- For electric fields (needed only for charged particle transport) only input via external files is possible. Default: $\vec{E} = 0$.



Table 6.2: **Input Tallies for Background, Input, Module: COMUSR, EIRENE version prior to Eirene_revised_input_tallies.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| -1 | TEIN | Plasma Temperature, Electrons | 1 | eV | / |
| -2 | TIIN | Plasma Temperature, Bulk Ions | NPLSTI | eV | / |
| -3 | DEIN* | Plasma Density, Electrons | 1 | $cm^{-3}$ | / |
| -4 | DIIN | Plasma Density, Bulk Ions | NPLS | $cm^{-3}$ | / |
| -5 | VXIN | Plasma Drift Velocity, x-component, Bulk Ions | NPLSV | $cm\,s^{-1}$ | / |
| -6 | VYIN | Plasma Drift Velocity, y-component, Bulk Ions | NPLSV | $cm\,s^{-1}$ | / |
| -7 | VZIN | Plasma Drift Velocity, z-component, Bulk Ions | NPLSV | $cm\,s^{-1}$ | / |

Table 6.2: **Input Tallies for Background, Input, Module: COMUSR, EIRENE version prior to Eirene_revised_input_tallies.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| -8 | BXIN | Magnetic field unit vector, x-component | 1 | / | / |
| -9 | BYIN | Magnetic field unit vector, y-component | 1 | / | / |
| -10 | BZIN | Magnetic field unit vector, z-component | 1 | / | / |
| -11 | BFIN | Magnetic field strength | 1 | T | / |
| -12 | ADIN | Additional input tally | NAIN | see INFCOP | / |
| -13 | EDRIFT* | Kinetic energy in drift motion, Bulk Ions | NPLS | eV | / |
| -14 | VOL | Zone Volume | 1 | $cm^3$ | / |
| -15 | WGHT ** | unused, importance fct. | NSPCMC | 1 | / |
| -16 | BXPERP * | unused | 1 | | / |
| -17 | BYPERP * | unused | 1 | | / |
| -18 | EXIN | electric field unit vector, x component | 1 | | / |
| -19 | EYIN | electric field unit vector, y component | 1 | | / |
| -20 | EZIN | electric field unit vector, z component | 1 | | / |
| -21 | EFIN | electric field strength | 1 | $V\,cm^{-1}$ | / |
| -22 | POT | electric potential | 1 | V | / |
| d1 | BVIN * | flow velocity parallel to B field | NPLSV | $cm\,s^{-1}$ | / |
| d2 | PARMOM * | parallel to B flow momentum | NPLS | $g\,cm\,s^{-1}$ | / |
| d3 | TEDTEDX ** | unused | 1 | – | / |
| d4 | TEDTEDY ** | unused | 1 | – | / |
| d5 | TEDTEDZ ** | unused | 1 | – | / |

**Note:**
We distinguish between so called primary input tallies, and derived input tallies, the latter being indicated by *. The latter are automatically computed from the former by EIRENE.
Primary input tallies are set or transferred from external data sources via subr. PLASMA.f, derived input tallies are set via subroutine PLASMA_DERIV.f. Storage for both kinds of tallies is allocated in module COMUSR.f

- NPLSTI: number of different bulk particle (bulk ion) temperatures in a run. See input block 5. Default: NPLSTI=NPLS

- NPLSV: number of different bulk particle (bulk ion) flow velocities in a run. See input block 5. Default: NPLSV=NPLS
- DEIN and EDRIFT are “derived” input tallies, internally computed from the ion densities (quasi-neutrality), and the flow field, respectively.
- Tallies d1 – d5 are further “internal” derived quantities, currently not selectable for printout nor for graphics.
- B-field and E-field tallies contain redundancy. Storage can be reduced
- WGHT space species weight window arrays. Currently not in use
- Magnetic field input is needed only for photon transport with Zeeman split line shapes, or in case of charged particle transport (Subr. FOLION). Only quite rudimentary preprogrammed input options for magnetic field are available, usually it is expected to receive B-field information from external files, see INDPRO(5) options for magnetic field tallies in input block 5 (section 2.5).
- For electric fields (needed only for charged particle transport) only input via external files is possible. Default: $\vec{E} = 0$

