Abstract

This is the user's manual for DEGAS 2 - A Monte Carlo code for the study of neutral atom and molecular transport in confined plasmas. It is intended to provide an introduction to DEGAS 2 from the user's point-of-view: obtaining the files, compiling them, setting up the input files, executing the code, and interpreting the output. References will be provided for the underlying physics, but the essential aspects will be highlighted. More detailed documentation of the procedures used to create input files is contained in the typeset source code of the relevant pre-processors; links to those files are provided.

User's Guide for DEGAS 2
Release V. 4.9[1]

Daren Stotler and Charles Karney
Princeton Plasma Physics Laboratory
Randall Kanzleiter
Rensselaer Polytechnic Institute
Jaishankar S.
Institute for Plasma Research, India

September 23, 2020

[1]This file is written in TEX. A hyper-linked PDF file is generated from that source using `pdflatex`. This document is distributed as part of the DEGAS 2 code. The user can be best assured of concurrence between code and manual by referring to the version of the manual that came with the code. The DEGAS 2 code itself is licensed under the terms spelled out in the user agreement. Permission is granted to make and distribute verbatim copies of this manual provided the copyright and permission notice is preserved on all copies. Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the entire resulting derived work is distributed under the terms of a permission notice identical to this one. Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions.

Chapter 1

Introduction

1.1 Purpose of Monte Carlo Neutral Transport

The interaction of neutral atoms and molecules with the background plasma is an important component in the physics of fusion research plasmas. These neutrals affect both the particle and energy balance of the plasma, providing a source of new plasma and a channel for heat transport across the magnetic field that confines the charged particles. In addition, hot neutrals which leave the plasma volume can interact with the wall of the plasma chamber, sputtering impurities into the plasma and (in reactors) possibly damaging the wall. Finally, these hot neutrals can also be used for plasma diagnostics. An accurate description of the neutral behaviour would, thus, have be useful in all of these areas.

The theory of neutral particle kinetics[1] treats the transport of mass, momentum, and energy in a plasma due to neutral particles that are themselves unaffected by magnetic fields. Though analytic models (solving the Boltzmann equation for the neutrals or using the diffusion approximation), both single- and multi-species, have been used to study neutral transport, a variety of approximations must be made. Some numerical methods for solving the Boltzmann equation employ simplified geometries and atomic physics. On the other hand, the Monte Carlo approach to integrating the Boltzmann equation can treat in great detail three-dimensional geometries, as well as complex atomic physics and wall interactions. Because of the similarity in the problems, Monte Carlo neutral transport codes can build on the techniques developed for neutron transport in reactor design[2].

1.2 Historical Background

The predecessor to DEGAS 2 was DEGAS[3]. The origin of the "DEGAS" name can best be explained by pointing out that the first Monte Carlo neutral transport code written by D. B. Heifetz was named SEURAT. Its name was a direct reference to the artist Georges Seurat. This appellation was inspired by the analogy between the artist's pointillist style and the nature of the Monte Carlo algorithm. Up close, one sees that Seurat's paintings consist of a huge number of very tiny dots of paint of a variety of colors. Only when viewed from a distance is the image and its coloring clearly apparent. In much the same way, the track of an individual neutral flight in the Monte Carlo code gives relatively little insight into the problem being studied. Yet, when the results of many flights are compiled together, an accurate solution can be obtained.

Heifetz' fondness for Impressionist artists was also manifested in succeeding codes like MONET and DEGAS. The latter name was further motivated by two associated puns. One is that "DEGAS" is an abbreviation of "DEuterium GAS"; DEGAS was written to satisfy a request for a code that could treat deuterium as well as hydrogen. The other is that "DEGAS" is just short for "degassing"; another proposed application for the code was in assessing measurements of wall degassing in fusion devices.

For those who insist that a code's name must be an acronym, we will reply that "DEGAS" stands for Divertor and Edge Gas Analysis System.

By the time the authors began contemplating the concepts that led to DEGAS 2, the original DEGAS code was in widespread use. It also enjoyed a very good reputation as being easy to use and well documented. For this reason, we were reluctant to abandon the DEGAS name, even though we knew we would be rewriting the code largely from scratch. Nonetheless, much of the atomic and surface physics from the original code has been carried over. Likewise, many of the numerical techniques of DEGAS served as inspiration for those in DEGAS 2.

1.3 Need for DEGAS 2

Two events pointed out limitations in the original DEGAS code[3].

- The combined DEGAS-B2 code was too slow (about 100 Cray hours)[4]. The stronger the coupling between neutral and plasma species, the more difficult it is to arrive at a converged solution. In this case which looked at a high recycling divertor for a compact, high field reactor, hundreds or

thousands of complete Monte Carlo profiles must be carried out in order to couple effectively to a fluid code.

- The problem physics was difficult to modify. In the radiative and detached divertor regimes being investigated for future reactors, it is believed that the list of important atomic processes will grow to include elastic collisions and molecular hydrogenic species other than ground state H_2.

Hence, it was decided that a new code should be written which draws on the extensive experience at PPPL and that from Garching and Jülich. The philosophy behind DEGAS 2 was defined during 1992-1993:

- High speed,
- Flexibility,
- Ease-of-use,
- Well documented.

1.4 DEGAS 2 Features

An effort was made to realise the above mentioned objectives of DEGAS 2 by the following means:

- Improve speed relative to DEGAS by utilizing the "track length" estimator.
- Ensure flexibility by designing the code using quasi-object oriented programming techniques. The modifier "quasi" refers to the fact that DEGAS 2 is written in FORTRAN (compatible with FORTRAN 77 and FORTRAN 90). Even though the basic "classes" in the problem have been abstracted, the degree of encapsulation actually achieved (via pre-processor macros) in DEGAS 2 is limited. Furthermore, there is no provision for inheritance within this programming environment.
- Provide an easy-to-use and well documented code by writing it with the FWEB[8] package. FWEB provides the powerful preprocessing and macro facility which underlies the "quasi-object oriented" implementation and internal TEX documentation.

Other significant features of DEGAS 2 include:

- Dynamic memory allocation for optimal performance in large problems,
- Operation on a number of UNIX platforms, compilable with both FORTRAN 77 and FORTRAN 90,
- Input and output data stored in self-describing, platform-independent binary files. Presently, the main code uses exclusively netCDF[9] (Network Common Data Format) files for input and output. However, a simple post-processing utility is included which generates graphics and data in HDF[10] (Hierarchial Data Format) or Silo[11] files. Hopefully, a single format will be settled upon eventually.
- Parallel execution on distributed workstations and massively parallel computers. This is the most promising technique for improving the speed of DEGAS 2. The reason is that most of the future advancements in computer speed will result from increases in the number of processors. Parallel processing is practical to Monte Carlo since the amount of communication required per computation is small.
- High quality coding standards. Good programming practices such as use of the `implicit none` statement and an assertion facility help to ensure that the code works properly. This document will eventually describe a basic test suite which actually demonstrates that the code is correct. In the code design process, clarity has been given a high priority as well, facilitating readability and maintainability.
- Code versions numbered and documented with the "CVS" (concurrent version system) package.

Chapter 2

Background

2.1 Generic Monte Carlo Algorithms

Monte Carlo methods generally involve sampling from some "parent population". The sampling process is usually facilitated by pseudo-random numbers generated via a numerical algorithm. The quantity or quantities sampled are then used as input to a subsequent (deterministic) calculation which results in the desired output. This process is repeated a number of times, so as to thoroughly sample the parent population, and a distribution of the output data is compiled. In some cases, only the mean of this distribution is of interest (as well as some estimate of its error), while in other cases, the distribution itself is the objective.

Note that the process being simulated need not be random. For example, Monte Carlo techniques may be the only way of computing the volume of some complex three-dimensional object. This is just one example of the versatility of Monte Carlo techniques. But, Monte Carlo is not always the most efficient solution for a problem because of the significant computational effort required. Two reasons for using Monte Carlo are:

1. No other solution is available.

2. The details of the problem are sufficiently important that no approximations can be tolerated.

In the case of particle transport, the latter reason is usually invoked. Examples of Monte Carlo particle transport include:

1. Neutron tranport in fission devices,

2. Radiation transport,

3. Electron transport in semiconductors,

4. Neutral particle transport in plasmas.

2.1.1 Sampling a Distribution

The most fundamental process of a Monte Carlo simulation is the sampling of probability distributions. Typical examples for a particle simulations would be:

1. Choosing a velocity vector from a Maxwellian distribution,

2. Picking a particle from a spatially varying source,

3. Determining the distance to the next collision.

2.1.2 Sampling Distance to Next Collision

To illustrate the third case, consider particles with a total, macroscopic cross section Σ_t. We write the *probability density function* for having the next collision between x and $x + dx$, $p(x)\,dx$, as:

$$p(x)dx = \Sigma_t \exp(-\Sigma_t x)\,dx. \tag{2.1}$$

Then, the *cumulative probability distribution* $P(y)$,

$$P(y) = \int_0^y p(x)\,dx, \tag{2.2}$$

is the probability of having a collision for any $x \leq y$. Note that $P(0) = 0$ and $P(\infty) = 1$.

As suggested by the above figure, $P(y)$ can be sampled by choosing a uniformly distributed random number ξ, with $0 \leq \xi < 1$. For this example, $P(y)$ is easily computed and inverted so that

$$y = -\frac{\ln \xi}{\Sigma_t}, \tag{2.3}$$

where we have surreptitiously replaced $1 - \xi$ with ξ since both are uniform over the unit interval. Again the logic of this process, as depicted in the figure, is that if ξ is uniformly distributed, then numbers between ξ to $\xi + \Delta\xi$ are chosen with the same frequency as particles having collisions between y and $y + \Delta y$. Thus, y is the sampled *distance to next collision*.

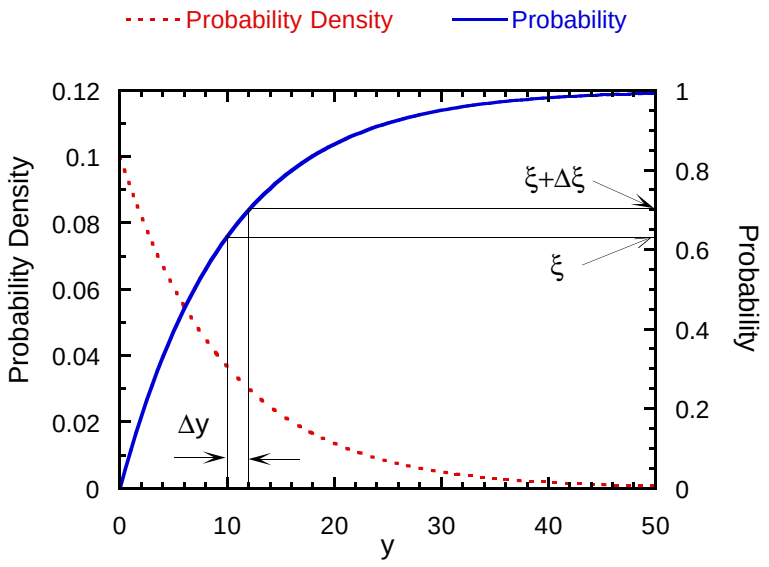

Figure 2.1: A uniformly sampled ξ value is translated into a sample of y distributed according to the "Probability Density" curve using the integrated "Probability" curve.

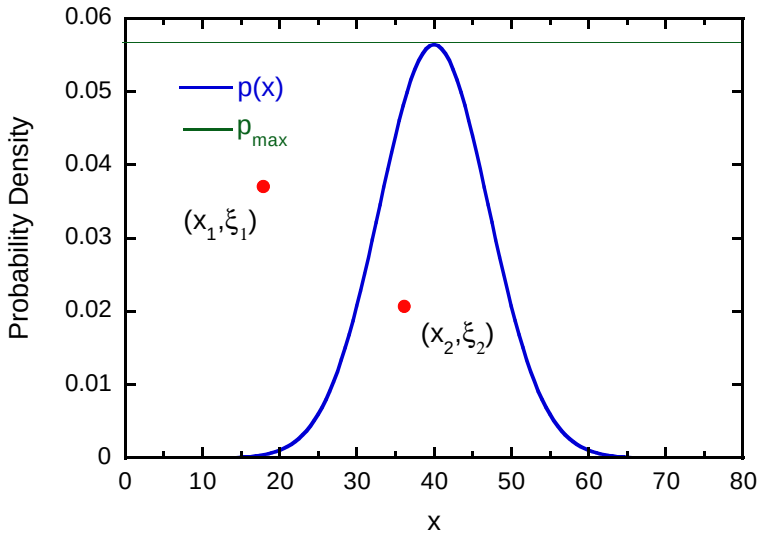

Figure 2.2: Values of x distributed according to $p(x)$ can be sampled by uniformly selecting (x, ξ) pairs and tossing out ones with $\xi > p(x)$.

2.1.3 Sampling More Complicated Distributions

Because of the simplicity of the functional form of the collision probability density function, the distance to the next collision can be easily sampled using a single, uniform random number. If this were not the case, more general methods, such as this *rejection technique*, would have to be employed.

As an example, consider the sampling of a velocity v from a Maxwellian distribution,

$$p(v) = \frac{1}{\sqrt{\pi} v_{\rm th}} \exp\left[-\frac{(v - v_0)^2}{v_{\rm th}^2} \right]. \tag{2.4}$$

For simplicity, limit the v values to $|v - v_0| < M v_{\rm th}$ for some appropriate M.

Now, we choose a pair of random numbers (x, ξ) with x uniformly distributed over the interval $v_0 - M v_{\rm th} \leq x < v_0 + M v_{\rm th}$ and ξ uniform over $0 \leq \xi < p_{\rm max}$, where $p_{\rm max}$ is the maximum value of $p(v)$. If $\xi < p(x)$, accept this x as the sampled velocity. Otherwise, if $\xi > p(x)$, reject these (x, ξ) and sample another pair. That the resulting set of x values is distributed as required can be seen by studying the figure above. This figure also makes the case that the rejection approach may not be very efficient for sampling a Maxwellian (indeed, DEGAS 2 uses the well-known Box-Mueller method for this task).

In general, a rejection technique factors the probability density $p(x)$ into two functions, $p(x) = g(x)h(x)$, where it is known how to sample x from the function $g(x)$. The acceptance condition is just $\xi < h(x)$. The Maxwellian example simply sets $g(x) = 1$.

2.2 Particle Transport

One of the main benefits of doing Monte Carlo particle transport is that most of the time the algorithm can be understood directly from a physical picture. Namely, we think of the code as tracking particles through a medium and having collisions with probabilities determined from the cross sections of the various collision processes. The outcome of those collisions is the same as for the actual physical particles, and so on. This is known as the *analog* description of Monte Carlo particle transport.

In this section, we want to give a rough idea of how this picture relates to the integral equations which provide the actual mathematical basis for Monte Carlo techniques. Only from these equations can we understand the *non-analog* techniques (the rejection technique described in the previous subsection is a non-analog technique) used to make the Monte Carlo process more efficient. We hope to familiarize the reader with the integral equations to the point where more detailed references[2, 12] can be readily digested.

From an analog point-of-view, we imagine a Monte Carlo particle simulation as consisting of four parts:

1. A way to sample an initial particle distribution,
2. A way to track particles from one point in space to another,
3. Tools to treat "interesting things" that might happen along the way,
4. Some useful quantities to compute and record at each stage of the process.

The last of these, referring to "scores" and their "estimators" will be addressed in Sec. 2.3.

2.2.1 Motivating the Integral Equation

The initial particle distribution, or source, is given by a probability density function $Q(x) > 0$. We take x here to represent a point in phase space so that this function provides both the velocity and spatial distribution of the source.

The tracking of particles is described in the integral equation by a function $L(y \to x)$ which gives the probability that a particle starting at a point y in phase space ends up at point x. This function includes the "interesting things" (collisions) that might happen along the way, as we will describe in the next subsection.

Using these two functions, we can write the *density of particles arriving at* x, $\chi(x)$ as the sum of two pieces:

1. Arrivals directly from source: $Q(x)$.

2. Particles transported from elsewhere:

$$\int \chi(y)L(y \to x)\, dy \tag{2.5}$$

And we have

$$\chi(x) = Q(x) + \int \chi(y)L(y \to x)\, dy. \tag{2.6}$$

2.2.2 Transport and Collision Kernels

In order to describe the kernel $L(y \to x)$ in more detail, substitute for y and x the phase space variables $(\vec{r}, \vec{v})$ and a species index i. The kernel L can then be broken into two pieces:

$$L(\vec{r}', \vec{v}', i' \to \vec{r}, \vec{v}, i) = T(\vec{r}' \to \vec{r}; \vec{v}, i)C(\vec{v}', i' \to \vec{v}, i; \vec{r}), \tag{2.7}$$

where $T(\vec{r}' \to \vec{r}; \vec{v}, i)$ is the (probability) density of particles leaving a collision at $\vec{r}'$ and having a subsequent collision at $\vec{r}$. The ";" is used in the arguments of T and C to separate changing parameters (on the left) from those held fixed (on the right). Basically, sampling from T is as described in Section 2.1.2.

Collisions are described by $C(\vec{v}', i' \to \vec{v}, i; \vec{r})$ which provides the probability that a particle of velocity $\vec{v}'$, species i' will have a collision resulting in a particle of velocity $\vec{v}$, species i. In practice, C is a sum over the physical reactions in which species i' participates.

2.2.3 Integral Equation Details

One of the problems with reading Monte Carlo particle references is that there are several slightly different "quantities of interest" equivalent to $\chi(x)$. Furthermore, there is no well-established notation for those quantities. In this subsection, we

attempt to list these various quantities and to provide a detailed form of an integral equation for later reference.

The quantity $\chi(x)$ is called the *post-collision density*. The origin of this designation should be clear from the previous discussion. Equivalently, the integral equation can be written in terms of the *pre-collision density* $\psi(x)$. The two are related by

$$\psi(\vec{r},\vec{v},i) = \int d^3r' \chi(\vec{r}',\vec{v},i) T(\vec{r}' \to \vec{r};\vec{v},i). \tag{2.8}$$

It is this pre-collision density which is most closely related to the more familiar particle flux Φ and particle distribution function f by the definitions:

$$\psi(\vec{r},\vec{v},i) = \Sigma_t(\vec{r},\vec{v},i)\Phi(\vec{r},\vec{v},i) = \Sigma_t(\vec{r},\vec{v},i)|\vec{v}|f(\vec{r},\vec{v},i). \tag{2.9}$$

The integral equation for ψ bears a strong resemblance to that for χ. We write it out here in detail to illustrate the details as well as to provide a reference point for the rest of the document and other reading:

$$\begin{aligned} \psi(\vec{r},\vec{v},i) &= S(\vec{r},\vec{v},i) \\ &\quad + \int_0^\infty dR \int d^3v\, C(\vec{v'},i' \to \vec{v},i;\vec{r}-\vec{\Omega}R) \\ &\quad \times T(\vec{r}-\vec{\Omega}R \to \vec{r};\vec{v},i)\psi(\vec{r}-\vec{\Omega}R,\vec{v'},i'), \end{aligned} \tag{2.10}$$

where

$$S(\vec{r},\vec{v},i) = \int_0^\infty dR\, Q(\vec{r}-\vec{\Omega}R,\vec{v},i) T(\vec{r}-\vec{\Omega}R \to \vec{r};\vec{v},i), \tag{2.11}$$

and

$$T(\vec{r}-\vec{\Omega}R \to \vec{r};\vec{v},i) = \Sigma_t(\vec{r},\vec{v},i) \exp\left[-\int_0^R dR'\, \Sigma_t(\vec{r}-\vec{\Omega}R',\vec{v},i)\right]. \tag{2.12}$$

The actual form of the collision kernel C need not be written down here. Instead, in this document we'll only talk about its implementation from an analog point of view. The only additional piece of information about it that we could add to the above equations is that C consists of a sum over the relevant collision processes involving species i'.

The approach of writing the path integrals as above instead of as general phase-space volume integrals with the path enforced via delta-functions[12] was copied from Case and Zweifel[13] We find that this leads to expressions which are easier to work with and think about.

2.3 Estimators

The objective of the Monte Carlo calculation is to evaluate integrals over the distribution function:

$$I(g) \equiv \int d^3r d^3v f(\vec{r},\vec{v}) g(\vec{r},\vec{v}). \tag{2.13}$$

For example, if we wish to estimate a reaction rate $\rho \equiv \Sigma|\vec{v}|$,

$$I(\rho) = \int d^3r d^3v \Sigma\Phi(\vec{r},\vec{v}). \tag{2.14}$$

Values of ρ are accumulated along the random walk of the neutral flight and used to estimate $I(\rho)$. The mathematical description of this accumulation process is called the *estimator* of I.

2.3.1 Simple Estimators

Consider a random walk γ which consists of k steps, $(x_1, x_2, \ldots, x_k)$. I.e., the particle is absorbed or otherwise terminated at the kth step. The simplest estimator involves writing down the value of ρ at the point of absorption. Thus, the absorption estimator is

$$\xi_a(\gamma) = \frac{\rho}{\rho_a} w\chi_V, \tag{2.15}$$

where ρ_a is the absorption rate in the volume of interest V and w is the statistical weight of the particle (constant for this random walk process). The function χ_V

$$\chi_V(x_m) = \begin{cases} 1 & \text{if } x_m \in V \\ 0 & \text{if } x_m \notin V \end{cases}, \tag{2.16}$$

just permits the score to be localized to the volume of interest.

The expectation value of the estimator is just the sum over all possible random walks γ of the probability of each walk $P(\gamma)$ times the corresponding estimator,

$$E(\xi_a) = \sum_{\gamma} P(\gamma)\xi_a(\gamma). \tag{2.17}$$

Obviously, only one data point is collected for each random walk.

We can instead accumulate a score at each step (scattering collision) of the random walk.

$$\xi_{c1}(\gamma) = \sum_{m=1}^{k} \left[\frac{\rho_a(x_m)}{\rho_t(x_m)}\right] w\chi_V(x_m), \tag{2.18}$$

where $\rho_t = \rho_a + \rho_s$ is the total reaction rate for the particle, with ρ_s being the scattering rate. The subscript $c1$ denotes the first collision estimator described here. The expectation value of the estimator is again formally written as

$$E(\xi_{c1}) = \sum_{\gamma} P(\gamma)\xi_{c1}(\gamma). \tag{2.19}$$

Because of the greater amount of data accumulated along each particle track, the variance of this estimator will usually be smaller than that of the absorption estimator.

Consider now a random walk process in which absorption is forbidden. This is our first departure from the usual analog processes. Instead of an absorption, the weight of the particle is reduced at each scattering collision by its survival probability. So, at the mth step of the walk the weight of the particle is

$$w_m = w \prod_{j=1}^{m} \frac{\rho_s(x_j)}{\rho_t(x_j)}. \tag{2.20}$$

The flight must be terminated in some manner; the usual practice is to have it undergo Russian roulette[2] when $w_m < w_{\min}$, where $w_{\min}$ is some specified minimum weight. The collision estimator for this nonanalog random walk process is then

$$\xi_{c2}(\gamma) = \sum_{m=1}^{k} w_m \left[\frac{\rho_a(x_m)}{\rho_t(x_m)}\right] \chi_V(x_m). \tag{2.21}$$

For formal proofs that these expectation values do indeed yield $I(\rho)$, see Spanier and Gelbard[2].