Table 6.3: **Volume Averaged Tallies, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 1 | PDENA | Particle density, Atoms | NATM | $cm^{-3}$ | TD |
| 2 | PDENM | Particle density, Molecules | NMOL | $cm^{-3}$ | TD |
| 3 | PDENI | Particle density, Test Ions | NION | $cm^{-3}$ | TD |
| 4 | PDENPH | Particle density, Photons | NPHOT | $cm^{-3}$ | TD |
| 5 | EDENA | Energy density, Atoms | NATM | eV $cm^{-3}$ | TE |
| 6 | EDENM | Energy density, Molecules | NMOL | eV $cm^{-3}$ | TE |
| 7 | EDENI | Energy density, Test Ions | NION | eV $cm^{-3}$ | TE |
| 8 | EDENPH | Energy density, Photons | NPHOT | eV $cm^{-3}$ | TE |
| 9 | PAEL | Particle Source (Electrons) from atom-plasma coll. | 1 | amp $cm^{-3}$ | TPS |
| 10 | PAAT | Particle Source (Atoms) from atom-plasma coll. | NATM | amp $cm^{-3}$ | TPS |
| 11 | PAML | Particle Source (Molecules) from atom-plasma coll. | NMOL | amp $cm^{-3}$ | TPS |
| 12 | PAIO | Particle Source (Test Ions) from atom-plasma coll. | NION | amp $cm^{-3}$ | TPS |
| 13 | PAPHT | Particle Source (Photons) from atom-plasma coll. | NPHOT | amp $cm^{-3}$ | TPS |
| 14 | PAPL | Particle Source (Bulk Ions) from atom-plasma coll. | NPLS | amp $cm^{-3}$ | TPS |
| 15 | PMEL | Particle Source (Electrons) from molecule-plasma coll. | 1 | amp $cm^{-3}$ | TPS |
| 16 | PMAT | Particle Source (Atoms) from molecule-plasma coll. | NATM | amp $cm^{-3}$ | TPS |
| 17 | PMML | Particle Source (Molecules) from molecule-plasma coll. | NMOL | amp $cm^{-3}$ | TPS |
| 18 | PMIO | Particle Source (Test Ions) from molecule-plasma coll. | NION | amp $cm^{-3}$ | TPS |
| 19 | PMPHT | Particle Source (Photons) from molecule-plasma coll. | NPHOT | amp $cm^{-3}$ | TPS |
| 20 | PMPL | Particle Source (Bulk Ions) from molecule-plasma coll. | NPLS | amp $cm^{-3}$ | TPS |
| 21 | PIEL | Particle Source (Electrons) from test ion-plasma coll. | 1 | amp $cm^{-3}$ | TPS |
| 22 | PIAT | Particle Source (Atoms) from test ion-plasma coll. | NATM | amp $cm^{-3}$ | TPS |
| 23 | PIML | Particle Source (Molecules) from test ion-plasma coll. | NMOL | amp $cm^{-3}$ | TPS |
| 24 | PIIO | Particle Source (Test Ions) from test ion-plasma coll. | NION | amp $cm^{-3}$ | TPS |
| 25 | PIPHT | Particle Source (Photons) from test ion-plasma coll. | NPHOT | amp $cm^{-3}$ | TPS |
| 26 | PIPL | Particle Source (Bulk Ions) from test ion-plasma coll. | NPLS | amp $cm^{-3}$ | TPS |
| 27 | PPHEL | Particle Source (Electrons) from photon-plasma coll. | 1 | amp $cm^{-3}$ | TPS |

Table 6.3: **Volume Averaged Tallies, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 28 | PPHAT | Particle Source (Atoms) from photon-plasma coll. | NATM | amp $cm^{-3}$ | TPS |
| 29 | PPHML | Particle Source (Molecules) from photon-plasma coll. | NMOL | amp $cm^{-3}$ | TPS |
| 30 | PPHIO | Particle Source (Test Ions) from photon-plasma coll. | NION | amp $cm^{-3}$ | TPS |
| 31 | PPHPHT | Particle Source (Photons) from photon-plasma coll. | NPHOT | amp $cm^{-3}$ | TPS |
| 32 | PPHPL | Particle Source (Bulk Ions) from photon-plasma coll. | NPLS | amp $cm^{-3}$ | TPS |
| 33 | EAEL | Energy Source (Electrons) from atom-plasma coll. | 1 | watt $cm^{-3}$ | T22 |
| 34 | EAAT | Energy Source (Atoms) from atom-plasma coll. | 1 | watt $cm^{-3}$ | T23 |
| 35 | EAML | Energy Source (Molecules) from atom-plasma coll. | 1 | watt $cm^{-3}$ | T24 |
| 36 | EAIO | Energy Source (Test Ions) from atom-plasma coll. | 1 | watt $cm^{-3}$ | T25 |
| 37 | EAPHT | Energy Source (Photons) from atom-plasma coll. | 1 | watt $cm^{-3}$ | T25 |
| 38 | EAPL | Energy Source (Bulk Ions) from atom-plasma coll. | 1 | watt $cm^{-3}$ | T26 |
| 39 | EMEL | Energy Source (Electrons) from molecule-plasma coll. | 1 | watt $cm^{-3}$ | T27 |
| 40 | EMAT | Energy Source (Atoms) from molecule-plasma coll. | 1 | watt $cm^{-3}$ | T28 |
| 41 | EMML | Energy Source (Molecules) from molecule-plasma coll. | 1 | watt $cm^{-3}$ | T29 |
| 42 | EMIO | Energy Source (Test Ions) from molecule-plasma coll. | 1 | watt $cm^{-3}$ | T30 |
| 43 | EMPHT | Energy Source (Photons) from molecule-plasma coll. | 1 | watt $cm^{-3}$ | T30 |
| 44 | EMPL | Energy Source (Bulk Ions) from molecule-plasma coll. | 1 | watt $cm^{-3}$ | T31 |
| 45 | EIEL | Energy Source (Electrons) from test ion-plasma coll. | 1 | watt $cm^{-3}$ | T32 |
| 46 | EIAT | Energy Source (Atoms) from test ion-plasma coll. | 1 | watt $cm^{-3}$ | T33 |
| 47 | EIML | Energy Source (Molecules) from test ion-plasma coll. | 1 | watt $cm^{-3}$ | T34 |
| 48 | EIIO | Energy Source (Test Ions) from test ion-plasma coll. | 1 | watt $cm^{-3}$ | T35 |
| 49 | EIPHT | Energy Source (Photons) from test ion-plasma coll. | 1 | watt $cm^{-3}$ | T35 |
| 50 | EIPL | Energy Source (Bulk Ions) from test ion-plasma coll. | 1 | watt $cm^{-3}$ | T36 |
| 51 | EPHEL | Energy Source (Electrons) from photon-plasma coll. | 1 | watt $cm^{-3}$ | T33 |
| 52 | EPHAT | Energy Source (Atoms) from photon-plasma coll. | 1 | watt $cm^{-3}$ | T33 |
| 53 | EPHML | Energy Source (Molecules) from photon-plasma coll. | 1 | watt $cm^{-3}$ | T34 |
| 54 | EPHIO | Energy Source (Test Ions) from photon-plasma coll. | 1 | watt $cm^{-3}$ | T35 |