2.3.2 Generalized Collision and Tracklength Estimators

Two principal estimators are used in DEGAS 2. One is a generalization of the collision estimator described in the previous subsection; the other is the tracklength estimator.

These estimators can be obtained by making a slight modification of the derivation given by Macmillan[54] (who had improved upon an earlier approach described by Spanier[55]). To make the connection to DEGAS 2 clearer, we replace the Σ and d variables of Macmillan with ρ and t.

Macmillan's paper, like the earlier one by Spanier, aims to derive the tracklength estimator. Collisions are described as either scatterings, rate ρ_s, or absorptions ρ_a. The starting point is the nonanalog random walk process described

in Sec. 2.3.1. The probability of an event occurring in a time interval dt is $(\rho_s + \rho_a)\, dt$. Again, the weight reduction factor applied at each scattering is $\rho_s/(\rho_s + \rho_a)$.

The trick used by Spanier is to introduce a fictitious "delta scattering" that has no impact on the particle's trajectory or weight. These "pseudocollisions" are not unrelated to those of the original DEGAS code[3]. Additional flexibility in the final results is obtained by associating some fraction of the absorptions α with the delta scatterings. To obtain the estimators used in DEGAS 2, we work with a total fictitious scattering reaction rate

$$\rho_{t,\delta} \equiv \rho_{s,\delta} + \alpha\rho_a, \tag{2.22}$$

where $\rho_{s,\delta}$ is equivalent to Macmillan's $\Sigma_{s,\delta}$. Then, the probability of a delta scattering in a time interval dt is $\rho_{t,\delta}\, dt$. At each delta scattering, the weight is reduced by a factor $(\rho_{t,\delta} - \alpha\rho_a)/\rho_{t,\delta}$. The probability of a real scattering in the same interval is $(\rho_s + (1-\alpha)\rho_a)\, dt$; at each such event, the weight is reduced by $\rho_s/[\rho_s + (1-\alpha)\rho_a]$.

If a particle travels a time t in a region of constant parameters before having a real collision, the conditional probability of having n pseudocollisions is

$$P(n \mid d) = \frac{(\rho_{t,\delta} t)^n}{n!} \exp(-\rho_{t,\delta} d). \tag{2.23}$$

At each pseudo or real collision, we use the standard collision estimator for the nonanalog random walk, taken from Eq. (2.21):

$$\xi = \frac{\rho}{\rho_t^*} w, \tag{2.24}$$

where w is the current weight of the particle and

$$\rho_t^* = \rho_{t,\delta} + \rho_s + (1-\alpha)\rho_a \tag{2.25}$$

is the overall total reaction rate.

The expected value of this estimator over the n collisions in the region is then

$$E(\xi \mid t) = \sum_{n=0}^{\infty} P(n \mid t) \frac{\rho}{\rho_t^*} \sum_{i=0}^{n} \left(\frac{\rho_{t,\delta} - \alpha\rho_a}{\rho_{t,\delta}} \right)^i, \tag{2.26}$$

where we have assumed that the initial weight of the particle is 1. The last factor represents the subsequent weight reductions with each pseudocollision.

Following Macmillan's evaluation of $E(\xi \mid t)$, we find that we can write

$$E(\xi \mid t) = c' \frac{\rho}{\rho_s + (1-\alpha)\rho_a} + (1-c')\rho \frac{1-\exp(-\alpha\rho_a t)}{\alpha\rho_a}, \tag{2.27}$$

where

$$c' \equiv \frac{\rho_s + (1-\alpha)\rho_a}{\rho_{t,\delta} + \rho_s + (1-\alpha)\rho_a} \tag{2.28}$$

should be compared with Macmillan's c.

Macmillan's expression for the weight reduction after time t is exactly the same in our case:

$$E(w \mid t) = \exp(-\alpha\rho_a t) \frac{\rho_s}{\rho_s + (1-\alpha)\rho_a}. \tag{2.29}$$

The factor at the end represents the weight reduction by the real collision at time t. If the particle travels a time t through the region without having a real collision, its weight should be reduced by just $\exp(-\alpha\rho_a t)$.

In this latter situation in which there is not a real collision at time t, only the pseudocollisions contribute to the score. An interesting difference from MacMillan's approach is that if we consider the limit $\rho_{t,\delta} \to 0$, we should get no score at all. In contrast, MacMillan's pure "collision estimator" still compiles a nonzero score as $\Sigma_{s,\delta} \to 0$ through the $\alpha\Sigma_a$ contribution to the pseudocollision cross section. Thus, MacMillan's "collision estimator" behaves somewhat like a track-length estimator if there are no collisions in a region. For simple scores, this behavior is desirable: lower variances will result in regions of small cross section. However, a few scores are sufficiently complex that averaging along the particle path cannot be done, and scoring can only be done at collisions. A pure collision estimator is required for handling such scores.

In the case of no real collision at time t, the expectation value for our estimator becomes:

$$E(\xi \mid t) = \frac{\rho}{\rho_t^*} \rho_{t,\delta} \frac{1-\exp(-\alpha\rho_a t)}{\alpha\rho_a}. \tag{2.30}$$

Clearly, $E(\xi \mid t) \to 0$ as $\rho_{t,\delta} \to 0$. The limiting expressions above for $\rho_{t,\delta} \to 0$ can be obtained from MacMillan's with $\Sigma_{s,\delta} \to -\alpha\Sigma_a$.

We now consider the expressions used in DEGAS 2. We start by using $\alpha = 1$ so that absorptions occur only at the delta scatterings. This is consistent with our use of the nonanalog random walk, i.e., suppressed absorption. The collision estimator is then given by $\rho_{t,\delta} = 0$,

$$\xi_c \equiv \frac{\rho}{\rho_s} \exp(-\rho_a t) w \chi_V, \tag{2.31}$$

where w is the weight of the particle at the beginning of timestep t and we have inserted the characteristic function of V χ_V for completeness. At the end of the timestep (after the collision is scored) the weight is reduced by

$$w' = w \exp(-\rho_a t). \tag{2.32}$$

The particle track continues from there, and multiple scores may be made inside the volume V. If the flight travels through the volume V without a collision, no contribution to the score is made. However, the weight is still reduced by the factor in Eq. (2.32).

The track-length estimator is obtained with $\rho_{t,\delta} \to \infty$,

$$\xi_t \equiv \rho \frac{1 - \exp(-\rho_a t)}{\rho_a} w \chi_V. \tag{2.33}$$

Note that Eq. (2.30) yields this same result in the case of no collisions. The same weight reduction, Eq. (2.32) is applied at the end of t. Conveniently, this allows us to use the same weight factor for both estimators (true in MacMillan's case as well).

Molecules and some other species are not treated with suppressed absorption. Instead, an ionizing collision terminates the flight (typically changing the species). The estimators for this case are obtained by setting $\alpha = 0$ in the above expressions:

$$\xi_c \equiv \frac{\rho}{\rho_s + \rho_a} w \chi_V, \tag{2.34}$$

and

$$\xi_t \equiv \rho t w \chi_V. \tag{2.35}$$

There is no weight reduction in this case, and w retains its initial value for the duration of the flight.

2.3.3 Specification of Tallies

This section briefly covers the practical implementation of the estimators described in the previous subsection. DEGAS 2 currently features three primary types of estimators. These estimators are used in accumulating three types of tallies. The tally types are

1. test,
2. reaction,

3. sector.

These designations can also be viewed as characterizing the nature of the information contained in the function g of Eq. (2.13). The following discussion of the three estimators applies to the first two of these; there is only one estimator for the sector tallies.

The first estimator, associated with the accumulating test particle information, is just the track length estimator (see also Sec. 2.3.2),

$$\xi_g^{\mathrm{TLE}} = w \frac{1 - \exp(-\sigma_i t_V)}{\sigma_i} g, \tag{2.36}$$

where t_V is the time the particle track spent in volume V, $w =$ weight at start of the time step. Compare this with Eq. (2.33). The function g can be a test particle attribute, such as mass, momentum, energy, or reaction-related, e.g., ion momentum source due to charge exchange. Volumetric plasma sources, such as the latter, enter the track-length estimator as averages over the background distribution[17].

The second type of estimator is the collision estimator,

$$\xi_g^{\mathrm{CE}} = \sum_{m_s} w_{m_s} \frac{g}{\sigma_s}, \tag{2.37}$$

where the sum is over m_s, all scattering collisions within V. The quantity w_{m_s} is the weight at m_sth collision (by using the instantaneous weight rather than the initial weight, the exponential weight reduction factor that appears explicitly in Eq. (2.31) is incorporated automatically). Again, g is a test particle attribute.

There is an analogous expression for the collision estimator associated with a particular reaction,

$$\xi_g^{\mathrm{CE}} = \sum_{m_j} w_{m_j} \frac{g}{\sigma_j}, \tag{2.38}$$

where the sum is now over m_j collisions of reaction j within V, The function g is some quantity related to the reaction and is likely something known only at collisions. One example of such a quantity would be the energy source due to H_2 dissociation (i.e., the energy exchange is determined only once the product velocities have been determined).

The third type of estimator arises from the observation that many integrals of interest can be written as:

$$I(g) \equiv \int d^3 r g(\vec{r}) \int d^3 v f(\vec{r}, \vec{v}) = \int d^3 r g(\vec{r}) n(\vec{r}). \tag{2.39}$$

This leads to the post-processing estimator. The requirement is that g not depend upon the test or background velocities. An example of this is the H_α emission rate.

Information accumulated when a flight crosses a diagnostic sector (surface) is compiled in the sector tallies. The stimator is just wg, where $w =$ is the weight when the track hits the sector. Examples of the $g =$ functions include mass, incident angle, and energy. This is effectively a collision estimator.

There is also a "null" or "none" estimator that can be used, say, to eliminate the contribution of a particular reaction to a tally without having to remove that reaction from the problem.

For each tally, we need to specify

1. A descriptive label,

2. The type of tally (test, reaction, or sector),

3. Its geometry, [volume (per zone), surface (diagnostic sector), or detector (e.g., chord-integrated)],

4. The dependent variable (e.g., energy, background momentum, H_α emission rate). The requested variables are matched directly against the labels in the atomic physics data files. If a new reaction and data file containing a new variable are added to the code, that variable can be used as the basis for a corresponding tally without having to modify the code.

5. The rank (or dimensionality) of the tally,

6. Its independent variables ($1 \rightarrow$ rank) [e.g., zone, test particle species, energy bin (diagnostic sector), wavelength bin (detector).

7. The estimator to be used (track-length, collision, post-processed, or null). For reaction tallies, the estimator chosen can vary with reaction. It may be advantageous to combine results from than one estimator, but no effort to implement such a capability has been made yet.

8. Any conversions that must be performed on the tally prior to output. Usually, these conversion will do some sort of scaling, e.g., by mass or volume. An important example is converting $\vec{v}$ from Cartesian to cylindrical coordinates when scoring.

2.4 Accumulation of Tallied Data

The estimators described in Sec. 2.3.3 are calculated by the routines in `score.web`, which are invoked during tracking. At the end of each flight, the resulting data are accumulated into the global arrays described in the statistics class (package abbreviation `sa`) via the subroutines in `stat.web`.

The data accumulation process takes place over a range of scales to optimize performance on massively parallel platforms. The basic breakdown is:

flt for "flight". This always refers to a single flight.

frag for "fragment". To facilitate load balancing, only a fraction (or "fragment") of the total number of flights per processor are sent at a time to each process. As each process completes, an additional fragment is sent until all have been completed.

fin for "final". In the simplest case, this refers to the global accumulation of data from all processes. Again, in the simplest case, the data are actually accumulated directly in an array from the output class (see also Sec. 3.12).

This process is complicated by two additional optimizations. First, as is documented in the statistics class, these arrays may be compressed, especially at the `flt` level, to reduce the amount of data transmitted. Second, in runs on larger machines (say, > 100 cores or processes) fragments are accumulated into intermediate "final" arrays over groups of processors to reduce the number of accumulations that must be performed by the master process, avoiding network transmission collisions. The machinery facilitating this is contained in `degasinit.web`.

The sources of neutral particles to be simulated in a given DEGAS 2 run are broken up into "groups" according to their physical nature, e.g., "plate recycling", "gas puff", etc., as described in the sources class. Because the number of flights and total particle current associated with each source group are specified independently by the user, the data for each group are accumulated separately and are available by themselves within the output arrays.

The fundamental expressions used in the data accumulation process are, thus, for a specific source group i. The other principal indexing parameters for the output arrays are the geometric zones, z, and the tallies, Sec. 2.3.3. To minimize confusion, we do not introduce any explicit index for the latter; the following expressions apply independently for each tally. The first accumulated moment, to

be associated with the mean in the end is:

$$S_1^i(z) = \sum_{j=1}^{N_i} \rho_j \left(\sum_{k \in z} \xi_{j,k} \right) / \sum_{j=1}^{N_i} \rho_j, \tag{2.40}$$

where: i is the source group index, the "1" refers to the first statistical moment, j runs over all of the N_i flights in group i, ρ_j is the relative statistical weight of flight j, k is the list of events scored by flight j (e.g., collisions) in zone z, and $\xi_{j,k}$ is the estimator value (as in Sec. 2.3.3) for this tally recorded by flight j during event k. The statistical weight $\rho_j \equiv 1$ in most cases, but can be set otherwise by the user to effect importance sampling via the input arrays `source_segment_rel_wt`. Note that this quantity is ***fixed*** for a given flight (indeed, is the same for all flights launched from that source segment) and is distinct from the statistical weight, designated by w in Secs. 2.3.1, 2.3.2, and 2.3.3, which may vary during flight tracking, but *always* has an initial value of $w = 1$.

The equivalent expression for the second moment is:

$$S_2^i(z) = \sum_{j=1}^{N_i} \rho_j \left(\sum_{k \in z} \xi_{j,k} \right)^2 / \sum_{j=1}^{N_i} \rho_j - \left[\sum_{j=1}^{N_i} \rho_j \left(\sum_{k \in z} \xi_{j,k} \right) / \sum_{j=1}^{N_i} \rho_j \right]^2. \tag{2.41}$$

While this expression is very close to that for the variance appearing in statistics textbooks [56], it is not suitable for the multi-level data accumulation used by DEGAS 2. How one would break up the sum in the numerator of Eq. (2.40) into smaller sets (fragments) of flights j is clear. To do likewise for the second moment, we need to instead work with:

$$\sum_{j=1}^{N_i} \rho_j \left(\sum_{k \in z} \xi_{j,k} \right)^2 / \sum_{j=1}^{N_i} \rho_j = S_2^i(z) + \left[\sum_{j=1}^{N_i} \rho_j \left(\sum_{k \in z} \xi_{j,k} \right) / \sum_{j=1}^{N_i} \rho_j \right]^2. \tag{2.42}$$

We now motivate the specific expressions used to accumulate data from an individual flight into a fragment, and a fragment into a final total. In the interest of clarity, we introduce some simplified notation: $\sum_{j=1}^{N_i} \rho_j \to w$, $S_1^i(z) \to S_1$, and $S_2^i(z) \to S_2$. The subscript i in the following represents "incremental" data (e.g., an individual flight), b represents the "base" data (e.g., the fragment containing data from earlier flights); the result of the accumulation will be new base data, indicated with a prime. The familiar form for the first moment is:

$$S'_{1b} = (w_i S_{1i} + w_b S_{1b})/(w_i + w_b), \tag{2.43}$$

with the new base weight being:

$$w'_b = w_i + w_b. \tag{2.44}$$

But, the code actually uses an equivalent form to maximize numerical accuracy:

$$S'_{1b} = S_{1b} + (S_{1i} - S_{1b})\frac{w_i}{w_i + w_b}. \tag{2.45}$$

The analogous expression following from Eq. (2.42) is:

$$(S'_{2b} + S'^2_{1b})w'_b = (S_{2b} + S^2_{1b})w_b + (S_{2i} + S^2_{1i})w_i. \tag{2.46}$$

Thus,

$$S'_{2b} = [(S_{2b} + S^2_{1b})w_b + (S_{2i} + S^2_{1i})w_i - S'^2_{1b}w'_b]/w'_b. \tag{2.47}$$

The equivalent form that is coded up in `stat.web` is:

$$S'_{2b} = [S_{2b}w_b + S_{2i}w_i + (S_{1i} - S_{1b})(S_{1i} - S'_{1b})w_i]/w'_b. \tag{2.48}$$

The equivalence of Eqs. (2.47) and (2.48) can be demonstrated most readily by adding and subtracting $S_{1b}(S_{1i} - S'_{1b})w_i$ inside the square brackets of Eq. (2.47) and substituting in Eq. (2.43). Again, this form improves numerical accuracy, e.g., in the case where $S_{1i} \simeq S_{1b}$.

The mean tally values for each source group are scaled by the total current in that group, which is obtained by summing over the group's $N_{\mathrm{seg},i}$ segments:

$$\Gamma_i = \sum_{\ell=1}^{N_{\mathrm{seg},i}} \Gamma_{i,\ell}. \tag{2.49}$$

In the steady state mode of operation, Γ_i represents a number of particles per second; in the time dependent mode, Γ_i is a number of particles. Obviously, these quantities are used in combining the $S^i_1(z)$ and $S^i_2(z)$ over source groups. But, they are also used to scale the user-specified relative statistical weights to get the ρ_j appearing in Eq. (2.40), etc. above and in setting the probabilities for sampling flights from the source segments.

Namely, the user can specify an arbitrary set of relative weights, $r_{i,\ell}$, for segments $\ell = 1 \to N_{\mathrm{seg},i}$ in group i. A normalization factor is computed for each source group:

$$r_{\mathrm{norm},i} = \left(\sum_{\ell=1}^{N_{\mathrm{seg},i}} \Gamma_{i,\ell}/r_{i,\ell}\right) / \left(\sum_{\ell=1}^{N_{\mathrm{seg},i}} \Gamma_{i,\ell}\right). \tag{2.50}$$

The cumulative probability distribution used in sampling flights in group i is, e.g., for segment ℓ:

$$p_{i,\ell} = \left(\sum_{m=1}^{\ell-1} \Gamma_{i,m}/r_{i,\ell} \right) / \left(\sum_{m=1}^{N_{\mathrm{seg},i}} \Gamma_{i,m}/r_{i,\ell} \right). \tag{2.51}$$

These probabilities are computed without the weight normalization factor Eq. (2.50) since it would appear in both numerator and denominator. The actual sampling procedure is more complex and offers additional flexibility; see the sources class and sourcetest.

If flight j in group i is sampled from segment i, the relative weight is set to:

$$\rho_j = r_{i,\ell} \times r_{\mathrm{norm},i}. \tag{2.52}$$

The final estimated value for a given tally in zone z is, for each individual group:

$$E^i(z) = \Gamma_i S_1^i(z) \tag{2.53}$$

In a time dependent run, this is divided by the simulated time interval $\Delta t = t_f - t_i$ so that the dimensions are the same as in a steady state run. This is then summed over all groups to get the total:

$$E(z) = \sum_{i=1}^{N_{\mathrm{groups}}} E^i(z). \tag{2.54}$$

The output quantity derived from the second moment S_2 is actually the relative standard deviation (see also Sec. 4.4). Namely, the expected error in a statistical estimate of the mean μ via N samples is $\sigma/\sqrt{N}$, where σ is the standard deviation and σ^2 is the variance. The relative standard deviation "rsd" is then

$$\mathrm{rsd} = \frac{\sigma}{\mu\sqrt{N}}. \tag{2.55}$$

Note that Eq. (2.41) and so on have $N \leftrightarrow \sum \rho_j$ in the denominator of the variance even though $N - 1$ appears in most textbook expressions (e.g., [56]). The "-1" accounts for the fact that the same sample is used to estimate the mean μ and the variance. While $N \gg 1$ in most DEGAS 2 runs, rendering the difference between N and $N - 1$ negligible, retaining $N - 1$ yields a maximum relative standard deviation of exactly unity when $\rho_j \equiv 1$ (a single score), facilitating interpretation of results. In the process of computing the relative standard deviation,

we multiply the S_2 by $N/(N-1)$, and let the N in this numerator be cancelled the N in the expression for the "rsd".

The "rsd" values required depend on the problem at hand and on location within the problem space. Some rough guidelines adapted from the manual for the MCNP [57] code can serve as a useful starting point:

rsd Range	Tally Quality
$0.5 \rightarrow 1$	Garbage
$0.2 \rightarrow 0.5$	Factor of a few
$0.1 \rightarrow 0.2$	Questionable
< 0.1	generally reliable

Iterative problems, such as with neutral-neutral scattering or coupling to a plasma code, demand precision, at least in high interaction regions, greater than that desired in the overall solution.

2.5 Tracking Procedure

The two previous subsections are tied together with an outline of the DEGAS 2 subroutine for following flights in Fig. 2.3.

The flight is initiated as it is sampled from the spatial and velocity distribution of a source. Conceptually, this step amounts to specifying $\chi_0(\vec{r}', \vec{v}', i)$ using the source term $Q(\vec{r}', \vec{v}')$. The subscript 0 denotes the initial track for the flight.

The next stage of the algorithm involves computing the reaction rates ρ_s, ρ_a, etc. for the flight. Continuing flights which require a recomputation (due to crossing from one plasma region to another or to a change in species or velocity) repeat the main loop starting at this same point. The post-collision density χ_n in Fig. 2.3 is intended to refer to them.

The time to the next collision is randomly sampled (Sec. 2.1.2) from the sum of the non-absorbing reaction rates using a random number ζ.

The weight w is compared with some minimum weight to guarantee that the flight terminates at some point. The "End flight" process indicated in Fig. 2.3 actually represents a Russian Roulette procedure[2] which may allow the flight to live a little longer.

The transport kernel T translates the flight from position $\vec{r}'$ to $\vec{r}$. At the same time, this step represents the conceptual transition from post-collision density to pre-collision density ψ, Eq. (2.8). The tracking algorithm stops when $t = t_{\max}$, the flight crosses a zone boundary (denoting a change in plasma parameters), or

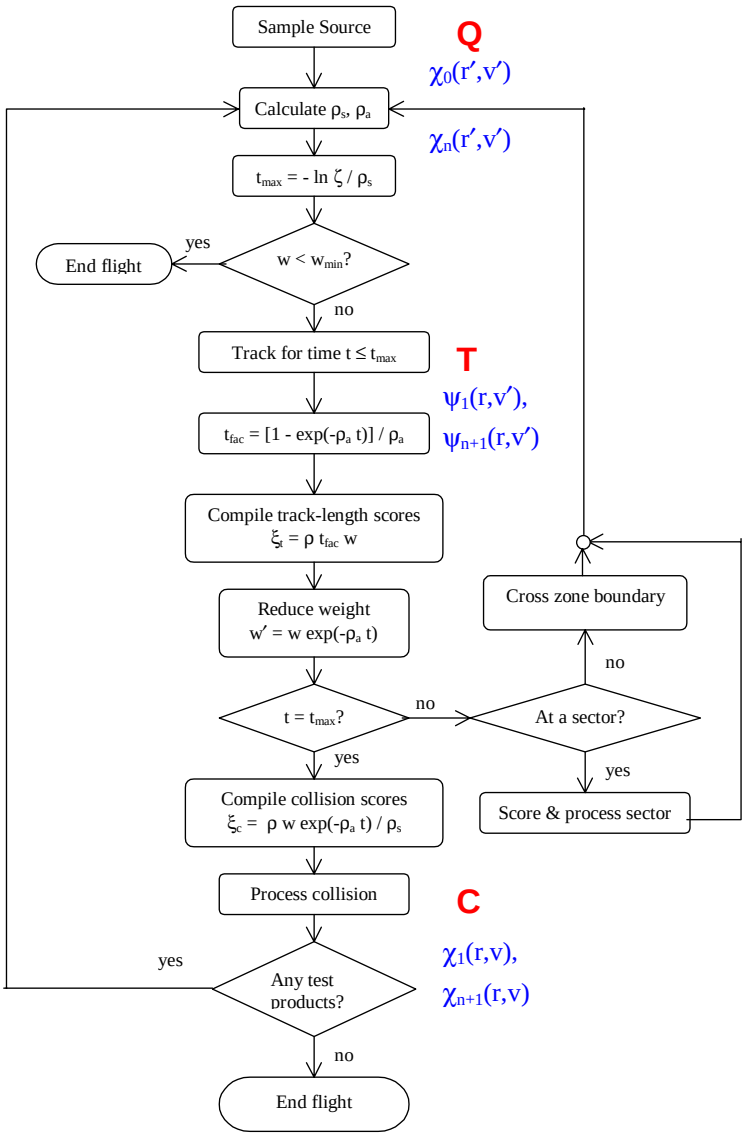

Figure 2.3: Simplified flow chart of DEGAS 2 subroutine `follow`, showing the steps involved in tracking a flight from its birth to its termination. Note the weight reduction, and collision and tracklength estimators (Sec. 2.3). The operators from the integral equations are shown along with the pre- and post-collision densities. These labels tie the practical algorithmic operations to the equations in Sec. 2.2.3. Some details have been omitted for clarity.

a "sector" (a material or diagnostic surface; these will be discussed in Sec. 2.7) is reached. If a zone boundary is crossed, the flight continues as is, although the rates must be recomputed. Processing a sector may result in the continuation or the termination of the flight, depending on the type of sector. Again, the details have been omitted for simplicity.

If the flight reaches $t = t_{\max}$, the implication is that it has had a collision. A particular collision is randomly chosen from amongst those possible for this particular flight. At this point, precollision scores are recorded and the collision itself is processed. Conceptually, this represents the transition via the operator C from ψ_{n+1} to χ_{n+1}. If the reaction chosen results in no particles to track (e.g., in no neutral species; particles which DEGAS 2 has been instructed to track are called "test" particles), the flight is terminated. Test particles resulting from the collision are tracked in the same manner, starting with the computation of the reaction rates.

2.6 Random Numbers

The concepts behind the random number generator used in DEGAS 2 are described in the file `random.web`. The documentation at the top of the file provides a nice introduction.

2.7 Geometry

The hiearchy of geometry-related objects in DEGAS 2 is, from the lowest level to the highest level, is:

1. surface,
2. cell,
3. polygon,
4. zone.

Individual cells are composed of surfaces, as discussed in the documentation for the internal geometry (e.g., see `geometry.web`; follow this link to the PDF version of this file). The next level up in two (and three, to a limited extent) dimensions is a polygon. Finally, a zone may consist of one or more polygons; properties

are constant across a zone. For example, a single zone might be used to represent the vacuum region around the plasma which is comprised of several (possibly disconnected) polygons. Or, a plasma flux surface on which density and temperature are constant might be a single zone. Note that polygons are used only for external interfaces to the geometry such as that provided by `definegeometry2d` or `readgeometry` and are not retained in the classes of the main code. Cells and surfaces are essentially used only by the code itself; the user would have to deal with them only in debugging. Zones are used primarily for the specification of input and output data. For more detailed background on the DEGAS 2 geometry and some guidance on how to set one up for your particular problem, see the introductions to the external geomtry interface codes `definegeometry` (PDF file here), `readgeometry` (PDF file here), or `boxgen` (less well documented).

Another important component of the geometry are the "sectors"; they are defined at interfaces between adjacent zones of different types to facilitate tracking or between zones of the same type for diagnostic purposes. Namely, flights are halted at sectors during tracking. If the next zone along the track represents, say, a solid material, the main code hands that flight off to the routines that effect the interaction of the current particle with that material. While stopped at that sector, scores to corresponding "diagnostics" (groups of sectors) are also tabulated. The essential identifying information of a sector are its surface and zone numbers. See the sector class documentation for a more detailed description. Default diagnostic sector groups defined by all DEGAS 2 geometry setup programs are discussed in Sec. 3.11.

2.8 Symmetry and Coordinate Systems

The coordinate system in the main DEGAS 2 code is purely 3-D Cartesian; that's essential to an efficient tracking algorithm. There are, however, two coordinate system related concepts.

1. The geometry's symmetry. In most problems, the user's objectives can be met with an approximate 2-D solution possessing an ignorable coordinate. The two situations considered in DEGAS 2 are invariance to translation along the y axis ("plane" symmetry) and invariance to rotation about the z axis ("cylindrical" symmetry). In both cases, the tracking is still done in 3-D, but the output results are averaged over the third dimension. As will be discussed in more detail below, 1-D and 3-D "symmetries" are also

permitted. Most geometry related variables refer to Cartesian coordinates. However, in cylindrically symmetric or nearly symmetric (3-D) cases, some geometry related variables are in cylindrical coordinates.

2. Coordinate system of input and output velocity fields. The main need for this is in cylindrically symmetric or nearly symmetric problems, e.g., when DEGAS 2 is coupled to a 2-D plasma code like UEDGE. In these cases, the plasma flow velocities are effectively specified at a toroidal angle $\phi = 0$, corresponding to $y = 0$. To use these velocities in a collision at some particular location for which $\phi \neq 0$ (equivalently, $y \neq 0$), the velocities need to be rotated through that angle ϕ. Likewise, a code like UEDGE expects to get from DEGAS 2 toroidally symmetric ion momentum sources specified at $\phi = 0$. So, the scores for those sources need to be rotated from the local toroidal angle $\phi \neq 0$ to $\phi = 0$ as they are being compiled. These two transformations are handled by macros in the background class. In problems with plane or no symmetry, these macros have no effect.

The currently permissible symmetry types (see also `geometry.hweb`) are:

None Currently indicates that the symmetry has not been defined, *not* that the geometry has no symmetry.

Plane A 2-D geometry in which the y coordinate is ignored.

Cylindrical A 2-D geometry in which the toroidal angle ϕ is ignored.

1-D A 1-D (plane) geometry in which the y and z coordinates are ignored.

3-D Plane This geometry is close to be planar symmetric; all of the geometrical objects are defined using 2-D shapes defined in a single plane. The problem is bounded in the y direction, and the y variation of the results can be computed and output. The nature (solid vs. plasma or vacuum) of the objects can also vary in the y direction. The problem is divided in the y direction by $y =$ constant surfaces. This might be described as a "bread slice" approach to a 3-D geometry.

3-D Cylinder This is the cylindrical equivalent of the 3-D Plane symmetry. The full range of toroidal angles (360 degrees or 2π radians) is included in this case. Obviously, the problem is divided in the ϕ direction by $\phi =$ constant surfaces. The analogous simplified characterization of this approach is the "pie slice" technique.

3-D Cylindrical Section This is also a cylindrical equivalent of the 3-D plane symmetry. The difference is that in this case only a finite range of toroidal angles is considered. The problem is bounded by ϕ = constant surfaces at the extrema of this range. Extending the "pie slice" analogy, this case represents some number of slices, but less than the whole pie!

2.9 Time Dependence

The default and most frequently used mode of running DEGAS 2 simulates a steady state, time independent system. The reason for this is physically obvious: unconstrained by magnetic field lines, neutral particles can travel through a plasma much more quickly than charged species, and in many cases, the neutral profile equilibrates on a time scale short compared with global transport times. For a rapidly evolving system, however, a time dependent calculation may be needed. The implementation of time dependence in DEGAS 2 follows that used in EIRENE [5].

Only the `defineback` input preprocessor is presently set up to handle direct user specification of a time dependent run; see the corresponding documentation for details on how this is done. However, the user can in principle also manually set the controlling parameters in the source class.

The user specified sources in a time dependent run are either fixed in time during a time interval (uniform) or start at the beginning of the interval (δ function). More complex time variation of the sources can be created by breaking up the source time dependence into a set of discrete time intervals with the sources being uniform over each.

To provide continuity between consecutive time intervals, an additional "snapshot" source group is set up to sample from the neutral particle distribution function (PDF) that was in the volume at the end of the end of a previous interval. Again following [5], we represent this PDF via the particle positions, velocities, and weights; this is the purpose of the snapshot class. These data are written to the file indicated by the `snapshotfile` in `degas2.in` (Sec. 3.5.1). With this approach, the amount of kinetic detail in the PDF is maximized.

If the user is truly solving an initial value problem and wishes to track the evolution of the neutral density in time from a null initial state, e.g., the temporal evolution of a gas puff from the start, no initial snapshot file is required. A more typical application, however, is to follow the common evolution of the neutral and plasma behavior at the same time from some nominal initial state. To do that, we

need a snapshot distribution representative of that initial state.

To generate such a distribution, the code is run in a special mode that is similar to a standard, steady state simulation in that the plasma background is fixed and the flights are tracked until they are ionized or exit the simulation volume. However, the path of each flight is broken up into a set of uniform time steps of the same size as those used in the actual time dependent calculation. The start time of a new flight coming from the gas source (e.g., a gas puff) is uniformly sampled from the initial time interval. The flight is tracked through the remainder of that interval; at the end, its characteristics are deposited in the snapshot distribution. The flight is then tracked forward from that point through another time interval, resulting in an additional entry to the snapshot distribution. The process continues in this manner until the flight is effectively ionized or leaves the volume. The end result is a snapshot distribution that fills the volume and is statistically equivalent to a steady state calculation. The number of snapshot particles generated per flight varies with the problem and is, actually, a measure of the average neutral density in the volume. Consequently, the user should perform a few scoping runs with a limited number of flights to estimate this ratio and determine the number of flights required to yield the desired snapshot population.

Sampling from a large snapshot distribution in a parallel run of `flighttest` can slow the initial broadcast of the source information to the slave processes and stress memory resources on those cores. The `MASTER_SAMPLE` switch in the `Makefile.local` file can be set to "yes" to allow sampling to be done instead on the master process so that only the flight data need be broadcast to the slaves. For additional information, see the documentation of the associated flight_frag class.

2.10 Atomic Physics

The interaction of electrons with atoms and ions in a divertor plasma is complicated by having comparable collision and radiative decay times. The techniques for treating such systems were first described by Bates and McWhirter[6, 7]. The physical processes included in such a collisional-radiative model are:

1. Electron collisional excitation $K_{e,mn}$ and de-excitation $K_{d,mn}$,
2. Spontaneous radiative transitions A_{mn},
3. Electron collisional ionization $K_{i,m}$,

4. Three-body recombination $K_{r,mn}$,

5. Radiative recombination $\beta_{rad,m}$,

6. Dielectronic recombination (for multi-electron atoms) $\beta_{dia,m}$.

where m and n are principal quantum numbers. For the two state terms, the initial state is the *second* number; the final is the *first*. This apparently backward notation allegedly has a mathematical origin.

A set of $m \geq 1$ equations describes the balance between these processes that determines the density of the m-th excited state of the atom, N_m,

$$\begin{aligned}\frac{dN_m}{dt} &= -N_m \left[n_e \left(K_{i,m} + \sum_{n>m} K_{e,nm} + \sum_{n<m} K_{d,nm} \right) \right. \\ &\qquad \left. + \sum_{n<m} A_{nm} \right] \\ &\quad + n_e \left(\sum_{n>m} N_n K_{d,mn} + \sum_{n<m} N_n K_{e,mn} \right) \\ &\quad + \sum_{n>m} N_n A_{mn} + n_e n_i (\beta_{rad,m} + \beta_{dia,m} + n_e K_{r,m}) \\ &\quad + n_e n_{H_2} E_{H_2,m} + n_e n_{H_2^+} (E_{H_2^+,m} + \alpha_m), \end{aligned} \tag{2.56}$$

where n_e is the electron density and n_i is the density of ions corresponding to the atom under consideration ($m \to \infty$, effectively). The two terms at the end account for atoms coming from the dissociation of H_2 and H_2^+ ($E_{H_2,m}$ and $E_{H_2^+,m}$) and dissociative recombination of H_2^+ (α_m).

The first step in developing a collisional radiative model involves breaking up Eq. (2.56) into a single equation for the $m = 1$ ground state and a set of equations for all of the excited states, $m > 1$. Writing the latter as a matrix equation for $m, n \geq 2$:

$$\begin{aligned}\frac{dN_m}{dt} &= n_e [Q_{mn} N_n + N_1 K_{e,m1} + n_i (\beta_m + n_e K_{r,m}) \\ &\qquad + n_{H_2} E_{H_2,m} + n_{H_2^+} (E_{H_2^+,m} + \alpha_m)], \end{aligned} \tag{2.57}$$

where the $K_{e,mn}$, $K_{d,mn}$, A_{mn}, and $K_{i,m}$ contributions have been lumped into the Q_{mn} matrix. Its diagonal terms represent sinks of N_m while the off-diagonal elements are sources from the other excited states. The remaining terms in Eq. (2.57)

are sources coming from the ground state, $\propto N_1$; ions, $\propto n_i$; molecules, $\propto n_{H_2}$; and molecular ions, $\propto n_{H_2^+}$.

The collisional radiative approximation[6, 7] consists of assuming that the excited states equilibrate amongst themselves much more quickly than the ground state evolves so that $\frac{dN_m}{dt} \rightarrow 0$; transport of the excited states is thus ignored. In contrast, interactions of the ground state with the excited states occur on time scales comparable to those of transport; simulating these processes consistently to obtain the N_1 in Eq. (2.57) is the job of the neutral transport code. The collisional radiative approximation allows the inversion of the $m \geq 2$ matrix equation with these various source terms, yielding a set of N_m for each:

$$N_m = \left(\frac{N_m^{(i)}}{N_1}\right) N_1 + \left(\frac{N_m^{(ii)}}{n_i}\right) n_i + \left(\frac{N_m^{(iii)}}{n_{H_2}}\right) n_{H_2} + \left(\frac{N_m^{(iv)}}{n_{H_2^+}}\right) n_{H_2^+}, \tag{2.58}$$

where:

$$\frac{N_m^{(i)}}{N_1} = -(Q^{-1})_{mn} K_{e,n1} \tag{2.59}$$

$$\frac{N_m^{(ii)}}{n_i} = -(Q^{-1})_{mn} (\beta_n + n_e K_{r,n}) \tag{2.60}$$

$$\frac{N_m^{(iii)}}{n_{H_2}} = -(Q^{-1})_{mn} E_{H_2,n} \tag{2.61}$$

$$\frac{N_m^{(iv)}}{n_{H_2^+}} = -(Q^{-1})_{mn} E_{H_2^+,n} - (Q^{-1})_{mn} \alpha_n. \tag{2.62}$$

In practice, the solution of these equations involves first obtaining the various reaction rates, $K_{e,mn}$, $K_{d,mn}$, A_{mn}, $K_{i,mn}$, $K_{r,mn}$, $\beta_{rad,m}$, $\beta_{dia,m}$, as a function of n_e and T_e. At each n_e, T_e pair, the inverse of the Q_{mn} matrix is computed and then used to generate each of the population coefficients above. Further discussion of the molecular terms is deferred to the descriptions of the collisional radiative versions of the molecular dissociation, ionization, and dissociative recombination processes.

Instead focusing on only the ground state and ion terms in Eq. (2.58), the $m = 1$ equation can becomes

$$\frac{dN_1}{dt} = -N_1 n_e S_{eff} + n_e n_i R_{eff} + \text{transport, etc.}, \tag{2.63}$$

with

$$S_{eff} = \sum_{m \geq 1} \left(\frac{N_m^{(i)}}{N_1}\right) K_{i,m}, \tag{2.64}$$

and

$$R_{eff} = -\sum_{m\geq 2}\left(\frac{N_m^{(ii)}}{n_i}\right) K_{i,m} + \sum_{m\geq 1}\left(\beta_{rad,m} + \beta_{dia,m} + n_e K_{r,m}\right). \qquad (2.65)$$

S_{eff} is thus the effective ionization rate of the ground state neutral, incorporating the multi-step processes associated with all of the excited states. Likewise, R_{eff} is the effective recombination rate. Both are functions of n_e and T_e. Selected values of the population coefficients $N_m^{(i)}/N_1$ and $N_m^{(ii)}/n_i$ may also be tabulated or used to compute photon emission rates for specific transitions of interest (e.g., Balmer-α).

The above assumes that just the ground state evolves on the transport time scales relevant to DEGAS 2. If metastable excited states are present in the system (e.g., helium's 2^1S and 2^3S states or hydrogen's $n = 2$ in an optically thick region), they may need to be treated on this same footing. The need for such a treatment and its formulation go beyond the scope of this presentation. See the discussion below on helium for some additional references.

Computing the electron energy sources and sinks associated with Eq. (2.56) involves accounting for the energy exchanges in each of the processes appearing in Eq. (2.56):

$$\begin{aligned}\frac{dE_e}{dt} &= E_1\left(-N_1 n_e S_{eff} + n_e n_i R_{eff}\right) \\ &\quad - \sum_{\substack{m,n \\ n<m}} N_m A_{nm}\left(E_n - E_m\right) \\ &\quad - n_e n_i \sum_{m\geq 1} \beta_{rad,m}\left(\langle E_{rad,m}\rangle + E_m\right) \\ &\quad - n_e n_i \sum_{m\geq 1} \beta_{dia,m}\left(\langle E_{dia,m}\rangle + E_m\right), \qquad (2.66)\end{aligned}$$

where

- E_1 is the ionization energy,
- the first term is the loss due to ionization,
- the second is the gain due to recombination.
- The sums represent lost photons from radiative transitions and recombination,

- $\langle E_{rad,m} \rangle$ and $\langle E_{dia,m} \rangle$ are the average energies of electrons in radiative and dielectronic recombination, respectively.

The two average energies are computed using the expression given by Reiter[17],

$$\langle E_{x,m} \rangle = T_e \left[\frac{3}{2} + \frac{T_e}{\beta_{x,m}} \frac{d\beta_{x,m}}{dT_e} \right], \tag{2.67}$$

where x denotes either rad or dia.

Using Eq. (2.58), Eq. (2.66) becomes

$$\frac{dE_e}{dt} = -E_1 \left(N_1 n_e S_{eff} - n_e n_i R_{eff} \right) - N_1 E_{\text{loss}}^{(i)} - n_i E_{\text{loss}}^{(ii)}, \tag{2.68}$$

where the terms $E_{\text{loss}}^{(i)}$, $E_{\text{loss}}^{(ii)}$ represent the loss rates due to coupling to the ground state and continuum, respectively. A different arrangement of these terms which can be generalized to the case involving multiple "transported states" has also been used:

$$\frac{dE_e}{dt} = -N_1 n_e E_{\text{loss}}^{(i)\prime} + n_i n_e E_{\text{source}}^{(ii)\prime} - n_i n_e E_{\text{loss}}^{(ii)\prime}. \tag{2.69}$$

The relationships between the terms are nearly obvious, but potentially prone to confusion. For clarity, they are:

$$E_{\text{loss}}^{(i)\prime} = E_1 S_{eff} + E_{\text{loss}}^{(i)}/n_e, \tag{2.70}$$

$$E_{\text{source}}^{(ii)\prime} = E_1 R_{eff}, \tag{2.71}$$

$$E_{\text{loss}}^{(ii)\prime} = E_{\text{loss}}^{(ii)}/n_e. \tag{2.72}$$

Note that the "unprimed" energies have dimensions of power, and the "primed" versions have dimensions of power times volume. All are stored in tabular form as functions of n_e and T_e for use in DEGAS 2.

2.10.1 Hydrogen Collisional Radiative Model

The code we use to solve these equations for the hydrogen atom, `collrad`, is based on the one described by Weisheit[20]. The original code assumed that ionization and recombination were in balance so that the neutral density could be expressed in terms of the ion density. While this might be valid in an astrophysical context, it is horribly wrong in fusion devices. The first major change to the code is to have it use Eq. (2.58) instead. The other change is to replace the

cross sections with those provided in the Janev-Smith database[18]. The resulting collisional-radiative data are believed to be in good agreement with ADAS. Although the `collrad` code is not itself distributed with DEGAS 2, it can be obtained from the DEGAS 2 authors (see Sec. 5.1).

The most recent data produced by the `collrad` code are contained in the file `ehr5.dat`, which are then incorporated in the `hionize5.nc` and `hrecombine5.nc` files via the `reactionwrite` code. Older versions of this file are included with DEGAS 2 for the purpose of verification, including with the original DEGAS code. The corresponding ionization netCDF files in the `degas2/data` directory are noted in parentheses; the recombination files are numbered in the same way.

`eh.dat` The file generated by Weisheit's original code.

`ehr1.dat` The result of modifying Weisheit's code to use Eq. (2.58) (`hionize1.nc`).

`ehr2.dat` Comprehensive update of the `collrad` code, primarily the switch to the Janev-Smith database[18] (`hionize2.nc`).

`ehr3.dat` Minor modifications associated with the removal of an erroneous factor of 0.75 multiplying A_{12} (`hionize.nc`).

`ehr4.dat` Were *not* produced by `collrad`, but via ADAS. Flaws in these calculations were subsequently identified; these data should *not* be used. Note that these data are distinct from the hydrogen data one would obtain from either the ADAS or OpenADAS libraries (`hionize4.nc`; at this point the authors decided to increment the number in the netCDF file name rather than continue to modify `hionize.nc`).

`ehr5.dat` Produced by the `collrad` code following "convergent close coupling" (CCC) computed updates to the $n = 1$, 3, 4, and 5 excitation rates; the fits to these cross sections were provided by Janev and Reiter (`hionize5.nc`).

See also Sec. 2.12.