Table 6.3: **Volume Averaged Tallies, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 55 | EPHPHT | Energy Source (Photons) from photon-plasma coll. | 1 | watt cm$^{-3}$ | T35 |
| 56 | EPHPL | Energy Source (Bulk Ions) from photon-plasma coll. | 1 | watt cm$^{-3}$ | T36 |
| 57 | ADDV | Additional volume av. Tally, Track-length estimated | NADV | see UPTUSR | TRL |
| 58 | COLV | Additional volume av. Tally, Collision estimated | NCLV | see UPCUSR | COL |
| 59 | SNPV | Additional volume av. Tally, Snapshot estimated | NSNV | see UPSUSR | SNP |
| 60 | COPV | Tallies for coupling to ext. code, see: Subr. UPTCOP | NCPV | see UPTCOP | COP |
| 61 | BGKV | Volume averaged tallies for **BGK!**-self-collision terms | NBGV | see BGK | T41 |
| 62 | ALGV | Algebraic expression in volume averaged tallies | NALV | Input | ALG |
| 63 | PGENA | Generation limit, Atoms | NATM | amp cm$^{-3}$ | CG |
| 64 | PGENM | Generation limit, Molecules | NMOL | amp cm$^{-3}$ | CG |
| 65 | PGENI | Generation limit, Test ions | NION | amp cm$^{-3}$ | CG |
| 66 | PGENPH | Generation limit, Photons | NPHOT | amp cm$^{-3}$ | CG |
| 67 | EGENA | ditto, Energy flux, Atoms | NATM | watt cm$^{-3}$ | CG |
| 68 | EGENM | ditto, Energy flux, Molecules | NMOL | watt cm$^{-3}$ | CG |
| 69 | EGENI | ditto, Energy flux, Test ions | NION | watt cm$^{-3}$ | CG |
| 70 | EGENPH | ditto, Energy flux, Photons | NPHOT | watt cm$^{-3}$ | CG |
| 71 | VGENA | ditto, momentum flux, Atoms | NATM | as MAPL | CG |
| 72 | VGENM | ditto, momentum flux, Molecules | NMOL | as MMPL | CG |
| 73 | VGENI | ditto, momentum flux, Test ions | NION | as MIPL | CG |
| 74 | VGENPH | ditto, momentum flux, Photons | NPHOT | as MPHPL | CG |
| 75 | PPAT | primary particle sources rate (Atoms) from plasma | NATM | amp cm$^{-3}$ | C1 |
| 76 | PPML | primary particle sources rate (Molecules) | NMOL | amp cm$^{-3}$ | C1 |
| 77 | PPIO | primary particle sources rate (Test ions) | NION | amp cm$^{-3}$ | C1 |
| 78 | PPPHT | primary particle sources rate (Photons) | NPHOT | amp cm$^{-3}$ | C1 |
| 79 | PPPL | primary particle sources rate (Bulk Ions) | NPLS | amp cm$^{-3}$ | C1 |
| 80 | EPAT | primary energy sources rate, Atoms | 1 | watt cm$^{-3}$ | C1 |
| 81 | EPML | primary energy sources rate, Molecules | 1 | watt cm$^{-3}$ | C1 |

Table 6.3: **Volume Averaged Tallies, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 82 | EPIO | primary energy sources rate, Test ions | 1 | watt $cm^{-3}$ | C1 |
| 83 | EPPHT | primary energy sources rate, Photons | 1 | watt $cm^{-3}$ | C1 |
| 84 | EPPL | primary energy sources rate, Bulk Ions | 1 | watt $cm^{-3}$ | C1 |
| 85 | VXDENA | momentum density, x-direction, Atoms | NATM | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 86 | VXDENM | momentum density, x-direction, Molecules | NMOL | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 87 | VXDENI | momentum density, x-direction, Test Ions | NION | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 88 | VXDENPH | momentum density, x-direction, Photons | NPHOT | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 89 | VYDENA | momentum density, y-direction, Atoms | NATM | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 90 | VYDENM | momentum density, y-direction, Molecules | NMOL | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 91 | VYDENI | momentum density, y-direction, Test Ions | NION | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 92 | VYDENPL | momentum density, y-direction, Photons | NPHOT | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 93 | VZDENA | momentum density, z-direction, Atoms | NATM | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 94 | VZDENM | momentum density, z-direction, Molecules | NMOL | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 95 | VZDENI | momentum density, z-direction, Test Ions | NION | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 96 | VZDENPL | momentum density, z-direction, Photons | NPHOT | g cm $s^{-1}$ $cm^{-3}$ | T1 |
| 97 | MAPL | parallel momentum source rate (Bulk Ions) | NPLS | amp g cm $s^{-1}$ $cm^{-3}$ | TPM |
| 98 | MMPL | parallel momentum source rate, (Bulk Ions) | NPLS | amp g cm $s^{-1}$ $cm^{-3}$ | TPM |
| 99 | MIPL | parallel momentum source rate, (Bulk Ions) | NPLS | amp g cm $s^{-1}$ $cm^{-3}$ | TPM |
| 100 | MPHPL | parallel momentum sources rate, (Bulk Ions) | NPLS | amp g cm $s^{-1}$ $cm^{-3}$ | TPM |

**Note:**
*Momentum density* is the directional momentum weighted velocity space integral, i.e. it is a (directional) particle flux (density), for the x,y and z directions, respectively, and then multiplied by the particle mass. (Relation to particle current? Check with neutron transport terminology.)
*Source* means: if the sign is positive, it is a gain for the specified type of particles; if it is negative, it is a loss. The specified "type of particle" is coded by the last two letters of the name of a tally, AT, ML, IO, EL, PL, PH, respectively, as well as by the corresponding "1st dimension" (species) NATM, NMOL, NION, NPHOT, NPLS and "1" (one) for electrons. E.g. MAPL: parallel momentum source/sink for bulk particle (plasma background), due to atom plasma interactions. The parallel direction is determined by the magnetic field unit vector

(input tallies BXIN, . . . ).