The `ehr5.dat` file contains

1. Ionization rate S_{eff} in cm^3 s^{-1},
2. Recombination rate R_{eff} in cm^3 s^{-1},
3. Neutral electron losses $E^{(i)}_{\text{loss}}$ in erg s^{-1},

4. Continuum electron losses $E_{\rm loss}^{(ii)}$ in erg $\rm s^{-1}$,
5. Neutral "n=3 / n=1", $N_3^{(i)}/N_1$,
6. Continuum "n=3 / n=1", $N_3^{(ii)}/n_i$,
7. Neutral "n=2 / n=1", $N_2^{(i)}/N_1$,
8. Continuum "n=2 / n=1", $N_2^{(ii)}/n_i$,
9. Neutral "n=4 / n=1", $N_4^{(i)}/N_1$,
10. Continuum "n=4 / n=1", $N_4^{(ii)}/n_i$,
11. Neutral "n=5 / n=1", $N_5^{(i)}/N_1$,
12. Continuum "n=5 / n=1", $N_5^{(ii)}/n_i$,
13. Neutral "n=6 / n=1", $N_6^{(i)}/N_1$,
14. Continuum "n=6 / n=1", $N_6^{(ii)}/n_i$,
15. Neutral "n=7 / n=1", $N_7^{(i)}/N_1$,
16. Continuum "n=7 / n=1", $N_7^{(ii)}/n_i$,
17. Neutral "n=8 / n=1", $N_8^{(i)}/N_1$,
18. Continuum "n=8 / n=1", $N_8^{(ii)}/n_i$,
19. Neutral "n=9 / n=1", $N_9^{(i)}/N_1$,
20. Continuum "n=9 / n=1", $N_9^{(ii)}/n_i$.

Figure 2.4 shows the variation of the effective ionization and recombination rates with electron density and temperature.

The portion of the file for each quantity consists of 15 separate sections (labeled by `jn`), corresponding to the various electron density values,

$$n_e(\texttt{jn}) = 10^{10+(\texttt{jn}-1)/2}\ \mathrm{cm}^{-3}. \tag{2.73}$$

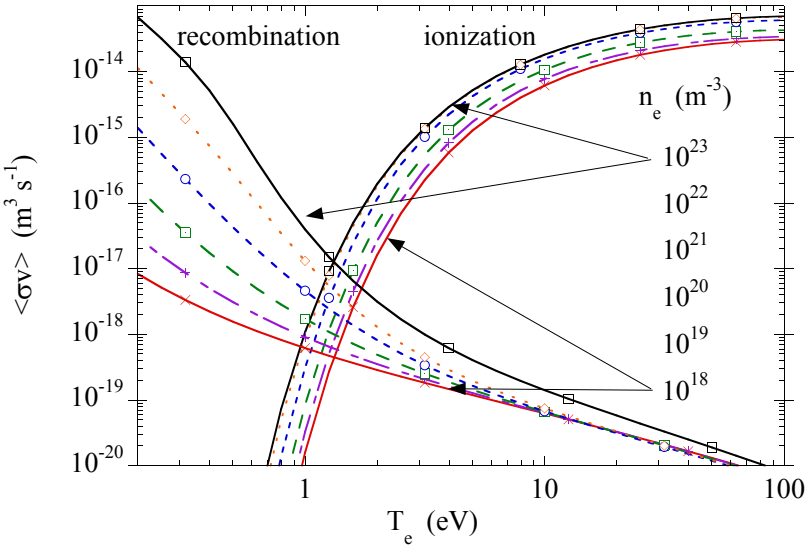

Figure 2.4: Effective hydrogen ionization and recombination rates as a function of electron temperature for selected electron densities. *Note that these are actually data from* `ehr2.dat`. *The most recent data differ only quantitatively.*

Each section consists of 60 data values, one for each $T_e(\texttt{jt})$,

$$T_e(\texttt{jt}) = 10^{-1.2+(\texttt{jt}-1)/10}\,\text{eV}. \tag{2.74}$$

The index `jt` starts at 1 in the first column and row and proceeds row-by-row. There are 10 rows of 6 entries each.

The density of the $n = 3$ excited state can be found with

$$N_3 = \left(\frac{N_3^{(i)}}{N_1}\right) N_1 + \left(\frac{N_3^{(ii)}}{n_i}\right) n_i. \tag{2.75}$$

The corresponding emission rate for the Balmer-α line (a.k.a. "H_α", $n = 3 \rightarrow 2$) is given by $N_3 A_{23} = 4.41 \times 10^7 N_3$ cm^{-3} s^{-1}. Note that the energy of this transition is $E_{23} = 13.595(2^{-2} - 3^{-2})$ eV. Similar expressions give the $n = 2$ density and the Lyman-α ($n = 2 \rightarrow 1$) emission rate.

The data in `ehr5.dat` are transformed into netCDF files containing the particular variables DEGAS 2 expects to find (see Sec. 3.9) by the `reactionwrite` code. Separate files for ionization and recombination are written. In these files, S_{eff} and R_{eff} become the "reaction rates". The "emission rate" variable is obtained by rewriting the above expression for the Balmer-α emission rate as a power, per atom (for ionization) or ion (for recombination) and per electron:

$$P_\alpha^{(i)} = E_{23} A_{23} \left(\frac{N_3^{(i)}}{N_1}\right) / n_e, \tag{2.76}$$

$$P_\alpha^{(ii)} = E_{23} A_{23} \left(\frac{N_3^{(ii)}}{n_i}\right) / n_e. \tag{2.77}$$

These quantities have dimensions of power times volume. The wavelengths of this transition for each of the hydrogen isotopes are also inserted into the file, $\lambda_{H_\alpha} = 6562.80$ Å, $\lambda_{D_\alpha} = 6561.04$ Å, and $\lambda_{T_\alpha} = 6560.45$ Å. The electron energy exchange terms in `ehr5.dat` correspond to those in Eq. (2.68); `reactionwrite` transforms them into those of Eq. (2.69).

2.10.2 Helium Collisional Radiative Model

The code used to solve the collisional radiative equations for the helium atom was developed by Goto[21], based on Fujimoto's original[22]. This code actually generates data for two collisional radiative models. In the first, only the ground

state (1^1S) atom is considered slowly varying (i.e., must be treated explicitly as a transported species in DEGAS 2); this model was designated as "formulation II" by Fujimoto. In the second, the metastable 2^1S and 2^3S states are also considered slowly varying; Fujimoto called this "formulation I". The appropriate species, reactions, and data needed to do simulations based on this model have been incorporated into DEGAS 2. However, investigations using it have not demonstrated that the added detail yields any significant differences in the final results. Moreover, a separate analysis of the helium collisional radiative equations [Eq. (2.56)] suggests that this particular model may not even be valid for typical fusion edge conditions. For these reasons, we will not discuss this model any further and will focus only on the simpler "one state" model.

The primary modification made to Goto's code in preparation for using its output with DEGAS 2 were to add the computation of the electron energy exchange terms in Eq. (2.69). The resulting formulation II model and data are, not surprisingly, very similar in structure to those of the hydrogen collisional radiative model. The file `he_crII_alad.dat` (in directory `Aladdin/data`) is an Aladdin formatted version of these data. These same data written in a format more like that of `ehr2.dat` are contained in the file `he_crII.txt` that can be downloaded from the "Data" section of the DEGAS 2 web site. This file has been provided as a convenience for those who would just like to use the data, but not download the entire DEGAS 2 code.

The contents of this file are:

1. Ionization rate S_{eff} in cm^3 s^{-1},
2. Neutral electron losses $E^{(i)\prime}_{\rm loss}$ in eV cm^3 s^{-1},
3. Neutral 5877 Å ($3^3D \rightarrow 2^3P$) emission rate $P^{(i)}_{5877}$ in eV cm^3 s^{-1},
4. Neutral 6680 Å ($3^1D \rightarrow 2^1P$) emission rate $P^{(i)}_{6680}$ in eV cm^3 s^{-1},
5. Recombination rate R_{eff} in cm^3 s^{-1},
6. Continuum electron losses $E^{(ii)\prime}_{\rm loss}$ in eV cm^3 s^{-1},
7. Continuum electron sources $E^{(ii)\prime}_{\rm source}$ in eV cm^3 s^{-1},
8. Continuum 5877 Å ($3^3D \rightarrow 2^3P$) emission rate $P^{(ii)}_{5877}$ in eV cm^3 s^{-1},
9. Continuum 6680 Å ($3^1D \rightarrow 2^1P$) emission rate $P^{(i)}_{5877}$ in eV cm^3 s^{-1},

The portion of the file for each quantity consists of 30 separate sections (labeled by `jn`), corresponding to the various electron density values,

$$n_e(\texttt{jn}) = 10^{8+8*(\texttt{jn}-1)/29}\ \mathrm{cm}^{-3}. \tag{2.78}$$

Each section consists of 65 data values, one for each $T_e(\texttt{jt})$,

$$T_e(\texttt{jt}) = 10^{(\texttt{jt}-1)/16}\ \mathrm{eV}. \tag{2.79}$$

The index `jt` starts at 1 in the first column and row and proceeds row-by-row. There are 10 rows of 6 entries each, followed by a single row of 5 entries.

The "emission rates" here correspond to the power in those lines and are analogous to Eqs. (2.76) and (2.77) above. The total emitted power (per unit volume) is obtained at run time by combining them with the ground state and ion densities:

$$P_{5877} = P^{(i)}_{5877} N_{1^1S} n_e + P^{(ii)}_{5877} n_i n_e. \tag{2.80}$$

Note that $E_{3^3D\to 2^3P} = 2.10955$ eV, $A_{3^3D\to 2^3P} = 7.069\times 10^7$ s^{-1}, and $E_{3^1D\to 2^1P} = 1.85606$ eV, $A_{3^1D\to 2^1P} = 6.369 \times 10^7$ s^{-1}. The more precise (vacuum) wavelengths for these transitions are 5877.1 Å and 6679.7 Å respectively. The first ionization energy for helium is $E_1 = 24.587$ eV.

The electron energy losses and sources can be used directly in Eq. (2.69).

The data in `he_crII_alad.dat` are transformed into netCDF files by the `ratecalc` code using the input files `heionII.input` and `herecII.input` (in the `data` directory). See the `DATA_HISTORY` file for additional details.

Recent re-examination by D. Reiter, the code authors, and others of the cross section fits input to the Goto code have identified some shortcomings that do quantitatively impact the collisional radiative rates. We are awaiting updates to the cross sections and the associated fits. Users with rate sensitive applications should contact the code authors (see Sec. 5.1) regarding the status of this work.

2.11 Elastic Scattering

2.11.1 Neutral-Ion Elastic Scattering

Note: this subsection was extracted from Ref. [23]. A few minor changes in notation have been made for consistency. Only limited experimental data are available for ion-neutral elastic scattering at low energies. The monograph by Janev[16] is one of the most widely used data sources available, but only contains charge

exchange at moderate to high collision energies and no data for elastic scattering. As divertor technology has progressed, low temperature operating environments have become a standard mode of operation. Under these conditions, a purely classical treatment of charge exchange is no longer sufficient. In particular, below approximately 2 eV for $D^+ + D$ collisions, the processes of elastic scattering and and resonant charge exchange can no longer be distinguished due to quantum mechanical effects. The need for a comprehensive treatment of ion-neutral elastic collisions at low energies motivated the Controlled Fusion Atomic Data Center at ORNL to produce a collection of total differential elastic scattering cross sections for 31 various isotopic combinations of hydrogen atoms and molecules[24].

DEGAS 2 represents an improvement over the original DEGAS code in that the pseudo-collision algorithm originally employed in scoring results[3] is replaced with a track-length type estimator. This allows efficient determination of momentum and energy exchange with the background plasma over a wide range of background densities. The rate expressions used by the track-length estimator involve the momentum transfer (or diffusion) cross section, σ_d[25].

A general expression for the integral cross sections is[25]

$$\sigma_l = 2\pi \int_0^\pi (1 - \cos\theta)^l \, \sigma(\theta, E_r) \sin\theta \, d\theta, \tag{2.81}$$

where $l = 0$ corresponds to the total elastic scattering cross section, $l = 1$ to the diffusion cross section and $l = 2$ to the viscosity cross section; θ is the center-of-mass scattering angle, E_r is the relative energy of the collision, and $\sigma(\theta, E_r)$ is the differential elastic scattering cross section. The total elastic collision cross section, σ_t, is used in Monte Carlo simulations to determine the time between elastic collisions. In this manner the size of the total elastic collision cross section influences the execution time of numerical simulations.

A simulation technique that reduces the total collision cross section while keeping the momentum transfer cross section invariant can be developed. Using the expression for σ_d and dividing Eq. (2.81) into small and large angle components,

$$\sigma_d(E_r) = 2\pi \int_0^{\theta_{\min}} C(E_r)(1-\cos\theta)\delta(\theta-\theta_{\min})d\theta + 2\pi \int_{\theta_{\min}}^\pi (1-\cos\theta)\sigma(\theta, E_r)d\theta. \tag{2.82}$$

Substituting $\sigma(\theta, E_r)$ from the ORNL data set here and directly computing $\sigma_d(E_r)$ [via Eq. (2.81)], allows Eq. (2.82) to be solved for $C(E_r)$ given a particular value of $\theta_{\min}$. The resulting value of $C(E_r)$ is then used in a similar expression for the viscosity cross section. This provides viscosity cross sections accurate to second

order in $\theta_{\min}$. The value of $\theta_{\min}$ is chosen so as to reduce the total elastic cross section as much as possible while retaining physical realism.

For low energy elastic collisions a significant portion of the collisions involve small-angle deflections. Due to this behavior, the majority of the reduction in total elastic cross section is obtained with a five-degree minimum scattering angle (Fig. 2.5). Minimum scattering angles greater than five degrees result in little computational savings. The resulting reduction in σ_t is approximately a factor of two at relative energies $E_r > 10^{-1}$ eV. The values for σ_d are unchanged during application of a small-angle cut-off.

The use of this small-angle cut-off leads to an efficient algorithm for simulating the dynamics of elastic collisions in Monte Carlo simulations. Previous models based on classical collision kinetics for direct elastic scattering face difficulties in that the cross section is a rapidly varying function of the impact parameter[25]. This precludes the use of efficient table-look-up methods and forces a time consuming evaluation of the collision integral for each individual interaction.

Utilization of the differential elastic cross section, $\sigma(\theta, E_r)$, allows development of a table-look-up interpolation for both direct elastic scattering and resonant charge exchange. Following the work by Abou-Gabal and Emmert[26], we divide the total elastic cross section into n pieces and interpret the result as a cumulative probability distribution function,

$$P_i = 2\pi \int_{-1}^{\cos\Theta_i} \frac{\sigma(\theta, E_r)}{\sigma_t(E_r)}\, d(\cos\theta),\ i = 0, 1, \ldots n. \tag{2.83}$$

Equation 2.83 is inverted to obtain the scattering angle as a function of relative energy and n equally spaced values of the cumulative probability spacing, $\Theta_i(P_i, E_r)$, where Θ_i is the scattering angle in the center-of-mass frame. Examples of $\Theta_i(P_i, E_r)$ for the $D^+ + D$ and $D^+ + D_2$ collisions at a center-of-mass energy of 1 eV are presented in Fig. 2.6. At this energy, inclusion of direct elastic scattering increases the probability that colliding particles undergo small angle deflections. Consideration of charge exchange alone would shift the probability distribution to favor collisions resulting in $\Theta = \pi$. As evident in Fig. 2.6, $\Theta_i(P_i, E_r)$ for like-species collisions exhibits a higher probability of undergoing large-angle collisions compared to the ion-molecular collision case. This is due to the absence of a charge exchange contribution to the scattering function for ion-molecular elastic collisions. Monte Carlo collision processing begins with the sampling of a suitable ion from a Maxwellian distribution at the local ion temperature, yielding, E_r. The P_i are next sampled from a uniform distribution. The

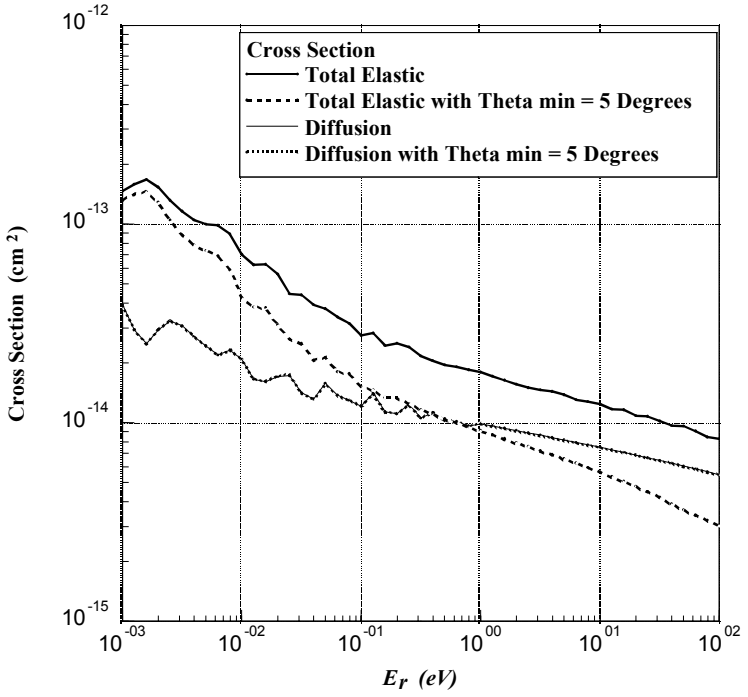

Figure 2.5: Comparison of total and diffusion cross sections for $D^+ + D$ collisions with and without the small-angle cut-off approximation.

collision angle follows from a bi-linear interpolation into this data table without evaluation of complex integral functions.

Previous treatments of charge exchange in Monte Carlo algorithms employed a 180° collision scattering model in the center-of-mass frame. The approach is simple to implement, requiring just the exchange of the ion and neutral velocity vectors and yields a momentum transfer cross section of about $2\sigma_{cx}$, where σ_{cx} is the charge exchange cross section. This is, in fact, roughly equal to the total diffusion cross section in ion-neutral collisions, $\sigma_d \sim 2\sigma_{cx}$ [27, 28, 29] The accuracy of this simple expression is demonstrated in Fig. 2.7, which compares the diffusion cross section with results for the "spin exchange" cross section from the ORNL data[24]. Spin exchange is defined as the exchange of the intrinsic quantum mechanical property of particle spin; for distinguishable particles this is equivalent to charge exchange. For like-species collisions, spin exchange and charge exchange are equivalent only within classical regimes. The relation $\sigma_d \sim 2\sigma_{cx}$ holds within the classical regime above approximately 2 eV for $D^+ + D$. At lower energies, the particles must be treated as indistinguishable and separation of direct elastic scattering and charge exchange is no longer possible. At these low energies, spin exchange is no longer equivalent to charge exchange and the relation $\sigma_d \sim 2\sigma_{cx}$ no longer holds. This illustrates the difficulties in treating ion-neutral elastic collisions by charge exchange alone under low-temperature detached conditions.

The addition of these data present the user with yet another option for handling hydrogen charge exchange in DEGAS 2. Figure 2.8 compares the new CFADC data (labeled "Krstic") with previous versions and with experimental data. The cross sections have been plotted as a function of the relative velocity so that both the hydrogen and deuterium data can be included. The Krstic cross sections are the "spin exchange" cross sections described in Ref. [24]. They are thus *smaller* than the total elastic scattering cross sections that are to be used in DEGAS 2. This underscores the point that these new elastic scattering cross sections are intended to *replace* the existing charge exchange data, not used in addition to them.

The default data used heretofore in DEGAS 2 have been computed from the "Janev-Smith" fit[18]. This fit was derived from older data published by Barnett[30]. Barnett's data were a compilation of various experimental results. Included was the work by Newman[31]. Why the Barnett compilation diverges slightly from the Newman data at the lowest energies is not clear. The Reviere[32] curve in Fig. 2.8 represents the charge exchange cross section employed in the original DEGAS code. Not surprisingly, they are used in DEGAS 2 in the `Degas_box_bench` example. Note that Reviere's intended application was neutral beam penetration. Hence, the lack of agreement at divertor-relevant velocities is not surprising.

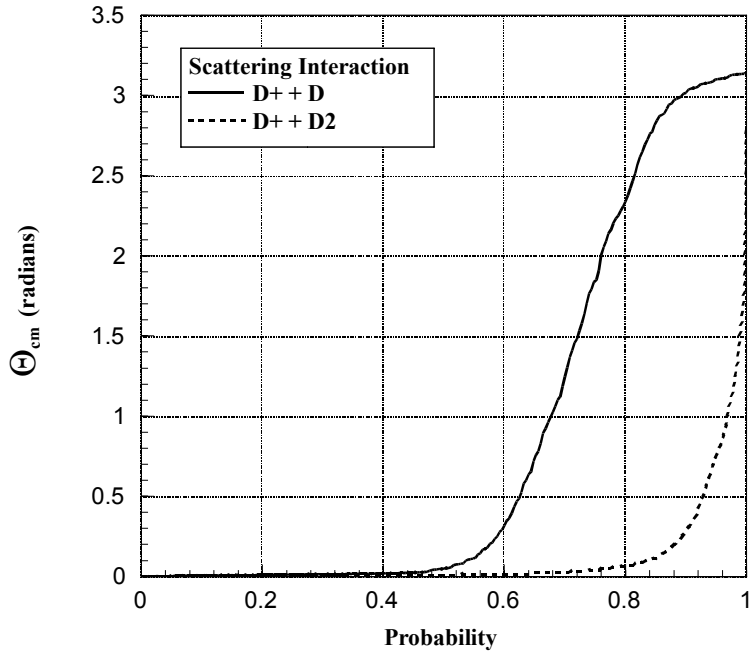

Figure 2.6: Scattering function $\Theta_i(P_i, E_r)$ for $D^+ + D$ and $D^+ + D_2$ ion-neutral elastic interactions at an interaction energy of 1 eV.

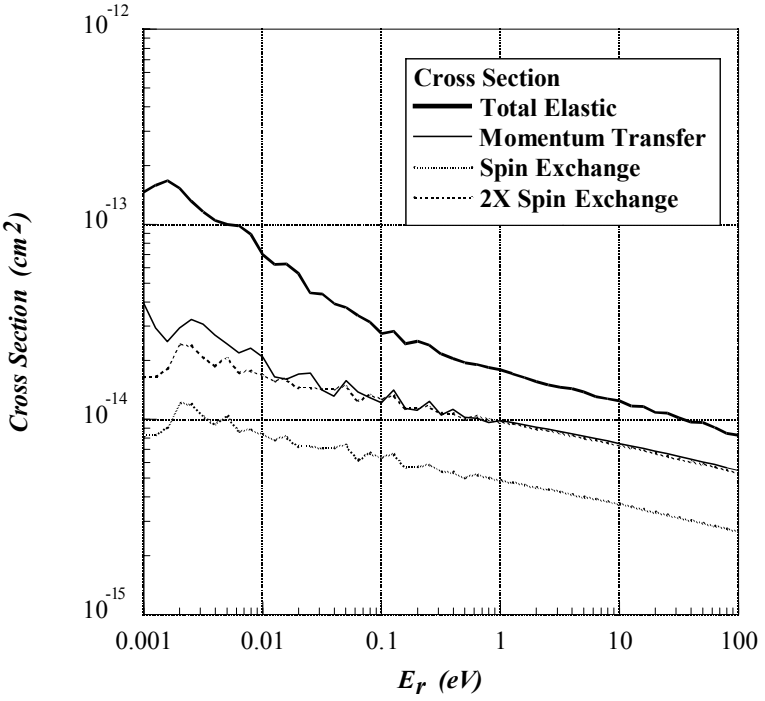

Figure 2.7: Comparison of diffusion and charge exchange cross sections for $D^+ + D$ elastic collisions.

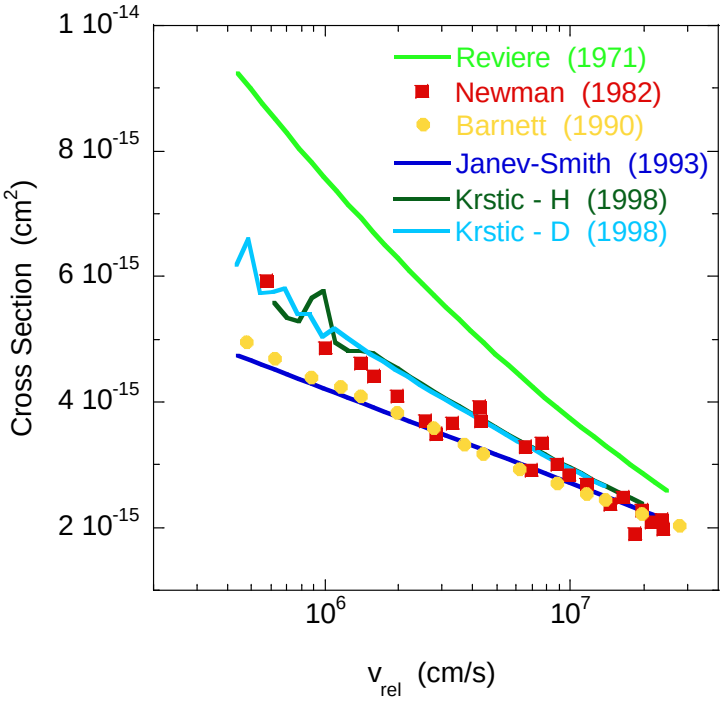

Figure 2.8: Comparison of various charge exchange cross sections. The curve labeled "Janev-Smith" represents the default data used in DEGAS 2 prior to the addition of the newer CFADC elastic scattering data, labeled here by "Krstic" (one curve for H, one for D).

2.11.2 Neutral-Neutral Elastic Scattering

Low temperature (a few eV or less) plasma conditions may result in neutral densities that are comparable to the plasma density. In such cases, collisions amongst the neutral species can be important in determining the exchange of momentum and energy in the system. For example, hydrogen molecules generated at a surface typically have energies comparable to room temperature, say, 0.03 eV. Yet, these molecules may be surrounded by much more energetic atoms, say 3 to 5 eV, resulting from molecular dissociation or charge exchange. Elastic collisions between the two species can significantly increase the molecules' energy, enabling them to penetrate much further into the plasma before becoming dissociated or ionized.

When the neutral-neutral mean free path is very short, $K_n = \lambda_{\mathrm{mfp}}/d \ll 1$ (d is a measure of the system size or of gradient scale lengths; K_n is the Knudsen number) cases, the most efficient approach would be to use a fluid model for the neutral species. This regime is also known as the "viscous flow" regime[33]. However, K_n is usually close to one in most fusion problems, even in vacuum regions. In that case, a kinetic treatment is required. Fortunately, such "transition regime" cases can be handled effectively with a Monte Carlo code. When $K_n \gg 1$, the "molecular flow" regime, neutral-neutral collisions can be neglected. For a practical discussion of these three regimes as they apply to the tokamak divertor, see Ref. [34].

The great difficulty with treating neutral-neutral collisions is that they make the problem nonlinear. Namely, the collision kernel is then a quadratic function of the neutral distribution function rather than a linear one. The exact problem is very difficult to solve, and one is forced to resort to approximations. Reiter et al. addressed this issue first with the EIRENE code and concluded that an iterative "BGK" approach (referring to the original algorithm developed by Bhatnagar, Gross, and Krook[35]; proposed independently by Welander[36]) was sufficient[37, 38]. The BGK approximation replaces the collision integral with a much simpler expression in which the distribution function relaxes to a Maxwellian distribution with a velocity independent collision time. A physically relevant result is ensured by determining the parameters of the Maxwellian distribution so as to enforce conservation of mass, momentum, and energy in the collisions. The collision time remains arbitrary and is chosen so as to reproduce measured viscosity and diffusion rates; this procedure is discussed in greater detail in Sec. 2.11.3.

Neutral-neutral collisions are included in DEGAS 2 in exactly the same way as in EIRENE. While more details on nature of the BGK algorithm are provided

in Kanzleiter's thesis[39], the essential elements and references are covered in the more readily available Ref. [38]. The BGK algorithm approximates the binary collision integral for species i as

$$\frac{\partial f_i}{\partial t} \approx \frac{M_i - f_i}{\tau_i} + \sum_{j \neq i} \frac{M_{ij} - f_i}{\tau_{ij}}, \tag{2.84}$$

where the M represent Maxwellian distributions,

$$M_\alpha \equiv n_\alpha \left(\frac{m_\alpha}{2\pi k T_\alpha} \right)^{3/2} \exp \left[- \frac{m_\alpha (\vec{v}_\alpha - \vec{U}_\alpha)^2}{2 k T_\alpha} \right]. \tag{2.85}$$

Here, the individual particle velocity is $\vec{v}_\alpha$; the mass m_α is always that of the species represented on the left-hand side of Eq. (2.84). In Eq. (2.84), a single subscript refers to a physical species. Dual subscripts below refer to fictitious, "mixed" species introduced for the purpose of handling binary collisions between unlike neutral species.

Conservation of mass, momentum, and energy in the self-collision case yields

$$n_i = \int_{-\infty}^{+\infty} d^3 v f_i(\vec{v}), \tag{2.86}$$

$$n_i m_i \vec{U}_i = \int_{-\infty}^{+\infty} d^3 v m_i \vec{v} f_i(\vec{v}), \tag{2.87}$$

$$\frac{3}{2} n_i k T_i = \int_{-\infty}^{+\infty} d^3 v \frac{1}{2} m_i v^2 f_i(\vec{v}). \tag{2.88}$$

Conservation of mass, momentum, and energy are also enforced in the the mixed-species case to determine the M_{ij}. Additional constraints are obtained by requiring that the ratio of the momentum and energy relaxation rates is the same as for the full collision integral. The results are[37, 38, 39] are

$$\vec{U}_{ij} = \vec{U}_{ji} = \frac{m_i \vec{U}_i + m_j \vec{U}_j}{m_i + m_j}, \tag{2.89}$$

$$k T_{ij} = k T_i - \frac{2 m_i m_j}{(m_i + m_j)^2} \left[(k T_i - k T_j) - \frac{m_i}{6} (\vec{U}_i - \vec{u}_j)^2 \right]. \tag{2.90}$$

One additional important constraint on the collision times is obtained,

$$n_j \tau_{ij} = n_i \tau_{ji}. \tag{2.91}$$

With $\tau_{ij} \equiv (n_j \langle \sigma v \rangle_{ij})^{-1}$ representing the typical time between physical species i and the fictitious species represented by M_{ij}, this becomes just

$$\langle \sigma v \rangle_{ij} = \langle \sigma v \rangle_{ji}. \tag{2.92}$$

May[37, 38] sets the values of the reaction rates for mixed collisions using empirical diffusion coefficients; the rates for self-collisions come from measured viscosity coefficients. The end result is that in both cases the reaction rate $\langle \sigma v \rangle_i \propto T_i^{1/4}$ or $\langle \sigma v \rangle_{ij} \propto T_{ij}^{1/4}$. The validity of these expressions is examined in the next subsection.

The practical implementation of the algorithm is iterative. By default, the initial iteration of the code proceeds without neutral-neutral collisions. The reason for this is that the neutral "background" with which the test species collide initially has zero density. A nonzero density could be specified during the setup of the background, if desired. The fluid moments of Eqs. (2.87), (2.88), and (2.88) are computed at the end of the iteration and inserted into the background density, velocity, and temperature arrays for the neutral background species. The calculation is then restarted; this time the test neutral species scatter against the neutral background. At the end of the run, the fluid moments are again updated. The iterations continue either a specified number of times or until some convergence criterion is satisfied.

Accurately simulating neutral-neutral scattering requires greater care in setting up the geometry and deciding on the number of flights than is exercised in linear runs. First, the entire problem space must be divided up finely enough to resolve all anticipated spatial variations of the neutral density and the other fluid moments. In comparison, without neutral-neutral scattering, the vacuum regions can be treated as large monolithic zones. The variation of the neutral parameters between iterations is asymptotically limited by Monte Carlo noise. In other words, if too few flights are used, the background densities are not accurately determined, causing the elastic scattering of the test species to be doubly inaccurate. The variations in neutral density, etc. in a given simulation will be more effectively reduced by increasing the number of flights than by carrying out more iterations.

2.11.3 Verification of BGK Model and Rates

As described by May[37], the reaction rates for the BGK algorithm can be determined by matching physical quantities, the viscosity and diffusion cofficients, against measured values. However, since May's thesis work and the publication

of Reiter[38], the ORNL CFADC has computed differential scattering cross sections [24], analogous to those described in Sec. 2.11.1, for these neutral-neutral systems. In this subsection we will show that the two approaches are largely consistent, at least near room temperature.

We proceed in a manner analogous to May. Namely, we will obtain from the CFADC differential cross sections values for the viscosity and diffusion coefficients. These will be transformed into reaction rates that can be compared with those described in Sec. 2.11.2.

For self-collisions, consider the viscosity. The general expression derived in Chapman and Cowling[40] [their Eq. (10.1,4)] is

$$\mu_1 = \frac{5}{8}\frac{kT}{\Omega_1^{(2)}(2)}, \tag{2.93}$$

where $\Omega_1^{(l)}(n)$ corresponds directly to the $\Omega^{l,n}$ integrals described by Bachmann[25]. The subscript "1" here is to be associated with the cross section and refers to collisions of species "1" with itself (as opposed to collisions between species "1" and "2" which will be labeled with "12" below).

Likewise, we have for the diffusion coefficient [Eq. (9.81,1) in Ref. [40]],

$$D_{12} = \frac{3}{16}\frac{kT}{(n_1+n_2)m_r\Omega_{12}^{(1)}(1)}, \tag{2.94}$$

where n_1 and n_2 are the species densities and m_r is the reduced mass for the system, $m_r = m_1 m_2/(m_1+m_2)$.

The $\Omega^{(l)}(n)$ integrals represent averaging of the collision cross sections over Maxwellian distributions for both species. The `ratecalc` code is able to integrate these cross sections over a single Maxwellian distribution, namely, the $I_{l,n}$ described in Reiter[17] and the documentation for `ratecalc`. Fortunately, the double integral we need here can be found as a limiting case of the single integral:

$$\langle I_{l,n}\rangle(u_p, u_n) = \lim_{u'\to 0} I_{l,n}(u_p', u'), \tag{2.95}$$

where $u_p'^2 \equiv u_p^2 + u_n^2$, and $u_p = \sqrt{2T_p/m_p}$ is the thermal velocity of the second species. Likewise, $u_n = \sqrt{2T_n/m_n}$ is the thermal velocity associated with the Maxwellian distribution used in the averaging over u. The $\langle I_{l,n}\rangle$ is then related to $\Omega^{(l)}(n)$,

$$\langle I_{l,n}\rangle = 8\left(\frac{u_p'}{u_p}\right)^n \Omega^{(l)}(n/2). \tag{2.96}$$

The same result is quoted in Ref. [25] and in [17], although in the latter case the variable equivalent to u'_p is incorrectly defined.

With these expressions, we can compute the single species viscosity and two species diffusion coefficients from the CFADC differential scattering cross sections. The corresponding expressions for the "Maxwell molecule" interaction [intermolecular force given by κ/r^5; e.g., see Eq. (12.1,1) in Ref. [40])] are

$$\mu_1 = \frac{kT}{3\pi A_2(5)}\sqrt{\frac{2m}{\kappa}}, \tag{2.97}$$

where $A_2(5) = 0.436$ is the result of a dimensionless integral. And from Eq. (14.2,2) of Ref. [40],

$$D_{12} = \frac{kT}{2\pi A_1(5)}\frac{1}{n\sqrt{\kappa_{12} m_r}}, \tag{2.98}$$

where $A_1(5) = 0.422$ is the result of another dimensionless integration.

The reaction rates needed for the BGK algorithm are related to the constant in the Maxwell molecule force law by (see Ref. [41] or [37]),

$$\langle \sigma v \rangle_{11} = 2\sqrt{2}\pi A_1(5)\sqrt{\frac{\kappa}{m}}, \tag{2.99}$$

and

$$\langle \sigma v \rangle_{12} = 2\pi A_1(5)\sqrt{\frac{\kappa_{12}}{m_r}}. \tag{2.100}$$

Now, we can use Eq. (2.97) to eliminate κ in Eq. (2.99) in favor of μ_1. Then, we can insert μ_1 as given by Eq. (2.93) to obtain an effective BGK reaction rate that has the same viscosity as that computed directly from the CFADC differential cross sections:

$$\langle \sigma v \rangle_{11} = 2.065\Omega_1^{(2)}(2). \tag{2.101}$$

The analogous procedure for the diffusion coefficient yields

$$\langle \sigma v \rangle_{12} = \frac{16}{3}\Omega_{12}^{(1)}(1). \tag{2.102}$$

Finally, we can use Eqs. (2.95) and (2.96) to replace the Ω integrals with the equivalent integrals obtainable from `ratecalc`,

$$\langle \sigma v \rangle_{11} = \frac{2.065}{32}\lim_{u'\to 0} I_{2,4}(u'_p = \sqrt{2}u_p, u'), \tag{2.103}$$

and

$$\langle \sigma v \rangle_{12} = \frac{2}{3} \frac{u_p^2}{u_p^2 + u_n^2} \lim_{u' \to 0} I_{1,2}(u'_p = \sqrt{u_p^2 + u_n^2}, u'). \tag{2.104}$$

In evaluating the latter expression, we will assume that one of the species (e.g., molecules at the wall temperature) are much cooler than the other, i.e., $u_n \ll u_p$. Equation (2.104) then becomes a function of only u_p.

The CFADC data exist only for the $D + D_2$ and $D + D$ systems (and isotopic equivalents; molecules are computationally intensive, hence, the absence of molecule-molecule data). These are compared with the May data in Fig. 2.9.

One concern with May's treatment of the $D_2 + D_2$ case is that the empirical viscosity formula quoted in Ref. [37] does not appear in the literature, nor is its origin clearly stated. The best guess for its origin is that it was obtained by rescaling the "Fuller" expression for the diffusion coefficient[42] into a viscosity using the theoretical expressions to relate the two. In Fig. 2.10, we compare May's expression with two that do appear directly in Ref. [42] ("Chung" and "Lewis") and with individual data points listed in the CRC Handbook[43]. The agreement of these values near room temperature (~ 0.03 eV) is expected. However, May's formula has a stronger temperature scaling than the others and exceeds them by about a factor of 15 at 10 eV. The relevant temperature range for this process is expected to be between 1 and 10 eV.

Figure 2.9 shows that at least near room temperature, May's expressions are also consistent with the CFADC data. However, in the 1 to 10 eV range, they are considerably larger than the CFADC values, suggesting that the values used in DEGAS 2 (and EIRENE) may be too large. However, we will demonstrate in Sec. 3.7.2 that the temperature dependence ascribed to these rates by May is inconsistent with Eq. (2.92). Consequently, DEGAS 2 now uses constant neutral-neutral reaction rates, i.e., independent of the "background" temperature. Since agreement of the May expressions with the CFADC and empirical viscosities is good at room temperature, and since room temperatures are physically relevant (at least for desorbed D_2) we fix the reaction rates at its room temperature (300 K = 0.02585 eV) value according to May's formulas.

2.12 Atomic Physics: Best Practices

The rates used for atomic physics processes in DEGAS 2 have evolved over the years. Yet, the older rates are retained in the interest of backward compatibility and verifiability of results. Details on the provenance of each set of rates are

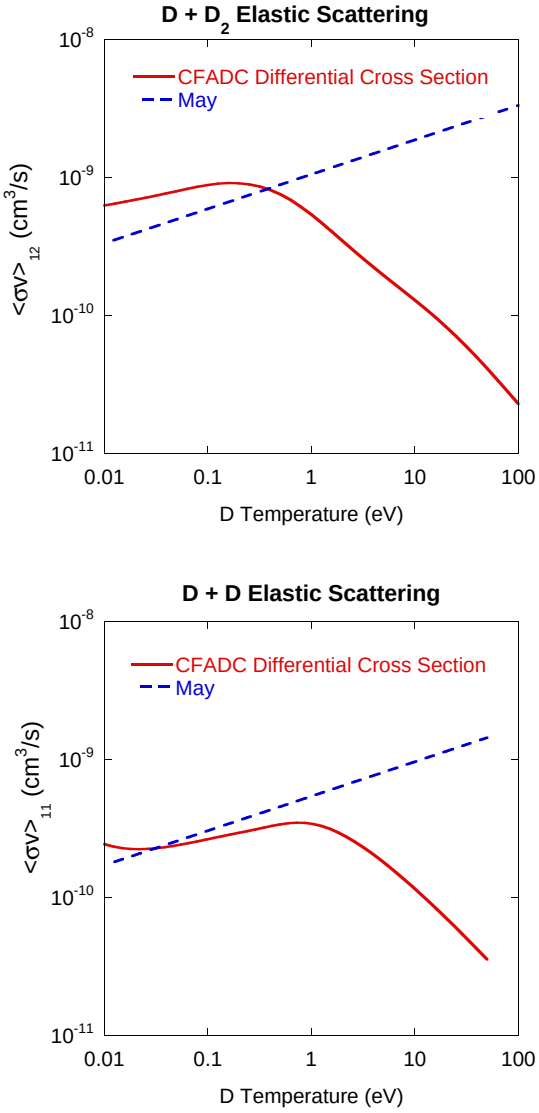

Figure 2.9: Comparison of reaction rates for the $D + D_2$ and $D + D$ systems. The original rates suggested by May[37] and effective rates computed from the CFADC data are shown.

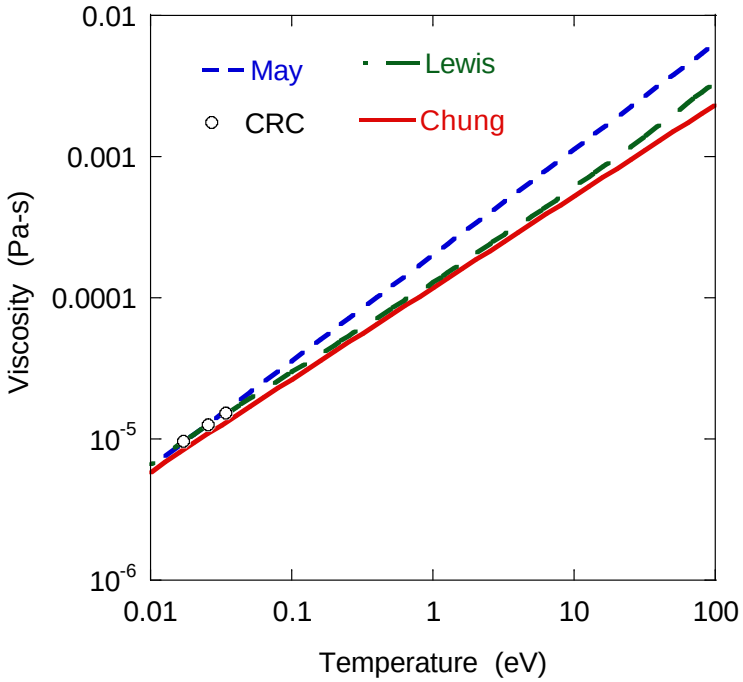

Figure 2.10: Comparison of viscosity formulas from May[37], Reid[42] (the "Chung" and "Lewis" expressions), and the CRC Handbook[43].

maintained in the DATA_HISTORY file in the degas2/data directory. The following subsections go into more detail on this and other points users should consider in setting up their problems.

2.12.1 Atomic Hydrogen Ionization and Recombination

hionize5 should be used for essentially all applications involving atomic hydrogen in a plasma; see Sec. 2.10.1 for a detailed description of earlier versions. The hrecombine5 data come from the same calculation as hionize5 and represent the best available data for recombination. Note, however, that recombination is significant only in high density ($n_e > 10^{20}$ m^{-3}), low temperature ($T_e < 5$ eV) plasmas. If in doubt, add the process to the problem input file and generate the background netCDF file, e.g., via defineback. The code will automatically add a recombination source to the end of the list of sources. Compare its source_total_current with the others in the problem. If the recombination source is much smaller, it need not be included.

2.12.2 Helium Ionization and Recombination

As noted in Sec. 2.10.2, the simpler "formulation II" model embodied in heionizeII and herecombineII should suffice for virtually all applications. The considerations noted above for hydrogen recombination also apply to helium.

2.12.3 Hydrogen Ion - Atom Charge Exchange and Elastic Scattering

A key point made in Sec. 2.11.1 that bears repeating is that *all* of the hydrogen ion-atom elastic cross section data provided with DEGAS 2 effectively include classical charge exchange as well. Thus, only one of the two processes should be included in a problem; using both amounts to doubling the charge exchange reaction rate. If the atom-ion interaction energies in a problem are well above 1 eV, a charge exchange process based on the ORNL data, e.g., dd_chargex can be chosen; if lower energies are expected, use *only* the corresponding elastic process, e.g., dd_05_elastic.nc.

A second important point is that the ORNL elastic / charge exchange data produced by ORNL[24] is considered definitive and unlikely to be superseded; the other charge exchange data files are retained only for backward compatibility

and verification exercises. All of the elastic scattering data, regardless of species, are made available with both 0 and 5 degree minimum scattering angles. Users generally should use the more efficient 5 degree versions; the 0 degree files can be used for a verification test in the event of questions.

The ORNL elastic scattering computations were performed for specific species; for interaction energies below a few eV, the cross sections do indeed differ (see Fig. 2.8). Above this range, where classical charge exchange is the dominant process, the cross sections are essentially identical when expressed as a function of the relative projectile velocity. Should one need to model a mix of isotopes, one can modify the reaction specification, e.g., for `hh_chargex`, in a local copy of the `reactions.input` file to change the reaction specification from "species specific" (with => appearing between reagents and products) to "generic" (using -> instead), just as was done for older charge exchange reactions, like `hchex`. As obvious as this equivalence may seem, the detailed considerations are more complex; the interested user should consult Ref. [45] for additional details.

An additional consideration is that great care has been taken with the files based on the ORNL data to ensure that the energy and temperature ranges used for the integrals are large enough that the user can be confident that the track length estimator will be accurate. With the older data files, e.g., `hchex`, the limits of the data can be exceeded, causing track length and collision estimator results to differ.

2.12.4 Hydrogen Ion - Molecule Elastic Scattering

This is a critical process in low temperature, high density divertor plasmas since it transfers momentum and energy from the plasma to the (cold) molecules, greatly increasing their mean free paths, and, hence, their ability to penetrate into the plasma. The comments made above regarding ion-atom elastic scattering largely apply, except that one typically ignores charge exchange (unless the molecules are vibrationally excited). The only available data are those from ORNL[24]; again, one should use the files with the 5 degree scattering cutoff, e.g., `dd2_05_elastic.nc`. The scaling of the cross sections with isotope is less straightforward than in the ion-atom case; users needing to model mixed isotope situations should contact the code authors (see Sec. 5.1).