**Estimators**

EIRENE resorts, by default, to track-length estimators (1.20). Default tallies are updated ("scoring") in routine UPDATE. All default estimators are constant within a cell $m$, i.e. depend only upon the cell index $m$ but not on the position $\mathbf{r}$ in the cell: $g_t(s) = const(m)$. Hence they also do not depend on the position $s$ along the track within a cell. UPDATE calls templates UPTUSR, in which any further quantity can be scored by programming the path integral of any function $g_t(s)$, see section 3.2.
In some instances still collision estimators (1.18) are employed but we are gradually removing them (and plan keep them in the code only to provide independent checks for code-verification runs.)

**TD** Track-length, particle density. $g_t = 1/v$, $v = VEL$ the test particle's velocity. Note that the path-integral of $g_t$ along a trajectory in a cell is equal to the time spend by that history in a cell.

**TE** Track-length, (kinetic) energy density. In case of photon tallies, the energy used is $E0 = h\nu$

**TPS** Track-length, particle source rate

**TPM** Track-length, parallel (to B-field) momentum source/sink. The (vectorial) momentum exchange rate $\Delta\vec{M}om$ is projected to the B-field (input volume tallies Bxin, Byin, Bzin). I.e. if $e_{\vec{B}}$ is the unit vector given by input tallies Bxin, Byin, Bzin, then the scalar product $\langle e_{\vec{B}}, \Delta\vec{M}om\rangle$ is used for scoring.

**CG** Collision estimator for generation limit (for fluid limit) tallies. These tallies contain particle, momentum and energy fluxes (per cell or per $cm^3$), when particles are removed from the fully kinetic tracing, due to e.g. transition into a diffusive or other continuum regime. These fluxes are scored in COLLIDE.F, i.e. by collision estimators by nature.

Table 6.4: **Surface Averaged Tallies, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 1 | POTAT | Particle Flux, incident, Atoms | NATM | amp | T/C |
| 2 | PRFAAT | Particle Flux, emitted, **Ats!**⇒Atoms | NATM | amp | T/C |
| 3 | PRFMAT | Particle Flux, emitted, **Mls!**⇒Atoms | NATM | amp | T/C |
| 4 | PRFIAT | Particle Flux, emitted, **TI!**⇒Atoms | NATM | amp | T/C |
| 5 | PRFPHAT | Particle Flux, emitted, **Pht!**⇒Atoms | NATM | amp | T/C |
| 6 | PRFPAT | Particle Flux, emitted, **BI!**⇒Atoms | NATM | amp | T/C |
| 7 | POTML | Particle Flux, incident, Molecules | NMOL | amp | T/C |
| 8 | PRFAML | Particle Flux, emitted, **Ats!**⇒Molecules | NMOL | amp | T/C |
| 9 | PRFMML | Particle Flux, emitted, **Mls!**⇒Molecules | NMOL | amp | T/C |
| 10 | PRFIML | Particle Flux, emitted, **TI!**⇒Molecules | NMOL | amp | T/C |
| 11 | PRFPHML | Particle Flux, emitted, **Pht!**⇒Molecules | NMOL | amp | T/C |
| 12 | PRFPML | Particle Flux, emitted, **BI!**⇒Molecules | NMOL | amp | T/C |
| 13 | POTIO | Particle Flux, incident, Test Ions | NION | amp | T/C |
| 14 | PRFAIO | Particle Flux, emitted, **Ats!**⇒Test Ions | NION | amp | T/C |
| 15 | PRFMIO | Particle Flux, emitted, **Mls!**⇒Test Ions | NION | amp | T/C |
| 16 | PRFIIO | Particle Flux, emitted, **TI!**⇒Test Ions | NION | amp | T/C |
| 17 | PRFPHIO | Particle Flux, emitted, **Pht!**⇒Test Ions | NION | amp | T/C |
| 18 | PRFPIO | Particle Flux, emitted, **BI!**⇒Test Ions | NION | amp | T/C |
| 19 | POTPHT | Particle Flux, incident, Photons | NPHOT | amp | T/C |
| 20 | PRFAPHT | Particle Flux, emitted, **Ats!**⇒Photons | NPHOT | amp | T/C |
| 21 | PRFMPHT | Particle Flux, emitted, **Mls!**⇒Photons | NPHOT | amp | T/C |
| 22 | PRFIPHT | Particle Flux, emitted, **TI!**⇒Photons | NPHOT | amp | T/C |
| 23 | PRFPHPHT | Particle Flux, emitted, **Pht!**⇒Photons | NPHOT | amp | T/C |
| 24 | PRFPPHT | Particle Flux, emitted, **BI!**⇒Photons | NPHOT | amp | T/C |
| 25 | POTPL | Particle Flux, incident, Bulk Ions | NPLS | amp | T/C |
| 26 | EOTAT | Energy Flux, incident, Atoms | NATM | watt | T/C |
| 27 | ERFAAT | Energy Flux, emitted, **Ats!**⇒Atoms | NATM | watt | T/C |
| 28 | ERFMAT | Energy Flux, emitted, **Mls!**⇒Atoms | NATM | watt | T/C |