2.12.5 Examples

Template problem input files for the indicated situations can be found in the following examples; they have been updated to reflect the above guidance.

NCSX, UEDGE_CMod Basic deuterium with Balmer-α light emission,

CMod_1D Deuterium with nonlinear elastic scattering; does not include all Balmer-α processes,

He_GPI Helium puffed into deuterium plasma,

He_metastables Single state helium model in `pr_boxgen_both.input` and three state helium model in `pr_boxgen_mboth.input`.

All of the these helium examples include light emission for the lines noted in Sec. 2.10.2 as part of the collisional radiative model.

Note that the other examples' problem input files have *not* been modified in this way so as to maintain consistency with previous results documented either in this manual or in the published literature.

2.13 Impurity Atomic Physics

The ability to incorporate impurity atom (and ion) generalized collisional radiative rate data from ADAS [46] was developed in preparation for the research described in Ref. [47]. However, apart from the `adaswrite` (Sec. 3.4.2) utility that reads the ADAS `adf11` files and reformats the data into DEGAS 2 netCDF files, none of that capability is included in this distribution. Interested users should contact the code authors (Sec. 5.1).

2.14 Recycling

Recycling of both ions and neutral species in DEGAS 2 is controlled by the set of PMI ("Plasma Material Interactions", e.g., see the PMI setup file) and one additional parameter generically referred to as the "recycling coefficient". The value of the recycling coefficient is a constant associated with each wall and target sector (see the sector class). These are prescribed during geometry definition, e.g., via the `recyc_coef` keyword in `definegeometry2d`.

The best way to understand the recycling coefficient, $\mathcal{R}$, is to note that $P_a = 1 - \mathcal{R}$ is the probability for a flight (the "projectile") striking that sector to be absorbed. Adsorption, is effectively one of the plasma-material interactions that the code will choose from in deciding the fate of the projectile.

The second PMI that is always present is referred to as the "default PMI" since the code defaults to that PMI should none of the other prescribed PMI (including adsorption) be selected. Since all PMI must have an associated probability or "yield", a "default PMI" is assigned a negative (i.e., unphysical) yield. The most common default PMI is desorption.

The third PMI we consider here is "reflection" or "backscattering" in which the incident particle returns immediately to the plasma with most of its incident energy. The data associated with the reflection PMI are used to compute a probability or yield, P_r, for the process. This probability is, in general, a function of the projectile energy and angle with respect to the surface normal.

Consider now the common case of an incident hydrogen atom for which both reflection and desorption (default) PMI's have been specified, as well as a recycling coefficient $\mathcal{R}$. The probability assigned to the desorption PMI is $P_d = 1 - P_r - P_a = \mathcal{R} - P_r$. In the case of unity recycling (the surface is in steady state), the physical interpretation is that atoms not reflected are adsorbed, eventually recombine with another atom, and are desorbed as molecules. In a steady state simulation, the time interval associated with "eventually" is meaningless, so the process might as well be instantaneous.

In the case of less than unity recycling, i.e. net adsorption, the value of P_d is reduced accordingly. If $\mathcal{R} < P_r$, P_d is negative and there is no desorption and the reflection probability is effectively $\mathcal{R}$.

If $\mathcal{R} = 0$, the sector is equivalent to an "exit". One operational difference is that flights striking an exit are terminated immediately. Flights striking a zero recycling sector will still undergo the process of selecting a PMI, even though the result will always be adsorption.

Chapter 3

Running DEGAS 2

3.1 Getting DEGAS 2

DEGAS 2 is now govern by a licensing procedure developed and managed by the PPPL Theory and Computation Department. Prospective users of the code should visit this site and follow the instructions spelled out there.

The DEGAS 2 source code is available in three formats:

1. `degas2.tar` An uncompressed tar file; extract with:

   ```
   tar xf degas2.tar
   ```

2. `degas2.tar.Z` The tar file compressed with the `compress` utility. To extract the code,

   ```
   uncompress degas2.tar.Z
   tar xf degas2.tar
   ```

3. `degas2.tar.gz` The tar file compressed with the `gzip` utility. To extract the code,

   ```
   gunzip degas2.tar.gz
   tar xf degas2.tar
   ```

There are a number of older versions of the code in this directory. The most recent is always linked to the names listed above. Please consult with the authors if you think you need an earlier version.

3.2 Getting Supporting Tools

DEGAS 2 relies on several pieces of utility software. In some cases (netCDF), these utilities are absolutely required. These utilities can be obtained from the indicated sites. You will probably want to have your system administrator download, compile, and install these packages system-wide for your convenience.

3.2.1 GNU Software

The DEGAS 2 Makefile automates several chores, including generating and compiling FORTRAN code, as well as weaving TEX files from from the FWEB source, The Makefile also controls the application of platform-specific lines of code and compilation options. Although not completely necessary, using DEGAS 2 without using GNU Make to run this Makefile would be extraordinarily inconvenient. The `gmake` (the GNU Make utility) can be found at https://www.gnu.org/software/make/

Likewise, the Emacs editor is not essential, but its use simplifies a number of tasks. For instance, Emacs provides a facility which allows the user to search for variable and routine names across many files. For example, say you're editing `randomtest.web` and you'd like to see the source code for `subroutine next_seed` called there. All you have to do is execute the Emacs command `M-. next_seed` (it will ask for the name of the tags file; the default value of `TAGS` is correct and you just need to hit `Enter` here). All of DEGAS 2 source and header files are included in this list of tags. Additional facilities such as a tag-query-replace (e.g., to change *all* occurences of a variable name) are available. See the Emacs Info file under the node "Tags" for more information.

3.2.2 netCDF

netCDF provides a device-independent binary file format. Furthermore, these files are self-describing. Most of the effort required to read and write these files is handled by macros. These are documented in the file `netcdf.hweb`. By adding the commands in `src/.emacs` to your .emacs file, Emacs will be able to convert automatically these files to text and back to binary for perusal and editing. netCDF is fully documented in the Info facility of Emacs. The netCDF libraries are required to run DEGAS 2 since virtually all of the input data is stored in netCDF files. The associated netCDF utilities `ncdump` (for translating a netCDF file into human-readable text) and `ncgen` (for generating a netCDF binary file from a text file) are useful for reading and making minor changes to DEGAS 2

input files. Two short scripts, `ncdump-filter` and `ncgen-filter`, allow Emacs to invoke these utilities via the commands in the .emacs file. These are not part of the netCDF distribution and are included amongst the DEGAS 2 source files for your convenience. These should be copied to a directory which is covered by your shell's `PATH` variable.

An additional note: if you use `ncgen`, be sure to either specify the `-b` or `-o netcdf_filename` option to let the utility know that you want it to generate a netCDF file (see: `man ncgen`).

netCDF can be obtained from http://www.unidata.ucar.edu/software/netcdf/. DEGAS 2 has been tested recently with versions 4.2.0, 4.4.1.1, and 4.4.4.2. Note, that the `ncgen` that comes with V. 4.1.1 and V. 4.2.0 has a bug that causes problems when creating large files; the work around is to use instead `ncgen3`. If you are using `ncgen-filter`, you would need use `ncgen3` there as well.

3.2.3 FWEB

The FWEB package provides a facility for code documentation as well as a powerful preprocessor. The typesetting abilities of FWEB allow authors to insert TEX (or LATEX) comments into their code. In DEGAS 2, the preprocessor is utilized extensively for writing macros which allow the code seen by the user to be significantly more powerful than pure FORTRAN. In fact, the code can be processed into FORTRAN 77 or FORTRAN 90 as needed. However, FWEB is *not optional*. The user must have access to at least its `ftangle` executable on some system in order to generate compilable FORTRAN files (once generated, these can be copied to other systems for compilation). For examples of the overall structure of FWEB programs, see the *.web files; most of the macros are located in the header *.hweb files. FWEB is fully documented in the Info facility of Emacs.

FWEB has been compiled and run on a wide variety of platforms, including, for example, the Edison machine at NERSC. Users should first try the most recent version: ftp://ftp.pppl.gov/pub/degas2/fweb_550bc1c.tar.gz Two slightly older versions are also available in the DEGAS 2 FTP area as alternatives should problems arise. The last version distributed by John Krommes (ca. 1998) can be found at: ftp://ftp.pppl.gov/pub/fweb/fweb-1.62.tar.gz.

The installation instructions distributed with FWEB are antiquated and needlessly complex for most systems. A simple approach should suffice:

1. Unpack the tarball in a convenient location.
2. In the `fweb/Web` directory, run: `./configure`

3. The `configure` script will choose a C compiler for you, likely `gcc` (recommended). You can override its choice by setting the `CC` environment variable,

   (a) If you have already run `configure`, delete the `config.cache` file before running it again.

   (b) Alternatively, one can set the `CC` variable in `defaults.mk` after running `configure`.

   (c) As of this writing, a bug in the "2017" version of the `icc` (Intel) compiler prevents it from working properly with FWEB's default optimization. The user can simply delete `-O2` from the `CFLAGS` variable in `defaults.mk`.

4. Run: `make` to compile both `ftangle` and `fweave`.

5. These executables can then simply be copied to a location in your `PATH`.

To typeset DEGAS 2 source code in "human-readable" formats such as DVI, PostScript, or PDF, the user will need FWEB's `fweave` utility and a version of TEX or LATEX. E.g., with `gmake tallysetup.pdf` Note that for most of the `.web` files, `fweave` will sound an error beep associated with the "memory allocation interface" macro; this is harmless.

LATEX is available in lots of places. To begin with, see the introductory material and pointers on the official CTAN TEX network page at http://www.tex.ac.uk/ctan. The recommended distribution for Unix (including Linux) systems is TEX Live (http://www.tug.org/texlive/. TEX Live is also available for Windows.

3.2.4 Triangle

Jonathan Shewchuck's `Triangle` program is used by the `definegeometry2d` code to break polygons up into triangles. The home for `Triangle` is http://www.cs.cmu.edu/ quake/triangle.html The `Triangle` V. 1.6 was used in the creation of the `definegeometry2d` examples described in Sec. 3.7.3. Anyone downloading a more recent version should notify the `DEGAS 2` authors. The `DEGAS 2 Makefile` will expect to find the `Triangle` files, `triangle.c`, `triangle.h`, etc., in `$HOME/Triangle`. If your copy is installed elsewhere, you can either create an appropriate link or modify `Makefile`.

3.2.5 Silo and HDF

DEGAS 2 produces no direct graphical output. Instead, the `geomtesta` utility (see Secs. 3.4.1 and the source code documentation.) generates device-independent binary files in either the HDF (more specifically, HDF Version 4) or Silo formats that can be processed by external graphical packages, such as VisIt or IDL. The VisIt package in particular is freely available
(see https://wci.llnl.gov/codes/visit/home.html), can read Silo files directly, and can generate both 2-D and 3-D images. A similar alternative to `geomtesta`, `ucd_plot`, is now available. It's more restrictive in that it requires the Silo library and works only with 2-D geometries created by `definegeometry2d`. On the other hand, the data contained in the Silo file are effectively represented on the actual DEGAS 2 mesh instead of a pixelated representation. Visualization of the `ucd_plot` Silo has only been performed with VisIt.

The Silo library can be found at https://wci.llnl.gov/codes/silo/index.html. It can use either its own PDB I/O driver or HDF5. In the latter case, the HDF5 library must also be installed. For HDF5 and / or HDF4, see http://www.hdfgroup.org/.

Note that `definegeometry2d` also produces an HDF or Silo file, `geomtestc.hdf` or `geomtestc.silo`, as part of its test of the geometry.

3.2.6 Git

The central DEGAS 2 source code is maintained in a Git repository. In principle, users planning to coordinate with the code authors can obtain access to the repository, although in practice none have done so. Instead, suggested modifications have been passed to the code authors via e-mail (See Sec. 5.1).

3.3 Structure of the DEGAS 2 File System

Here is an example structure of the directories under the top-level `degas2` directory. Those labeled with "(D)" represent source and data directories maintained with CVS. The others are created by the user as described in the compiling section. Note that the Makefile assumes that this `degas2` directory sits in the directory pointed to be the `$HOME$` environment variable on your workstation. If you have installed the code elsewhere, see Sec. 3.6.3.

data Input atomic and surface physics data and information (D)

src All source code (D)

dg_carre Source and executable files for DG and Carre (D)

Doc Location of the various formats of this document and ancilliary files (D)

examples Input and output files from example and benchmark runs (D)

Aladdin Source code and data for use with IAEA Aladdin package (D)

ALPHA Object and executable files for Dec Alpha machines

LINUX Object and executable files for Linux machines

LINUX64 Object and executable files for 64-bit Linux machines

SUN Object and executable files for Sun Solaris and SunOS machines

CRAY Pre-processed source code for export to Cray computers

tex Intermediate files and final DVI documentation files generated from source code

If you need to have multiple run directories for a given system (e.g., focussed on different problems or using different compilers), you can create additional directories with names like `SUN-foo` where "foo" is any string you like.

3.4 Components of the Code

DEGAS 2 is not a single code, but a complex package of setup, test, simulation, and post-processing tools. Which ones are needed depend on the user's objectives and familiarity with DEGAS 2. A few additional "targets" are listed in the Makefile. These are either sufficiently out of date or infrequently used to warrant mention here.

3.4.1 Main Code

Virtually every user will be running these executables, usually in this order.

problemsetup Reads a text file describing the species, materials, atomic, and surface physics required for the current problem. The code accesses the required "reference" data, and compiles them into a netCDF file (the `problem.nc` file).

definegeometry2d Generates a DEGAS 2 netCDF geometry file from from a text input file. Input files for simple geometries may be generated manually; more complex geometries can be specified via other computer generated text files files referenced by the input file. For example, `definegeometry2d` is able to read data produced by the `DG` package (extracted from the `SOLPS` distribution and included with DEGAS 2 for convenience) or by UEDGE. A newer option suitable for "main chamber" geometries of limited spatial extent is `efit2dg2d`, the output of which can be pasted directly into a `definegeometry2d` input file. Of the three tools for setting up DEGAS 2 geometries `definegeometry2d` is the most flexible.

readgeometry Generates a DEGAS 2 netCDF geometry file from an existing DEGAS, UEDGE, or SONNET geometry description. This package has largely been superseded by `definegeometry2d`. However, users with legacy input files for the old DEGAS code will still want to use `readgeometry`. A short text input file helps readgeometry through the transformation process.

defineback Uses a short text input file to control the mapping of plasma data in external text files onto the DEGAS 2 geometry. Designed to be used in conjunction with `definegeometry2d`, `defineback` can utilize data produced by other codes, including UEDGE. In particular, `defineback` can be run in an iterative mode with UEDGE analogous to that provided by `updatebackground` (see below and in the source file `defineback.web`).

readbackground Plasma data in an external file is mapped onto the zones of the DEGAS 2 geometry. Presently, only DEGAS and UEDGE files can be used; readgeometry and readbackground read the same data file.

updatebackground Is similar to `readbackground`, but is intended for use in iterations with the UEDGE code *only*. The source information generated by the initial run of `readbackground` is stored in a separate file (`oldsourcefile`, Sec. 3.5.1). On subsequent iterations, small changes

in the source terms are accounted for by allowing non-unit weighting factors at each source segment. Once the accumulated changes exceed a certain size (specified by parameters in the sources class), the DEGAS 2 source sampling arrays are re-initialized (i.e., to once again use unit weights) and the `oldsourcefile` is rewritten. The effectiveness of this procedure is currently being evaluated and should be considered as experimental.

tallysetup This code specifies the types of "scores" (the essential output of the code), along with their dependencies, which will be computed in the DEGAS 2 run. Although the input files(s) should not be modified by the novice user, this code needs to be run to factor the values of physics and geometry parameters for the problem at hand into the specification of these scores.

flighttest The core of DEGAS 2. This program launches, follows, and scores the Monte Carlo trajectories which make up a DEGAS 2 simulation. (The name is a holdover from early versions in which this did tracking with a minimum of hardwired physics).

outputbrowser Reads the output netCDF file generated by `flighttest` (in addition to all other netCDF files for the problem) and allows the user to interactively browse its contents. A script facility is provided for batch-like operation.

geomtesta Another inappropriately named routine which serves as the principal post-processing code for DEGAS 2. Although originally designed as a diagnostic for the geometry, it has proven too easy to continue extending this code to warrant development of an honest post-processing tool. The DEGAS 2 input and output files are read in by the code. Using the geometry information, the zone-based plasma and neutral data are transcribed onto a high density uniform rectangular mesh and dumped into Silo or HDF files suitable for further manipulation by external graphics packages such as VisIt or IDL.

ucd_plot Alternative post-processing code that uses DEGAS 2's own mesh for representing output data. Creates a Silo format file for visualization with VisIt.

degas2_xgc.a This library contains the core routines of DEGAS 2 and an interface for a subroutine based coupling to a plasma code. The primary quantities exchanged between the codes are moments of the plasma and neu-

tral distribution functions. An interface into the DEGAS 2 atomic physics routines is also provided. This library was designed for and is used with the XGC0 guiding center neoclassical particle transport code as part of the Center for Plasma Edge Simulation [44].

3.4.2 Other Setup Routines

These are occasionally used to add new atomic or surface physics reactions, or to generate new atomic and surface physics data files.

datasetup This routine possibly belongs in the first list. It reads text files describing the complete lists of elements, species, reactions, materials, and plasma-material interactions available to DEGAS 2 and generates the corresponding DEGAS 2 netCDF files. These lists are occasionally referred to in the code as the "reference" lists; the input file for problemsetup specifies subsets of these lists. A new reaction is added to DEGAS 2 by inserting the requisite information into the "reactionfile" and rerunning datasetup. The new reaction can then be added to the problem input file.

ratecalc A routine for computing averages of atomic physics cross sections. Since it is designed to read data from the Aladdin database, ratecalc is presently limited in its flexibility. The principal application thus far has been the generation of reaction rates and higher moments for charge exchange and other interactions between atoms, molecules, and ions.

reactionwrite A simple routine designed to translate collisional radiative data for the electron impact ionization of hydrogen from the format used by the old DEGAS code into netCDF files (ionization and recombination) for DEGAS 2. This code has been run a handful of times as needed to keep up with changes in the content and format of the atomic physics data files.

pmiwrite Is analogous to reactionwrite, but handles almost all of the plasma-material interaction data. Presently, two data files from the old DEGAS code and one from EIRENE are used as input. A few processes describable without external data are also included. This code can be used, in a somewhat clumsy manner, to add new plasma-material interaction data to DEGAS 2 using either one of these three existing data bases or via a new datafile (with the rest of the code serving as an example).

adaswrite Reads ADAS `adf11` [46] formatted files containing generalized collisional radiative rates, arranges the data to conform to DEGAS 2's `xsection` format, and writes them out as netCDF files.

boxgen Is the third of geometry / background generation tool provided with DEGAS 2. It is intended to serve as an example of how to generate geometry and background files from scratch. As the name implies, `boxgen` sets up a simple box geometry with a linearly varying plasma. However, no input file for `boxgen` has been developed; all modifications have to be effected through direct source code modification. The typical user in search of a simple linear geometry to study can be overwhelmed by the complexity of the code encountered there. For this reason, the `definegeometry2d` / `defineback` pair may be more suitable for setting up simple problems.

efit2dg2d Is effectively a pre-processor for `definegeometry2d` used for problems such as simulating gas puff imaging in the main chamber of a tokamak in which the spatial extent in the R-Z plane can be confined to a rectangular region with no intervening material surfaces or X-points. An EQDSK "g" file is read in and then used to trace out flux surface contours in the rectangle. These are transformed into "polygons", and a `definegeometry2d`-compatible representation of them is produced on output. Note, however, that additional effort is required by the user to assemble the `definegeometry2d` input file.

tri_to_sonnet Transforms an unstructured mesh specified in the `Triangle` format into a Sonnet-format mesh file suitable for use in `definegeometry2d`.

poly_to_tri Reads the "plasma" and "vacuum" triangles and rectangles from a polygon netCDF file (as produced by `definegeometry2d`), splits the rectangles into two triangles, and writes them all out as `Triangle` element and node files. Zone index and number data are transferred as well.

3.4.3 Test Routines

reactiontest Can be run after `problemsetup` to check the computation of the reaction rate, and collision products (including velocities and scoring data) for a given reaction.

pmitest The analog of `reactiontest` for plasma-material interactions. Again, this code can be executed for a given problem once `problemsetup` has

been run. Since the velocity distributions of some plasma-material interactions are described statistically, pmitest is also set up to average the outcome of a given interaction over a specified number of trials and print out a list of the product velocities for external manipulation.

sourcetest Tests the distribution in phase space of the source terms for a particular problem. Since the source is described in the "background" netCDF file, `defineback`, `readbackground`, `boxgen`, or something equivalent must be run prior to using this routine. Like `pmitest`, the code is able to run many trials, compute averages, and printout data from each individual trial.

dataexam Provides the user with a convenient interface to the atomic physics data files. Although netCDF files can be converted to a human-readable text form, interpreting those data is since DEGAS 2 collapses the multi-dimensional structure components of the atomic physics data into a single one-dimensional array. This routine reads the "raw" (at the "reference" level) data files prior to their insertion into the problem netCDF file and, thus, can be run at any time. An analogous routine for the plasma material interactions is needed.

sysdeptest Tests several system-dependent features of DEGAS 2. This would be run only if the code was being ported to a new operating system or if the operating system on a currently supported architecture underwent a signficant upgrade.

randomtest Is used to verify that DEGAS 2 will give the same results on different architectures (provided the random number seeds are set the same!).

3.4.4 Miscellaneous Routines

matchout Is used to compare the current DEGAS 2 output netCDF file with another from an equivalent run (i.e., all of the arrays must be exactly the same size) on the command line. The most frequent use of `matchout` is to verify that two separate runs have yielded results which are the same to within roundoff error. Because of the size of the ouptut netCDF file, verifying this by visually comparing the files is impractical.

matcheir Is specifically designed for comparing with output from the EIRENE code. The name of the EIRENE file is specified on the command line. See Sec. 4.3.

datamatch Provides a means for comparing the contents of the two netCDF data files specified on the command line.

3.4.5 DG and Carre

The graphical interface DG code is used here in conjunction with `definegeometry2d`, as in B2-Eirene / SOLPS (its original application), to simplify specification of 2-D divertor tokamak geometries . Since DG is also available separately from DEGAS 2, its manual, `dg.pdf` is contained in the `dg_carre/DG/DOC` directory. Carre[14] generates 2-D quasi-orthogonal meshes of the sort used by B2 and UEDGE. This version of Carre includes pre- and post-processing scripts that simplify its use with DG. A number of utility programs for manipulating magnetic equilibrium files also come with this DG distribution; see the DG manual for details. Note, however, that these versions of DG and Carre have not been kept up to date with those distributed with SOLPS-ITER. Users requiring additional capabilities not available in the DEGAS 2 versions may wish to contact the SOLPS-ITER development team.

3.5 Input Files

The essential inputs to a Monte Carlo neutral transport code are:

- The identity of the "test" species (usually neutral in plasma problems) that are of interest,
- The identity and phase space distributions of the "background" (typically electrically charged in plasma problems) species with which the test species will interact,
- The interaction processes between the test and background species. The relative probability of these processes as a function of the test and local background properties are required as well as a prescription for specifying the outcome of each interaction.

- The simulation geometry, including a framework for holding the background species data, as are a means for tracking the test species through the background.
- The boundaries of the geometry, including a prescription for specifying the outcome of interactions between the test species and these boundaries.

DEGAS 2 utilizes several text based input files for controlling the actions of its component executables (see Sec. 3.4). In this section we will discuss a few of these input files. The more complex input files are described in their respective codes; links to the documentation for those files will be given instead. More detailed information on running DEGAS 2 will follow in later sections.

These text input files are, for the most part, read with the same set of text processing utilities that adhere to the following guidelines. In some cases, particularly those involving files generated by other codes, the files are read directly with formatted FORTRAN read statements.

1. Spacing and blank lines are ignored. The user is free to utilize white space in whatever way facilitates reading and maintaining the input file.
2. Comments, either a complete line or at the end of a line, are begun with a # sign.
3. File names can be a relative or full path name.

3.5.1 degas2.in

`degas2.in` is the input file which controls (most of) the other input files. Each line consists of DEGAS 2's symbolic name for the file (this can be changed only by modifying `readfilenames.hweb` and `readfilenames.web`) followed by the path name for the file to be used in the current problem. The order of the lines in this file is unimportant.

```
elementsfile  ../data/elements.nc
backgroundfile bk_uers.nc
geometryfile ge_uers.nc
problemfile pr_uers.nc
reactionfile reactions.nc
speciesfile ../data/species.nc
```

```
aladinfile ../data/aladinput.nc
aladoutfile ../data/aladoutput.nc
elements_infile ../data/elements.input
problem_infile  pr_uers.input
reaction_infile reactions.input
species_infile ../data/species.input
materials_infile ../data/materials.input
materialsfile ../data/materials.nc
pmi_infile ../data/pmi.input
pmifile ../data/pmi.nc
cramdproblemfile cramdprob.nc
tallyfile tally_uers.nc
outputfile degas2_uers_out.nc
oldsourcefile os_uers.nc
snapshotfile sn_uers.nc
```

Note that some of these are outdated or rarely used (unused entries may be deleted from the file, if desired); see the readfilenames class for more specific information. Most of the files come in pairs with the name `XXX_infile` corresponding to a text input file and `XXXfile` being a netCDF file generated by a program like `datasetup`, `problemsetup`, etc.

3.5.2 elements_infile

`elements_infile` lists all of the elements available to DEGAS 2 for use in constructing the species (see Sec. 3.5.3). Additional detail is given at the beginning of the file `elementsetup.web`.

3.5.3 species_infile

Inputs for species are contained in the file with the symbolic name `species_infile`. Additional detail is given at the beginning of the file `speciesetup.web`.

3.5.4 reaction_infile

The file `reaction_infile` describes all of the reactions available in DEGAS 2. Additional detail is given at the beginning of the file `reactionsetup.web`.

Note that adding a new reaction to this file involves two tasks beyond inserting the appropriate lines in `reaction_infile`. First, the netCDF file for the atomic physics data must be generated (see Sec. 3.9). The second task would be to write subroutines for setting up the products and handling collisions (see Sec. 3.9.3). This would be necessary only if a new reaction type were being added.

3.5.5 ratecalc Input

The `ratecalc` code is used to compute atomic physics reaction rates from cross section data, as well as a few other tasks. The code is documented internally. Follow this link to the introductory section to learn more.

3.5.6 materials_infile

The materials described in `materials_infile` are essentially just labels which will be used in conjunction with the plasma material interactions (see Sec. 3.5.7) to specify how test species interact with non-transparent surfaces. Additional detail is given at the beginning of the file `materialsetup.web`.

3.5.7 pmi_infile

The file `pmi_infile` describes all of the plasma-material interactions (PMI) available in DEGAS 2. Additional detail is given at the beginning of the file `pmisetup.web`.

3.5.8 problem_infile

As noted already in the section describing the DEGAS 2 components (see Sec. 3.4), there are two levels of physics input to DEGAS 2. The input files noted thus far comprise the reference level: in practice, the sum total of the data available to the code (although in principle it could be smaller). The second level is the problem level of data. This prescribes the species, reactions, materials, and PMI which will serve as the physical model to be used in carrying the simulation at hand. In some parts of the (internal) code, these are also referred to as the subset data since they represent a subset of the reference data (see also Sec. 3.9 and Fig. 3.2).

Additional concepts alluded to above in connection with the reaction input file (see Sec. 3.5.4) are those of test species and background species. Most simply,

test species are the ones DEGAS 2 will track as they collide off of background species. The use of the word species here is important: both of the test and background lists are subsets of the "species" list (see Sec. 3.5.3). One more precise distinction between the two species types is that we assume that we know the distribution function (in space and velocity) of the background species; in fact, such information is required input to DEGAS 2. On the other hand, we are attempting to *compute* the test species distribution function, moments of which serve as the primary output of DEGAS 2.

More information about the input file can be found at the beginning of the file `problemsetup.web`.

3.5.9 readgeometry Input

The documentation for `readgeometry` input is maintained in the source code, `readgeometry.web`. For readers of the PDF version of this manual, here's a link to the corresponding PDF file.

3.5.10 definegeometry2d Input

The documentation for `definegeometry2d` input is maintained in the source code, `definegeometry2d.web`. For readers of the PDF version of this manual, here's a link to the corresponding PDF file.

3.5.11 defineback Input

The documentation for `defineback` input is maintained in the source code, `defineback.web`. For readers of the PDF version of this manual, here's a link to the corresponding PDF file.

3.5.12 readbackground Input

Currently, the only possible inputs to `readbackground` are the `UEDGE` (a specifically formatted text file generated during `UEDGE` post-processing) and `DEGAS` formats (an input file for the old DEGAS code). This routine makes some specific assumptions about the contents of these files. Since this situation is unsatisfactory from a number of viewpoints, a better long-term approach is being contemplated. For that reason, no additional documentation is provided at this point.

3.5.13 tally_infile

The documentation on the input file for `tallysetup` is maintained in the source code, `tallysetup.web`. This link will take the reader to the corresponding PDF file. The tally class contains more extensive and detailed information.

3.5.14 outputbrowser Input

This post-processing utility can be run interactively or with an input file. When run interactively, the session is logged into a script file called `outputscript` (in the run directory) which can be subsequently used as input to `outputbrowser`. Details on the use of `outputbrowser` and on the format of the script file can be found in the introduction to the file `outputbrowser.web`.

3.5.15 geomtesta Input

The documentation for `geomtesta` is maintained in the source code, `geomtesta.web`. For readers of the PDF version of the manual, here's a link to the corresponding PDF file.

3.5.16 ucd_plot Input

The documentation for `ucd_plot` is maintained in the source code, `ucd_plot.web`. For readers of the PDF version of the manual, here's a link to the corresponding PDF file.

3.6 Compiling DEGAS 2

The Makefile has been designed to compile on a variety of architectures and operating systems. It uses the shell command `uname` to determine the operating system upon which `gmake` is being run. The root name of the current directory (see Sec. 3.6.1) tells it the operating system for which the code is being compiled. In most cases, these are the same; a mechanism for cross compiling (i.e., target is a different operating system) is provided (Sec. 3.6.4). The systems used thus far are:

Generic Name	uname	Run Directory Root
Linux 32-bit	Linux	LINUX
Linux 64-bit	Linux, uname -m contains 64	LINUX64
Sun Solaris	SunOS	SUN
IBM AIX	AIX	IBM
Digital OSF1	OSF1	ALPHA
Silicon Graphics IRIX	IRIX or IRIX64	SGI
Mac OS X	Darwin	MACOSX

Note that due to the proliferation of relatively cheap and fast Linux clusters, usage of DEGAS 2 on other systems has essentially ceased and support for those systems cannot be guaranteed. The most frequently used compiler on Linux systems is Portland Group F90 compiler (PGROUP in the Makefile), albeit with F77 format code (FORTRAN90=no). The Pathscale (PATHSCALE in the Makefile) and gfortran (version 4.3) have been successfully tested with the F90 formatted code (in DEBUG mode only). The use of the F90 option eliminates the need for system dependent code so that other compilers can be used without modifications to the sysdep.hweb or sysdep.web files; only the compiler related variables in the Makefile or Makefile.local need be defined. On the other hand, some F9x compilers (e.g., Lahey-Fujitsu F95) may result in code that runs very slowly.

By default, the Makefile assumes that the DEGAS 2 main directory exists at $HOME/degas2. If you unpacked the tar file in some other location, you'll need to tell the Makefile where to look (see Sec. 3.6.3). Blindly change directory names is not recommended. Some flexibility is again provided; see below for details.

3.6.1 Basics

Let's start by compiling the randomtest utility (see Sec. 3.4.3) on a Sun system. To use other systems, replace SUN with the appropriate directory root name (more than one run directory can be used; see Sec. 3.6.3) from the above list.

First, change to the main DEGAS 2 directory and create the SUN subdirectory:

```
cd ~/degas2
mkdir SUN
cd SUN
cp ../src/Makefile .
```

Once the Makefile is present here, it will "update itself" with respect to the copy in the `src` directory when needed. To get an idea of how `ftangle` works, just "make" the main FORTRAN file for `randomtest`:

```
gmake randomtest.f
```

At some point, you may want to visually compare this file with the original source code `randomtest.web` (in the `src` directory, with the other source files) to gain an appreciation for the amount of work the FWEB macros do in DEGAS 2. For more information on these macros, see the documentation in `array.hweb` and the other header files.

Now, finish making `randomtest`:

```
gmake randomtest
```

To run `randomtest`, just type

```
./randomtest
```

The output to the right of the "=" sign should match the numbers in parentheses.

3.6.2 Making Documents

Generating "woven" (i.e., using FWEB's `fweave` utility) documentation is similar:

```
cd ~/degas2
mkdir tex
cd tex
cp ../src/Makefile .
gmake randomtest.dvi
```

The DVI file can then be printed or viewed (using `xdvi`), as desired. In fact, the Makefile also provides additional "targets", e.g., `randomtest.print` (uses `lpr` to print to your default printer), `randomtest.view` (launches `xdvi`) for you, and `randomtest.ps` (a PostScript file generated via `dvips`). You can generate woven documents for all of the "`.web`" source files. For some of the more useful ones, see Sec. 3.13.

You could also just generate the TEX file (again, from the `degas2\tex` directory),

```
gmake randomtest.tex
```

You could then use `latex`, `pdflatex`, or whatever other TEX application you wished.

3.6.3 Adjustments to Makefile

A number of the default settings in the Makefile can be overridden by placing the desired values in a file called `Makefile.local` in the working (e.g., `SUN`) directory. Some of the more frequently changed variables are:

- `DEBUG=no` turns on optimization. The default `DEBUG=yes` is required for debugging.
- `DEGASROOT=somedir` will tell Makefile to find the main DEGAS 2 directory at `somedir/degas2`. `somedir` should be an absolute path name. E.g.,

  ```
  DEGASROOT = /u/somedir
  ```

- `FORTRAN90=yes` switches from using the default FORTRAN 77 compiler to FORTRAN 90. Be aware that only some of the FORTRAN 90 compilers available work satisfactorily. Some will compile the code, but run slowly.
- `FCF77`, `FCF90` tell `Makefile` which "normal" (non-MPI) compiler to use.
- `MPI=yes` will enable the MPI commands in the source code and direct the `Makefile` to use the MPI compiler flags.
- `FCMPI77`, `FCMPI90` tell Makefile which MPI compliant compiler to use.
- `NETCDFV2=yes` tells DEGAS 2 that your system has only the older version 2 of the netCDF library (by default, DEGAS 2 expects version 3).
- `GRAPH_FILE` can be set to either `HDF4`, `SILO`, or `SILO_HDF5` to set the type of graphics output produced by `geomtesta` and `definegeometry2d`, according to the availability and capabilities of the Silo library. Note that this flag is also used in compiling `ucd_plot`, but that the code requires one of the two Silo options. The `postdetector` code does not use this flag since it produces only HDF4 files.
- The Makefile will accomodate more than one working directory on a given machine provided the additional directory has a name like `SUN-junk`, where "junk" is some string meaningful to you.

- If you would like to have more than one source directory, we recommend creating a new `degas2` directory heirarchy somewhere else in your file system and use the `DEGASROOT` variable to get the Makefile working there (you can set up symbolic links to common directories such as `data`).

3.6.4 Cross-Compiling

The mechanism for cross-compilation has been updated to use `ssh` and `scp` and no longer relies on the presence of AFS. However, this section of the manual has not been correspondingly updated. Please contact the code authors if you need this capability.

As an example of how to cross-compile, here are the procedures for making `randomtest` on the NERSC Crays. This assumes that you have access to AFS from your local workstation (and at NERSC, of course):

```
mkdir <some-afs-directory>/degas2/CRAY
ln -s <same-afs-directory>/degas2/CRAY ~/degas2/CRAY
cd ~/degas2/CRAY
cp ~/degas2/src/Makefile .
gmake randomtest
```

This will use `ftangle` on the local workstation to generate the FORTRAN source code files from the FWEB files in the `src` directory. It is advisable to set up a `Makefile.local` (see Sec. 3.6.3) in this directory that contains the same flags that will be used on the remote machine, particularly `DEBUG`, `FORTRAN90` and `MPI`.

Then, log on to the remote machine and do:

```
ln -s <same-afs-directory>/degas2 ~/degas2
cd <some-work-directory>/CRAY
cp ~/degas2/CRAY/Makefile .
make randomtest
```

In addition to the usual variables in `Makefile.local`, one needs to add `STANDALONE=no`. This will tell `make` that the FORTRAN (e.g., `.f`) files have already been generated on your local workstation, and that it can find them in `~/degas2/CRAY` on the remote machine, which should be linked back to your local workstation via AFS.

The degas2/data directory will likely also be needed on the remote machine. You can manually copy the directory and its contents to the remote machine (e.g., via ftp) or keep a copy in the AFS directory and use a link to allow the code on the remote machine to find the files. E.g., for the second option, starting on the local workstation:

```
cd ~/degas2
cp -R data <same-afs-directory>/degas2
```

And on the remote machine

```
cd <same-work-directory>
ln -s <same-afs-directory>/degas2/data data
```

Be aware that in either case, the data directory could eventually get out of sync with the one on your local workstation. If this is a likely possibility, one could consider also linking the data directory on the local workstation to the one in the AFS directory, making the latter the only real copy.

3.6.5 Miscellaneous Targets

The Makefile provides some other occasionally useful targets. Please see the Makefile for more details.

- clean removes source, object, and other unessential files. This is pretty thorough and indescriminate; use with caution.
- TAGS updates the Emacs tags table (see Sec. 3.2.1). You may need to do this if you subtstantially altered any of the source files or have other reason to believe that the tags table is out-of-date. You would run gmake TAGS from the src directory only.
- depend updates the Makefile dependencies (in Makefile.depends). Do this if the header file dependencies of one or more executables have been changed or if you believe that the dependencies are out of date. Note that source code (i.e., object files) dependencies are explicitly stated in the Makefile and must be updated by hand.
- foof dumps out values of some of the internal Makefile variables. This is useful for debugging Makefile problems.

3.7 Examples

The `examples` directory contains several documented DEGAS 2 examples, with input output, and auxiliary files. They are included not only for instructive purposes, but also for verifying that successive versions of the code give the same results. Namely, the user should be able to download the code and generate exactly (to within roundoff error) the results contained within each of the example subdirectories. Any discrepancies should be reported to the primary code authors (see Sec. 5.1).

3.7.1 Analytic_fluid_bench

This simple run demonstrates that DEGAS 2 matches the results of an analytic model in the fluid limit. More information on this comparison can be found in the IAEA proceedings[15] or on the Web as a PostScript or PDF file.

To run this example, follow these steps:

1. **Set source code switch.** The source code file `boxgen.web` has within it a few different settings available which can be selected by changing the value of the flag `BOXRUN`. With this file open in an editor, change the line

   ```
   @m BOXRUN 0
   ```

   so that it now reads

   ```
   @m BOXRUN 32
   ```

   Save the file and exit the editor.

2. **Switch to the example directory.** If the main DEGAS 2 directory sits within your home directory:

   ```
   cd ~/degas2/examples/Analytic_fluid_bench
   ```

3. **Copy the** `degas2.in` **file into your working directory.** E.g., if you are working on a Sun,

   ```
   cp degas2_boxgen.in ../../SUN/degas2.in
   cd ../../SUN
   ```

(overwriting any `degas2.in` file you may have had there already!).

4. **Prepare reference data.** This example uses the standard reference data files in the `data` directory. Hence, this step can be skipped if those files have not been altered since you downloaded the code.

   ```
   gmake datasetup
   ./datasetup
   ```

5. **Prepare problem data.** This and subsequent steps are *not* optional! Note that the `degas2.in` file points back to the `examples` directory for the problem input file (see Sec. 3.5.8).

   ```
   gmake problemsetup
   ./problemsetup
   ```

6. **Generate geometry and background files.** The `boxgen` program takes care of both of these:

   ```
   gmake boxgen
   ./boxgen
   ```

7. **Set the number of flights.** The background file `bk_boxgen.nc` contains the specification of the number of flights to be run. Open this file in an editor (Emacs will be the most convenient) and find the line

   ```
   source_num_flights = 100 ;
   ```

   Change the "100" to either "1600" or "6400". The example contains the output for both cases. Save the file and exit the editor.

8. **Set up the tallies.**

   ```
   gmake tallysetup
   ./tallysetup
   ```

9. **Run the code.**

```
gmake flighttest
./flighttest
```

10. **Check the text output.** The file `density.out` can be directly compared with either `density_1600.out` or `density_6400.out`, depending on the number of flights. The numbers in here have only five digits of precision. If an exact match is not obtained, something is amiss. Please contact the code authors (see Sec. 5.1) if you feel that a problem exists with the code as you downloaded it.

11. **Check the binary output.** You can use the `matchout` utility (see Sec. 3.4.4) to do this.

12. **Compare with the analytic solution.** The file `soln_32` contains the columns:

    x Distance along the problem space in meters.

    N1(x)/N(0) Relative density (to density at $x = 0$) predicted by one analytic model (see the above references) solution.

    N2(x)/N(0) Relative density predicted by a second analytic model solution.

    Ti(x) Ion temperature profile.

    Ni(x) Ion density profile.

    The columns of the `density.out` file contain

    (a) First zone number. These zones correspond directly to the x values in the `soln_32` file.

    (b) Second zone number. This is always 0 since this is a 1-D problem.

    (c) Neutral hydrogen density in m^{-3}. To normalize this column, divide the whole column by the value in the first row. Since this corresponds to the first value of x in `soln_32` and *not* to $x = 0$, multiply the whole column by the relative density from the analytic solution at the first value of x, normalizing the DEGAS 2 result to the analytic solution at one point.

    (d) Relative standard deviation of the neutral density (see Sec. 4.4).

    (e) Neutral hydrogen pressure in pascals.

    (f) Relative standard deviation of the neutral pressure.

    (g) The last two columns are no longer used and should be filled with zeroes.

3.7.2 Neutral-Neutral Scattering Examples

The initial implementation of the BGK algorithm for handling neutral-neutral collisions (see Sec. 2.11.2) in EIRENE has been tested by comparison with analytic expressions for the time dependence of the relaxation of a distribution function (both in self-collision and mixed-species collision systems) and against models describing the Couette flow problem.[37] The relaxation benchmarks have been repeated with DEGAS 2, but have not been formally made into an example run and will not be discussed here. Further details can be provided upon request. However, the Couette flow problem can be set up with only a few minor modifications to the code and does make a suitable example.

Couette Flow

The Couette flow problem of fluid mechanics involves the flow of fluid between two parallel, sliding plates. The fluid is assumed to have "no slip" boundary condition at the plates. The viscosity of the fluid drags along adjacent fluid elements, resulting in a velocity gradient between the boundary velocities represented by the two plates.

In this example, two semi-infinite plates are a distance d apart in the x direction. Periodic boundary conditions are enforced in the y and z directions so that the problem is effectively infinite in those directions. Particles are initialized at the $x = 0$ plate with a thermally distributed velocity to which a constant velocity in the y direction, v_{y0}, is added; this represents the velocity of the plate. The second plate at $x = d$ is treated as stationary. Particles striking it are assumed to be re-emitted with a thermal distribution pointed in the $-x$ direction. Particles striking the $x = 0$ plate are absorbed. Hence, in the no collision limit, particles make only one round trip across the box.

The choice of the distributions used at the plates is crucial to reproducing the analytic models. In particular, the "Maxwell flux" distribution [12] must be used. This distribution describes a recycling thermal flux of particles moving in a particular direction (in this case, perpendicularly away from the wall). This is the distribution used in some of the PMI data files, such as in `h2_des_maxw_mo`. In this example, however, the distributions of the particles coming off of the two plates are enforced through hardwired code, not through the data files.

The initial velocity is chosen from the distribution

$$Q(\vec{v}) \propto v_x \exp\left\{-\frac{m}{2kT}[v_x^2 + (v_y - v_{y0})^2 + v_z^2]\right\}. \tag{3.1}$$

Note the plate velocity v_{y0} in the exponent. For particles "reflected" at the $x = d$ surface, $v_{y0} \equiv 0$. This is just a thermal (Maxwellian) distribution multiplied by v_x, the velocity in the direction normal to the surface and is required for describing sources that simulate a recycling process.

A brief digression will clarify the relationship between this *source* distribution and the thermal distribution that characterizes the particles in the volume of the fluid. Consider the number of particles of arbitrary volume distribution f that recycle at, say, the $x = 0$ surface in a time interval Δt (a source distribution specifies a density in phase space per unit time). For particles in the velocity interval $v_x \rightarrow v_x + dv_x$, this is $f dv_x A \Delta x$, where A is the area of the recycling surface and $\Delta x = v_x \Delta t$ is the distance over which particles with velocity v_x can reach $x = 0$ in Δt. So, the phase space density of the recycling particles per unit time (dividing this number by Δt and dv_x) is $\propto v_x f$. Another way of saying this same thing is that the source needs to emphasize the faster particles in the distribution since more of them will reach the recycling surface in a given time interval. One upshot is that the average energy of the source particles is $\langle E \rangle_{\text{source}} = 2T$ while the average energy in the volume (for $v_{y0} = 0$) is the familiar $3/2T$.