Table 6.4: **Surface Averaged Tallies, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 29 | ERFIAT | Energy Flux, emitted, **TI!**⇒Atoms | NATM | watt | T/C |
| 30 | ERFPHAT | Energy Flux, emitted, **Pht!**⇒Atoms | NATM | watt | T/C |
| 31 | ERFPAT | Energy Flux, emitted, **BI!**⇒Atoms | NATM | watt | T/C |
| 32 | EOTML | Energy Flux, incident, Molecules | NMOL | watt | T/C |
| 33 | ERFAML | Energy Flux, emitted, **Ats!**⇒Molecules | NMOL | watt | T/C |
| 34 | ERFMML | Energy Flux, emitted, **Mls!**⇒Molecules | NMOL | watt | T/C |
| 35 | ERFIML | Energy Flux, emitted, **TI!**⇒Molecules | NMOL | watt | T/C |
| 36 | ERFPHML | Energy Flux, emitted, **Pht!**⇒Molecules | NMOL | watt | T/C |
| 37 | ERFPML | Energy Flux, emitted, **BI!**⇒Molecules | NMOL | watt | T/C |
| 38 | EOTIO | Energy Flux, incident, Test Ions | NION | watt | T/C |
| 39 | ERFAIO | Energy Flux, emitted, **Ats!**⇒Test Ions | NION | watt | T/C |
| 40 | ERFMIO | Energy Flux, emitted, **Mls!**⇒Test Ions | NION | watt | T/C |
| 41 | ERFIIO | Energy Flux, emitted, **TI!**⇒Test Ions | NION | watt | T/C |
| 42 | ERFPHIO | Energy Flux, emitted, **Pht!**⇒Test Ions | NION | watt | T/C |
| 43 | ERFPIO | Energy Flux, emitted, **BI!**⇒Test Ions | NION | watt | T/C |
| 44 | EOTPHT | Energy Flux, incident, Photons | NPHOT | watt | T/C |
| 45 | ERFAPHT | Energy Flux, emitted, **Ats!**⇒Photons | NPHOT | watt | T/C |
| 46 | ERFMPHT | Energy Flux, emitted, **Mls!**⇒Photons | NPHOT | watt | T/C |
| 47 | ERFIPHT | Energy Flux, emitted, **TI!**⇒Photons | NPHOT | watt | T/C |
| 48 | ERFPHPHT | Energy Flux, emitted, **Pht!**⇒Photons | NPHOT | watt | T/C |
| 49 | ERFPPHT | Energy Flux, emitted, **BI!**⇒Photons | NPHOT | watt | T/C |
| 50 | EOTPL | Energy Flux, incident, Bulk Ions | NPLS | watt | T/C |
| 51 | SPTAT | Sputtered Flux, by incident Atoms | NATM | amp | T/C |
| 52 | SPTML | Sputtered Flux, by incident Molecules | NMOL | amp | T/C |
| 53 | SPTIO | Sputtered Flux, by incident Test Ions | NION | amp | T/C |
| 54 | SPTPHT | Sputtered Flux, by incident Photons | NPHOT | amp | T/C |
| 55 | SPTPL | Sputtered Flux, by incident Bulk Ions | NPLS | amp | T/C |

Table 6.4: **Surface Averaged Tallies, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 56 | SPTTOT | Sputtered Flux, total | 1 | amp | T/C |
| 57 | ADDS | Additional Surface Tally | NADS | Input | T/C |
| 58 | ALGS | Algebraic expression in surface averaged tallies | NALS | Input | |
| 59 | SPUMP | Pumped flux, by species | NSPZ | amp | |

**Note:**

1. The tallies listed here are two dimensional arrays. The 2nd index I2 is the number of the surface or the surface segment. For I2 = 1, . . . , NLIMI the tallies correspond to the "Additional Surfaces", (input block 3B).

2. For I2 = NLIM + 1, NLIM + NSTSI * NGITT the data correspond to the "Non-default Standard Surfaces" (input block 3A). On each such surface of block 3A, there is a spatial resolution with up to NGITT surface segments, depending upon the standard grid dimensionality.

3. by default the sputter tallies are type and species resolved with respect to incident type and species. E.g. the sputtered fluxes SPTAT(2) is the total sputtered flux (all species), sputtered by incident atoms of species IATM=2. Other conventions regarding sputter tallies are used in EIRENE versions operated at ITER-IO (e.g. SOLPS4.x), see general description above.

4. the sputter tallies have been revised at revision SVN 360, Jan 2014, to allow full resolution with respect to both: incident type (for scaling) and (emitted) sputtered particle type and species, see table 6.5.

Table 6.5: **Sputter tallies, new version, SVN 360 ff, 2014, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 51 | SPTAAT | Sputtered Flux, Atoms, by incident Atoms | NATM | amp | T/C |
| 52 | SPTAML | Sputtered Flux, Molecules, by incident Atoms | NMOL | amp | T/C |
| 53 | SPTAIO | Sputtered Flux, Test Ions, by incident Atoms | NION | amp | T/C |
| 54 | SPTAPHT | Sputtered Flux, Photons, by incident Atoms | NPHOT | amp | T/C |
| 55 | SPTAPL | Sputtered Flux, Bulk Ions, by incident Atoms | NPLS | amp | T/C |
| 56 | SPTMAT | Sputtered Flux, Atoms, by incident Molecules | NATM | amp | T/C |
| 57 | SPTMML | Sputtered Flux, Molecules, by incident Molecules | NMOL | amp | T/C |
| 58 | SPTMIO | Sputtered Flux, Test Ions, by incident Molecules | NION | amp | T/C |
| 59 | SPTMPHT | Sputtered Flux, Photons, by incident Molecules | NPHOT | amp | T/C |
| 60 | SPTMPL | Sputtered Flux, Bulk Ions, by incident Molecules | NPLS | amp | T/C |
| 61 | SPTIAT | Sputtered Flux, Atoms, by incident Test Ions | NATM | amp | T/C |
| 62 | SPTIML | Sputtered Flux, Molecules, by incident Test Ions | NMOL | amp | T/C |
| 63 | SPTIIO | Sputtered Flux, Test Ions, by incident Test Ions | NION | amp | T/C |
| 64 | SPTIPHT | Sputtered Flux, Photons, by incident Test Ions | NPHOT | amp | T/C |
| 65 | SPTIPL | Sputtered Flux, Bulk Ions, by incident Test Ions | NPLS | amp | T/C |
| 66 | SPTPHAT | Sputtered Flux, Atoms, by incident Photons | NATM | amp | T/C |
| 67 | SPTPHML | Sputtered Flux, Molecules, by incident Photons | NMOL | amp | T/C |
| 68 | SPTPHIO | Sputtered Flux, Test Ions, by incident Photons | NION | amp | T/C |
| 69 | SPTPHPHT | Sputtered Flux, Photons, by incident Photons | NPHOT | amp | T/C |
| 70 | SPTPHPL | Sputtered Flux, Bulk Ions, by incident Photons | NPLS | amp | T/C |
| 71 | SPTPAT | Sputtered Flux, Atoms, by incident Bulk Ions | NATM | amp | T/C |
| 72 | SPTPML | Sputtered Flux, Molecules, by incident Bulk Ions | NMOL | amp | T/C |
| 73 | SPTPIO | Sputtered Flux, Test Ions, by incident Bulk Ions | NION | amp | T/C |
| 74 | SPTPPHT | Sputtered Flux, Photons, by incident Bulk Ions | NPHOT | amp | T/C |
| 75 | SPTPPL | Sputtered Flux, Bulk Ions, by incident Bulk Ions | NPLS | amp | T/C |
| 76 | SPTATOP | Sputtered Flux, by incident Atoms | 1 | amp | T/C |
| 77 | SPTMTOT | Sputtered Flux, by incident Molecules | 1 | amp | T/C |

Table 6.5: **Sputter tallies, new version, SVN 360 ff, 2014, Output, Module: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 78 | SPTITOP | Sputtered Flux, by incident Test Ions | 1 | amp | T/C |
| 79 | SPTPHTOT | Sputtered Flux, by incident Photons | 1 | amp | T/C |
| 80 | SPTPLTOT | Sputtered Flux, by incident Bulk Ions | 1 | amp | T/C |
| 81 | SPTTOT | Sputtered Flux, total | 1 | amp | T/C |
| 82 | ADDS | Additional Surface Tally | NADS | Input | T/C |
| 83 | ALGS | Algebraic expression in surface averaged tallies | NALS | Input | |
| 84 | SPUMP | Pumped flux, by species | NSPZ | amp | |

**Note:**
This extended set of sputter tallies has now species naming conventions SPT[A][BC] fully identical to those for volumetric source rates, as well as those for emitted particle and energy fluxes from surfaces. The full terminological equivalence leads to presence of quite strange sputter tallies (e.g. bulk ion fluxes sputtered by incident photons, tally 70), but redundant tallies (and related storage) are automatically removed from a run. Physically irrelevant tallies are listed here only to maintain complete symmetry in notation.

### 6.1.2 old version, w/o photon gas tallies (Eirene$_{2001}$ and older)

In these older versions photon test particle tallies are not available, nor are the momentum source rates MAPL, ... and test particle momentum fluxes VXDEN, ..., VYDEN, ..., VZDEN. In these versions such tallies may still have been available, but then in problem specific tallies COPV (interfacing to particular plasma fluid codes) or in tally BGKV (internal iterations due to neutral–neutral collisions in **BGK!** approximation).

With respect to surface averaged tallies (in and outgoing surface fluxes, sputter fluxes, etc.) there have been major revisions in versions 2014 or later, see "sputter tallies, new version, SVN 360 ff, 2014" in section 6.1.1.
A particular modification (A. Kukushkin) of the sputter tallies has been used in many B2-EIRENE applications, in versions SOLPS4.x, see under section 2.14 in which problem specific issues regarding EIRENE interfacing to the particular external plasma codes B2, B2.5, etc. are described.

Table 6.6: **Input Tallies for Background, Input, Common: COMUSR.**

| No | Name | Macroscopic quantity | 1st Dim. | Units |
|---|---|---|---|---|
| -1 | TEIN | Plasma Temperature, Electrons | 1 | eV |
| -2 | TIIN | Plasma Temperature, Bulk Ions | NPLS | eV |
| -3 | DEIN* | Plasma Density, Electrons | 1 | $cm^{-3}$ |
| -4 | DIIN | Plasma Density, Bulk Ions | NPLS | $cm^{-3}$ |
| -5 | VXIN | Plasma Drift Velocity, x-component, Bulk Ions | NPLS | $cm\,s^{-1}$ |
| -6 | VYIN | Plasma Drift Velocity, y-component, Bulk Ions | NPLS | $cm\,s^{-1}$ |
| -7 | VZIN | Plasma Drift Velocity, z-component, Bulk Ions | NPLS | $cm\,s^{-1}$ |
| -8 | BXIN | Magnetic field unit vector, x-component | 1 | / |
| -9 | BYIN | Magnetic field unit vector, y-component | 1 | / |
| -10 | BZIN | Magnetic field unit vector, z-component | 1 | / |
| -11 | BFIN | Magnetic field strength | 1 | T |
| -12 | ADIN | Additional input tally | NAIN | see INFCOP |
| -13 | EDRIFT* | Kinetic energy in drift motion, Bulk Ions | NPLS | eV |
| -14 | VOL | Zone Volume | 1 | $cm^3$ |
| -15 | WGHT** | Space and species dependent importance | NSPCMC | 1 |
| -22 | POT | electric potential | 1 | V |

**Note:**
We distinguish between so called primary input tallies, and derived input tallies. The latter are automatically computed from the former by EIRENE.
Primary input tallies are set or transferred from external data sources via subr. PLASMA.f, derived input tallies are set via subroutine PLASMA_DERIV.f.

- DEIN and EDRIFT are “derived” input tallies, internally computed from the ion densities (quasi-neutrality), and the flow field, respectively.
- WGHT space species weight window arrays. Currently not in use
- Magnetic field input is needed only for photon transport with Zeeman split line shapes, or in case of charged particle transport (Subr.

FOLION). Only quite rudimentary preprogrammed input options for magnetic field are available, usually it is expected to receive B-field information from external files, see INDPRO(5) options for magnetic field tallies in input block 5 (section 2.5).

Table 6.7: **Volume Averaged Tallies, Output, Common: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 1 | PDENA | Particle density, Atoms | NATM | $cm^{-3}$ | T1 |
| 2 | PDENM | Particle density, Molecules | NMOL | $cm^{-3}$ | T2 |
| 3 | PDENI | Particle density Test Ions | NION | $cm^{-3}$ | T3 |
| 4 | EDENA | Energy density, Atoms | NATM | $eV\,cm^{-3}$ | T4 |
| 5 | EDENM | Energy density, Molecules | NMOL | $eV\,cm^{-3}$ | T5 |
| 6 | EDENI | Energy density, Test Ions | NION | $eV\,cm^{-3}$ | T6 |
| 7 | PAEL | Particle Source (Electrons) from atom-plasma coll. | 1 | $amp\,cm^{-3}$ | T7 |
| 8 | PAAT | Particle Source (Atoms) from atom-plasma coll. | NATM | $amp\,cm^{-3}$ | T8 |
| 9 | PAML | Particle Source (Molecules) from atom-plasma coll. | NMOL | $amp\,cm^{-3}$ | T9 |
| 10 | PAIO | Particle Source (Test Ions) from atom-plasma coll. | NION | $amp\,cm^{-3}$ | T10 |
| 11 | PAPL | Particle Source (Bulk Ions) from atom-plasma coll. | NPLS | $amp\,cm^{-3}$ | T11 |
| 12 | PMEL | Particle Source (Electrons) from molecule-plasma coll. | 1 | $amp\,cm^{-3}$ | T12 |
| 13 | PMAT | Particle Source (Atoms) from molecule-plasma coll. | NATM | $amp\,cm^{-3}$ | T13 |
| 14 | PMML | Particle Source (Molecules) from molecule-plasma coll. | NMOL | $amp\,cm^{-3}$ | T14 |
| 15 | PMIO | Particle Source (Test Ions) from molecule-plasma coll. | NION | $amp\,cm^{-3}$ | T15 |
| 16 | PMPL | Particle Source (Bulk Ions) from molecule-plasma coll. | NPLS | $amp\,cm^{-3}$ | T16 |
| 17 | PIEL | Particle Source (Electrons) from test ion-plasma coll. | 1 | $amp\,cm^{-3}$ | T17 |
| 18 | PIAT | Particle Source (Atoms) from test ion-plasma coll. | NATM | $amp\,cm^{-3}$ | T18 |
| 19 | PIML | Particle Source (Molecules) from test ion-plasma coll. | NMOL | $amp\,cm^{-3}$ | T19 |
| 20 | PIIO | Particle Source (Test Ions) from test ion-plasma coll. | NION | $amp\,cm^{-3}$ | T20 |
| 21 | PIPL | Particle Source (Bulk Ions) from test ion-plasma coll. | NPLS | $amp\,cm^{-3}$ | T21 |
| 22 | EAEL | Energy Source (Electrons) from atom-plasma coll. | 1 | $watt\,cm^{-3}$ | T22 |
| 23 | EAAT | Energy Source (Atoms) from atom-plasma coll. | 1 | $watt\,cm^{-3}$ | T23 |
| 24 | EAML | Energy Source (Molecules) from atom-plasma coll. | 1 | $watt\,cm^{-3}$ | T24 |
| 25 | EAIO | Energy Source (Test Ions) from atom-plasma coll. | 1 | $watt\,cm^{-3}$ | T25 |
| 26 | EAPL | Energy Source (Bulk Ions) from atom-plasma coll. | 1 | $watt\,cm^{-3}$ | T26 |
| 27 | EMEL | Energy Source (Electrons) from molecule-plasma coll. | 1 | $watt\,cm^{-3}$ | T27 |

Table 6.7: **Volume Averaged Tallies, Output, Common: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 28 | EMAT | Energy Source (Atoms) from molecule-plasma coll. | 1 | watt cm$^{-3}$ | T28 |
| 29 | EMML | Energy Source (Molecules) from molecule-plasma coll. | 1 | watt cm$^{-3}$ | T29 |
| 30 | EMIO | Energy Source (Test Ions) from molecule-plasma coll. | 1 | watt cm$^{-3}$ | T30 |
| 31 | EMPL | Energy Source (Bulk Ions) from molecule-plasma coll. | 1 | watt cm$^{-3}$ | T31 |
| 32 | EIEL | Energy Source (Electrons) from test ion-plasma coll. | 1 | watt cm$^{-3}$ | T32 |
| 33 | EIAT | Energy Source (Atoms) from test ion-plasma coll. | 1 | watt cm$^{-3}$ | T33 |
| 34 | EIML | Energy Source (Molecules) from test ion-plasma coll. | 1 | watt cm$^{-3}$ | T34 |
| 35 | EIIO | Energy Source (Test Ions) from test ion-plasma coll. | 1 | watt cm$^{-3}$ | T35 |
| 36 | EIPL | Energy Source (Bulk Ions) from test ion-plasma coll. | 1 | watt cm$^{-3}$ | T36 |
| 37 | ADDV | Additional volume av. Tally, Track-length estimated | NADV | see UPTUSR | TRL |
| 38 | COLV | Additional volume av. Tally, Collision estimated | NCLV | see UPCUSR | COL |
| 39 | SNPV | Additional volume av. Tally, Snapshot estimated | NSNV | see UPSUSR | SNP |
| 40 | COPV | Tallies for coupling to ext. code, see: Subr. UPTCOP | NCPV | see UPTCOP | COP |
| 41 | BGKV | Volume averaged tallies for **BGK!**-self-collision terms | NBGV | see BGK | T41 |
| 42 | ALGV | Algebraic expression in volume averaged tallies | NALV | Input | ALG |
| 43 | PGENA | Generation limit, Atoms | NATM | amp cm$^{-3}$ | T1 |
| 44 | PGENM | Generation limit, Molecules | NMOL | amp cm$^{-3}$ | T1 |
| 45 | PGENI | Generation limit, Test ions | NION | amp cm$^{-3}$ | T1 |
| 46 | EGENA | ditto, Energy flux, Atoms | NATM | watt cm$^{-3}$ | T1 |
| 47 | EGENM | ditto, Energy flux, Molecules | NMOL | watt cm$^{-3}$ | T1 |
| 48 | EGENI | ditto, Energy flux, Test ions | NION | watt cm$^{-3}$ | T1 |
| 49 | VGENA | ditto, momentum flux, Atoms | NATM | as COP | T1 |
| 50 | VGENM | ditto, momentum flux, Molecules | NMOL | as COP | T1 |
| 51 | VGENI | ditto, Momentum flux, Test ions | NION | as COP | T1 |

**Note:**
*Source* means: if the sign is positive, it is a gain for the specified type of particles; if it is negative, it is a loss.

Table 6.8: **Surface Averaged Tallies, Output, Common: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 1 | POTAT | Particle Flux, incident, Atoms | NATM | amp | T/C |
| 2 | PRFAAT | Particle Flux, emitted, **Ats!**⇒Atoms | NATM | amp | T/C |
| 3 | PRFMAT | Particle Flux, emitted, **Mls!**⇒Atoms | NATM | amp | T/C |
| 4 | PRFIAT | Particle Flux, emitted, **TI!**⇒Atoms | NATM | amp | T/C |
| 5 | PRFPAT | Particle Flux, emitted, **BI!**⇒Atoms | NATM | amp | T/C |
| 6 | POTML | Particle Flux, incident, Molecules | NMOL | amp | T/C |
| 7 | PRFAML | Particle Flux, emitted, **Ats!**⇒Molecules | NMOL | amp | T/C |
| 8 | PRFMML | Particle Flux, emitted, **Mls!**⇒Molecules | NMOL | amp | T/C |
| 9 | PRFIML | Particle Flux, emitted, **TI!**⇒Molecules | NMOL | amp | T/C |
| 10 | PRFPML | Particle Flux, emitted, **BI!**⇒Molecules | NMOL | amp | T/C |
| 11 | POTIO | Particle Flux, incident, Test Ions | NION | amp | T/C |
| 12 | PRFAIO | Particle Flux, emitted, **Ats!**⇒Test Ions | NION | amp | T/C |
| 13 | PRFMIO | Particle Flux, emitted, **Mls!**⇒Test Ions | NION | amp | T/C |
| 14 | PRFIIO | Particle Flux, emitted, **TI!**⇒Test Ions | NION | amp | T/C |
| 15 | PRFPIO | Particle Flux, emitted, **BI!**⇒Test Ions | NION | amp | T/C |
| 16 | POTPL | Particle Flux, incident, Bulk Ions | NPLS | amp | T/C |
| 17 | EOTAT | Energy Flux, incident, Atoms | NATM | watt | T/C |
| 18 | ERFAAT | Energy Flux, emitted, **Ats!**⇒Atoms | NATM | watt | T/C |
| 19 | ERFMAT | Energy Flux, emitted, **Mls!**⇒Atoms | NATM | watt | T/C |
| 20 | ERFIAT | Energy Flux, emitted, **TI!**⇒Atoms | NATM | watt | T/C |
| 21 | ERFPAT | Energy Flux, emitted, **BI!**⇒Atoms | NATM | watt | T/C |
| 22 | EOTML | Energy Flux, incident, Molecules | NMOL | watt | T/C |
| 23 | ERFAML | Energy Flux, emitted, **Ats!**⇒Molecules | NMOL | watt | T/C |
| 24 | ERFMML | Energy Flux, emitted, **Mls!**⇒Molecules | NMOL | watt | T/C |
| 25 | ERFIML | Energy Flux, emitted, **TI!**⇒Molecules | NMOL | watt | T/C |
| 26 | ERFPML | Energy Flux, emitted, **BI!**⇒Molecules | NMOL | watt | T/C |
| 27 | EOTIO | Energy Flux, incident, Test Ions | NION | watt | T/C |
| 28 | ERFAIO | Energy Flux, emitted, **Ats!**⇒Test Ions | NION | watt | T/C |

Table 6.8: **Surface Averaged Tallies, Output, Common: CESTIM.**

| No | Name | Macroscopic quantity | 1st Dim. | Units | Estim. |
|---|---|---|---|---|---|
| 29 | ERFMIO | Energy Flux, emitted, **MIs!**⇒Test Ions | NION | watt | T/C |
| 30 | ERFIIO | Energy Flux, emitted, **TI!**⇒Test Ions | NION | watt | T/C |
| 31 | ERFPIO | Energy Flux, emitted, **BI!**⇒Test Ions | NION | watt | T/C |
| 32 | EOTPL | Energy Flux, incident, Bulk Ions | NPLS | watt | T/C |
| 33 | SPTAT | Sputtered Flux, by incident Atoms | NATM | amp | T/C |
| 34 | SPTML | Sputtered Flux, by incident Molecules | NMOL | amp | T/C |
| 35 | SPTIO | Sputtered Flux, by incident Test Ions | NION | amp | T/C |
| 36 | SPTPL | Sputtered Flux, by incident Bulk Ions | NPLS | amp | T/C |
| 37 | SPTTOT | Sputtered Flux, total | 1 | amp | T/C |
| 38 | ADDS | Additional Surface Tally | NADS | Input | T/C |
| 39 | ALGS | Algebraic expression in surface averaged tallies | NALS | Input | |
| 40 | SPUMP | Pumped flux, by species | NSPZ | amp | |



**Note:**
The tallies listed here are two dimensional arrays. The 2nd index I2 is the number of the surface or the surface segment. For I2 = 1, . . . , NLIMI the tallies correspond to the "Additional Surfaces", (input block 3B).
For I2 = NLIM + 1, NLIM + NSTSI * NGITT the data correspond to the "Non-default Standard Surfaces" (input block 3A). On each such surface, there is a spatial resolution with up to NGITT surface segments.