For clarity, we note that the volume distribution function in the free-molecular limit is

$$f(\vec{v}) = f_+(\vec{v}) + f_-(\vec{v}), \tag{3.2}$$

$$f_+(\vec{v}) = \frac{n}{2} 2 \left(\frac{m}{2\pi T}\right)^{3/2} \exp\left\{-\frac{m}{2T}[v_x^2 + (v_y - v_{y0})^2 + v_z^2]\right\}, \; v_x > 0 \tag{3.3}$$

$$f_-(\vec{v}) = \frac{n}{2} 2 \left(\frac{m}{2\pi T}\right)^{3/2} \exp\left[-\frac{m}{2T}(v_x^2 + v_y^2 + v_z^2)\right], \; v_x < 0 \tag{3.4}$$

With this distribution, one can show that the fluid pressure is

$$P = \frac{2}{3}\langle nE \rangle = nT + \frac{1}{6} m v_{y0}^2. \tag{3.5}$$

The fluid velocity is

$$\begin{aligned} \langle \vec{v} \rangle &= \langle \vec{v} \rangle_+ + \langle \vec{v} \rangle_- \\ &= \frac{1}{2} v_{y0} \hat{y} + 0, \end{aligned} \tag{3.6}$$

so that $|\langle \vec{v} \rangle| = v_{y0}/2$. To compute the fluid temperature used in the BGK distributions, we need to subtract the drift energy,

$$T = \frac{P}{n} - \frac{1}{3} m \langle \vec{v} \rangle^2 \tag{3.7}$$

$$= \quad T + \frac{1}{12} v_{y0}^2. \tag{3.8}$$

The primary physical quantity appearing in the analytic models[48, 49, 50] of Couette flow is the x-y component of the stress tensor,

$$\Pi_{xy} \equiv \int d^3v \; m v_x v_y f(\vec{v}) - nm U_x U_y, \tag{3.9}$$

where $\vec{U}$ is the first (velocity) moment of f (the flow velocity). Note that while we should have $U_y \neq 0$, U_x should be identically 0 since there is no net flow in the x direction. In the molecular-flow regime, the distribution function is not changed by collisions as the particles move across the box. Furthermore, particles coming from the $x = d$ side of the box make no net contribution to Π_{xy} (since the integrand is an odd function of v_y). Inserting Eq. (3.3), we find

$$\Pi_{xy}^{\mathrm{fm}} = \frac{nm}{4} \left(\frac{8T}{\pi m} \right)^{1/2} v_{y0}. \tag{3.10}$$

As collisions become more important, the plate velocity information is dissipated into random motions of the fluid to a greater and greater extent, and Π_{xy} falls below Π_{xy}^{fm}.

The variational theory of Cercignani[49, 50] yields a relatively compact formula for Π_{xy} that matches the numerical results of Willis[48] to within 0.5%. With $\delta = 1/K_n$,

$$\frac{\Pi_{xy}}{\Pi_{xy}^{\mathrm{fm}}} = \frac{a + \sqrt{\pi}\delta}{a + b\delta + c\delta^2}, \tag{3.11}$$

where for the case of a BGK collision operator,

$$\begin{aligned} a &= \frac{4 - \pi}{\pi - 2} \\ b &= \frac{\pi^{3/2}}{2(\pi - 2)} \\ c &= 1. \end{aligned} \tag{3.12}$$

The x-y component of the stress tensor is computed along with other test particle data during tracking. However, final processing of that information into a usable form is only done when the `COUETTE` macro is enabled in `flighttest.web`.

The Couette flow problem can be simulated in DEGAS 2 with only a few code modifications to set up the required boundary conditions (at the moment,

the required particle distributions cannot be specified completely by input or data files). Because the problem is relatively simple, it represents a good starting point for learning how to use neutral-neutral collisions in DEGAS 2.

Most of these changes are invoked with a `COUETTE` FWEB macro inserted into the files

To run this example, follow these steps:

1. **Set source code switches.** The source code file `boxgen.web` has within it a few different settings available which can be selected by changing the value of the flag `BOXRUN`. With this file open in an editor, change the lines

   ```
   @m DIM 2

   @m BOXRUN 0
   @m SOLN 1
   ```

   so that they now read

   ```
   @m DIM 1

   @m BOXRUN 41
   @m SOLN 0
   ```

   Save the file and exit the editor. The `DIM` flag controls the dimensionality of the geometry. By setting it to 1, we get a purely one-dimensional geometry, with periodic boundary conditions in the y and z directions. The `SOLN` flag controls the analytic solution used with the "Analytic_fluid_bench" example (Sec. 3.7.1); since that's not needed here, we can turn it off. The files `plate.web`, `sources.web`, and `flighttest.web` each have the line

   ```
   @m COUETTE 0
   ```

   near the top. Edit these files, changing the "0" to a "1" in each case (be sure to revert to the default setting of "0" before running other problems).

2. **Switch to the example directory.** If the main DEGAS 2 directory sits within your home directory:

```
cd ~/degas2/examples/Couette_flow
```

3. **Copy the** `degas2.in` **file into your working directory.** E.g., if you are working on a Sun,

   ```
   cp degas2.in ../../SUN/degas2.in
   cd ../../SUN
   ```

   (overwriting any `degas2.in` file you may have had there already!).

4. **Prepare reference data.** This example uses some nonstandard data files so, unlike the previous example, `datasetup` must be run. Note that the `degas2.in` file points back to the `examples` directory for the input files needed here and in subsequent steps.

   ```
   gmake datasetup
   ./datasetup
   ```

5. **Prepare problem data.**

   ```
   gmake problemsetup
   ./problemsetup
   ```

   If you look at the problem input file, you will see that there is only one reaction, that for the neutral-neutral collisions. Note also that the test species `D2` also appears in the list of background species (the usual background species `e` and `D+` appear here just because `boxgen` expects them to be there; they are not used in this simulation).

6. **Generate geometry and background files.** The `boxgen` program takes care of both of these (be sure that the `BOXRUN` macro has been set to 41):

   ```
   gmake boxgen
   ./boxgen
   ```

7. **Set the number of flights.** The background file `bk_bgktest.nc` contains the specification of the number of flights to be run. Open this file in an editor (Emacs will be the most convenient) and find the line

```
source_num_flights = 100 ;
```

The results in this directory were obtained with 10 BGK iterations of 4,000,000 flights each. Such a run could easily require more than a day if run on a single processor (this run was executed in 15 minutes using 16 Dual-core AMD Opteron 1 GHz processors). You can probably run a smaller number of flights (say, 100,000) and still get reasonable results. Save the file and exit the editor.

8. **Set up the tallies.**

```
gmake tallysetup
./tallysetup
```

9. **Run the code.**

```
gmake flighttest
./flighttest
```

The number of BGK iterations is controlled by the value of the `bgk_max` macro in flighttest.web. The value used here is 10. If you just want to get an idea of how the calculation proceeds, you can choose a smaller number. Note that the macro parameters controlling the convergence tests, `bgk_cvg_dens` and `bgk_cvg_pres`, are set to ridiculously small values (10^{-8}) to ensure that all 10 iterations are completed.

10. **Check the text output.** At each iteration, the code writes out the test species density and other data in a file with a name of the form `densityxx.out` where `xx` is replaced by the iteration number. The contents of these files are analogous to those of the other examples (the density and pressure for each zone are in the third and fifth columns, respectively; see the description in Sec. 3.7.1), except for the last two columns. These are normally not used; here, they contain the values and standard deviations of the x-y stress tensor, Π_{xy}. The comparison of those values with theory will be discussed below.

    At the end of each iteration, the background netCDF file (`bk_bgktest.nc`) is also updated (overwritten) with the density, velocity, and temperature of the neutral background species. A third file, `cvg_global.txt`, will provide information on the global progress of the BGK iterations towards convergence. The four columns are:

(a) Iteration number,

(b) Test species number,

(c) Global fractional change in density (dimensionless); this is compared with `bgk_cvg_dens` to decide whether or not to stop the iterations.

(d) Global fractional change in pressure. (dimensionless); this is compared with `bgk_cvg_pres` to decide whether or not to stop the iterations.

Note that on most machines this file does not appear until the unit is closed at the end of the run.

These files can be directly compared with the corresponding files in the `examples` directory. If you have repeated the run at the above-described length, you should be able to match the text files exactly. The level of agreement in the netCDF file should be several digits (perhaps as many as 10). If a satisfactory match is not obtained, something is amiss. Please contact the code authors (see Sec. 5.1) if you feel that a problem exists with the code as you downloaded it.

11. **Check the binary output.** You can use the `matchout` utility (see Sec. 3.4.4) to do this. Again, you should expect the largest differences to be $\sim 10^{-8}$ or better.

12. **Compare with the analytic solution.** The rightmost set of data in the `densityxx.out` file are the stress tensor values, Π_{xy}. Ideally, these are constant across the problem space. Some deviation exists here due to Monte Carlo noise. You can compute the free molecular value from Eq. (3.10). With both of these numbers in hand, you can compare with Eq. (3.12). To do that, you need to know δ. The explicit formula used is

$$\delta = \frac{n\langle\sigma v\rangle d}{\sqrt{2T/m}}, \tag{3.13}$$

where $n = 10^{20}$ m^{-3} (enforced in `flighttest.web` by code enabled with the `COUETTE` macro), $\langle\sigma v\rangle = 1.1\times10^{-16}$ m^3/s (from the `d2d2_bgktest.nc` file), $d = 0.1$ m (set in `boxgen.web`), and $m = m_{\mathrm{D}_2} = 6.689 \times 10^{-27}$ kg (also from `boxgen.web`). One would expect the temperature T of the fluid to be equal to that of the wall by virtue of the initial conditions. However, in cases where the energy associated with the plate velocity v_{y0} is significant compared with T_{wall}, the actual temperature of the fluid is slightly

higher than that of the wall as is indicated by Eq. (3.8). In fact, this example with $v_{y0} = 1000$ m/s shows better agreement with the theoretical result Eq. (3.12) if T is set to the simulated fluid temperature (e.g., from the the `bk_bgktest.nc` file).

With $T = T_{\rm wall} = 0.0258{\rm eV} = 4.1336 \times 10^{-21}$ J, we find $\delta_{\rm wall} = 0.9894$. Plugging this into Eq. (3.12),

$$\left(\frac{\Pi_{xy}}{\Pi_{xy}^{\rm fm}}\right)_{\rm c}(\delta_{\rm wall}) = 0.6047. \tag{3.14}$$

The free-molecular stress, Eq. (3.10), is $\Pi_{xy}^{\rm fm}(T_{\rm wall}) = 0.20977$ Pa. If we average the 10 values for Π_{xy} in the `density10.out` file, we get $\Pi_{xy,{\rm sim}} = 0.13045$ Pa and

$$\left(\frac{\Pi_{xy}}{\Pi_{xy}^{\rm fm}}\right)_{\rm sim} = 0.6219. \tag{3.15}$$

This would qualify as pretty good agreement. However, if we average the fluid temperature values in the `bk_bgktest.nc` file, we get $T_{\rm sim} = 4.5866 \times 10^{-21}$ J. Then, $\delta_{\rm sim} = 0.9394$, and

$$\left(\frac{\Pi_{xy}}{\Pi_{xy}^{\rm fm}}\right)_{\rm c}(\delta_{\rm sim}) = 0.6157, \tag{3.16}$$

about a factor of two better agreement. Now, one might argue that we should also use $T_{\rm sim}$ in computing $\Pi_{xy}^{\rm fm}$. However, the free-molecular stress, Eq. (3.10) was computed directly from the general distribution function for arbitrary v_{y0} so that $T = T_{\rm wall}$ in that formula, without any ambiguity. Qualitatively speaking, for small δ the character of the source distribution dominates. As δ approaches or exceeds 1, the temperature of the fluid in the volume, $T_{\rm sim}$ computed from Eq. (3.8), better describes the distribution. The better agreement is shown over a range of inverse Knudsen numbers and two values of v_{y0} in Fig. 3.1.

C-Mod 1-D: Conservation Checks

This example demonstrates a run with complete hydrogen physics in a simple geometry. In particular, this run includes both neutral-ion and neutral-neutral elastic scattering processes. The complexity of the physics also makes for a good demonstration of DEGAS 2's conservation checks.

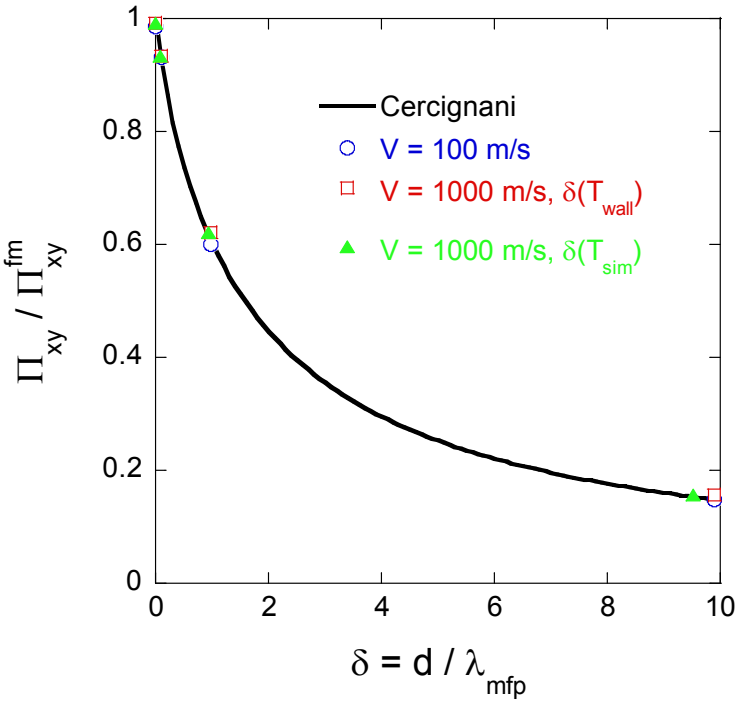

Figure 3.1: Comparison of 8 DEGAS 2 simulations (symbols) of the Couette flow problem with the analytic expression for the normalized shear stress Eq. (3.12) obtained by Cercignani[49, 50]. Two different plate velocities $V = v_{y0}$ were used in the simulations. For the higher velocity, better agreement with the analytic expression is obtained if the inverse Knudsen parameter is computed from the actual fluid temperature from the simulation, $T_{\rm sim}$, rather than the wall temperature, $T_{\rm wall}$.

Experiments by Pitcher et al. [59, 60] on the Alcator C-Mod tokamak indicated that the flow of neutral gas through material structures around the plasma was being limited by its diffusion rate through the divertor plasma. Pitcher [60] developed a semi-analytic model that reproduced this behavior. An attempt was made to precisely reproduce the results of that model with DEGAS 2. The overall effort consisted of several simulations in which the model assumptions were gradually relaxed towards those consistent with normal DEGAS 2 operation. This example is from a simulation in having atomic and surface physics assumptions more like a normal DEGAS 2 run. The remaining difference from a "full physics" DEGAS 2 run (apart from the greatly simplified geometry) is that the neutral source is a gas puff rather than a recycling source. The full physics and geometry simulations are described in Ref. [61].

To run this example, follow these steps:

1. **Set source code switch.** The source code file `boxgen.web` has within it a few different settings available which can be selected by changing the value of the flag `BOXRUN`. With this file open in an editor, change the lines

   ```
   @m BOXRUN 0
   @m SOLN 1
   ```

   so that they now read

   ```
   @m BOXRUN 40
   @m SOLN 0
   ```

   Save the file and exit the editor. The second parameter disables the analytic solutions that are used in the original boxgen example (Sec. 3.7.1). Note that the `DIM` parameter should remain at its default value of 2.

2. **Switch to the example directory.** If the main DEGAS 2 directory sits within your home directory:

   ```
   cd ~/degas2/examples/CMod-1D
   ```

3. **Copy the `degas2.in` file into your working directory.** E.g., if you are working on a Sun,

   ```
   cp degas2.in ../../SUN/degas2.in
   cd ../../SUN
   ```

(overwriting any `degas2.in` file you may have had there already!).

4. **Prepare reference data.** This example uses some nonstandard data files so, unlike the previous example, `datasetup` must be run. Note that the `degas2.in` file points back to the `examples` directory for the input files needed here and in subsequent steps.

```
gmake datasetup
./datasetup
```

5. **Prepare problem data.**

```
gmake problemsetup
./problemsetup
```

6. **Generate geometry and background files.** The `boxgen` program takes care of both of these. Be sure the `BOXRUN` macro was set to 40:

```
gmake boxgen
./boxgen
```

7. **Set the number of flights.** The background file `bk_oned.nc` contains the specification of the number of flights to be run. Open this file in an editor (Emacs will be the most convenient) and find the line

```
 source_num_flights = 40000 ;
```

   Save the file and exit the editor.

8. **Set up the tallies.**

```
gmake tallysetup
./tallysetup
```

9. **Run the code.**

```
gmake flighttest
./flighttest
```

The code will execute 5 iterations automatically, each with 40,000 flights. The number of BGK iterations is controlled by the parameter `bgk_max` in flighttest.web; the default setting is 5. The BGK iterations can alternatively be controlled by the convergence parameters `bgk_cvg_dens` and `bgk_cvg_pres`. The corresponding tests monitor global measures of the changes in the density and pressure of the BGK species. Note that these can be satisfied only if there are sufficiently many flights to achieve a correspondingly high precision in the local density and pressure values, no matter how many BGK iterations are run. In this example, the parameters chosen are such that `bgk_max` will be governing the number of iterations.

10. **Check the text output.** At each iteration, the code writes out the test species density and other data in a file with a name of the form `densityxx.out` where `xx` is replaced by the iteration number. The contents of these files are analogous to those of the other examples (the density and pressure for each zone are in the third and fifth columns, respectively; see the description in Sec. 3.7.1).

    At the end of each iteration, the background netCDF file (`bk_oned.nc`) is also updated with the density, velocity, and temperature of the neutral background species. This file can be used to restart the BGK iterations, e.g., with an increased number of flights. A third file, `cvg_global.txt`, providea information on the global progress of the BGK iterations towards convergence. The four columns are:

    (a) Iteration number,

    (b) Test species number,

    (c) Global fractional change in density (dimensionless); this is compared with `bgk_cvg_dens` to decide whether or not to stop the iterations.

    (d) Global fractional change in pressure. (dimensionless); this is compared with `bgk_cvg_pres` to decide whether or not to stop the iterations.

    Note that on most machines this file does not appear until the unit is closed at the end of the run. The output netCDF file from this run is included as well.

    These files can be directly compared with the corresponding files in the `examples` directory. The numbers in the text files have only five digits

of precision and should be matched exactly. The level of agreement in the netCDF file should be several digits (perhaps as many as 10). If a satisfactory match is not obtained, something is amiss. Please contact the code authors (see Sec. 5.1) if you feel that a problem exists with the code as you downloaded it.

11. **Check the binary output.** You can use the `matchout` utility (see Sec. 3.4.4) to do this. Again, you should expect the largest differences to be $\sim 10^{-8}$ or better.

12. **Examine test particle balances** To simplify examination of the conservation checks, an input script for the `outputbrowser` code is included, `bgkbalances`. The results are in `bgkbalances.out`. The checks can be performed for the density, momentum, and energy of each of the test species. Since `D2+` does not move, the checks on it are satisfied trivially and are not considered further.[1] For each quantity, the procedure consists of totaling the negative of the quantity lost to the walls (the "current into wall" tally), the quantity coming in from the walls (the "current out of wall" tally), and the "source rate" tally. The last item is further broken down by reaction (or source) to provide an indication of the relative importance of each process. For this example, we consider only the X-component of the momentum (dimension #1) since the problem does not vary in the other two directions.

    For this case, the following totals are obtained:

    particles

D	6×10^{16}
D_2	5×10^{15}

    momentum 1

[1] In fact, the values of the "total" tallies for D2+ are all determined by roundoff errors. For this reason, you will not likely match these values exactly. You should convince yourself that they are numerically much smaller than the other totals. If you use `matchout` to compare with the reference output netCDF file, these D2+ totals will also stand out as apparently significant discrepancies.

D	$< 1 \times 10^{-6}$
D_2	3×10^{-8}

energy

D	2×10^{-2}
D_2	-2×10^{-3}

In addition, a global check involving the background particle number can be performed:

\+ Total source of deuterium atoms (the puff source),
$+4.0 \times 10^{21}$

\− Total number of deuterium atoms lost to the walls,
$-(3.97174 \times 10^{22} + 2. \times 4.88694 \times 10^{21})$
$+(3.68843 \times 10^{22} + 2. \times 5.61804 \times 10^{21})$

\− Total number of deuterium atoms lost to the plasma (D+ source rate),
-2.62903×10^{21}

$= 7. \times 10^{16}$

There will usually be a small remainder in these balances; its magnitude in this case roughly coincides with the accuracy provided by `outputbrowser`. This remainder results from the enforcement of a minimum weight via Russian roulette in subroutine `follow`. By default, that minimum weight is 10^{-3} of the initial weight. Consequently, particles, momentum, and energy are conserved only on the average (i.e., after many flights). You can demonstrate this effect by reducing the minimum weight (`WMIN` in `flight.web`) to, say, 10^{-7}. Doing so will make the run substantially longer. To see the

difference in the conservation checks, you may need to examine the output netCDF file directly (hint: run `outputbrowser` first to get "rough" values for these quantities and then use the search facility in Emacs to locate the corresponding data in the output netCDF file).

13. **Transfers between BGK species.** In looking at the energy and momentum transfers between BGK species, you should notice that no sources appear for the background partners. The reason for this is that the transfers to and from these species do not have a clear physical meaning; all relevant information can be obtained from the corresponding test species.

    However, the transfers between BGK test species, say between `D` and `D2` here, are significant. The process `d2d_neut` (see the "momentum source vector by reaction" and "energy source by reaction" sections of `bgkbalances.out` under reaction 11) transfers energy and momentum between the background `D2` and the test `D` (problem species, "`problem_sp`", number 6); likewise, `dd2_neut` (reaction number 12) connects the background `D` and test `D2` (`problem_sp` 7). One would expect that in a converged state these two transfer rates would be equal (this actually goes into the derivation of the algorithm). In this example, they are (to within the error bars),

mom. 1 to `D`: -1.31488×10^{-2}	mom. 1 to `D2`: $+1.23601 \times 10^{-2}$
energy to `D`: -5.62387×10^{2}	energy to `D2`: $+5.49343 \times 10^{2}$

    (The molecules come off the wall at low energy and are being heated by the warmer atoms that arise from elastic scattering with the plasma ions and from dissociation.)

    For this to occur, the reaction rate for these two processes must be numerically identical. The original implementation of the BGK algorithm in EIRENE had the same temperature dependent expression for the two rates. However, because the temperatures of `D2` and `D` species are different in general, the resulting rates had different numerical values. The initial implementation of the BGK algorithm in DEGAS 2 repeated this same mistake. Due to a dearth of applications for these reactions, this problem has not yet been remedied.

    To obtain the nearly equal transfer rates seen above, the `degas2.in` file in this example actually points to a dedicated `reactions.input` file in the

same directory that uses modified ("hacked") data files for these two elastic scattering processes. To see the effect of unequal reaction rates, modify the `degas2.in` file to instead point to the `reactions.input` in the `data` directory. Rerun both `datasetup` and `problemsetup`, then run the code again. Note that you should also rerun `boxgen` to start the code off in the same initial state. The result after five iterations can be compared with `bgkbalances_default.out` that comes with the examples:

mom. 1 to `D`: -6.15071×10^{-3}	mom. 1 to `D2`: $+9.65537 \times 10^{-3}$
energy to D: -2.68765×10^{2}	energy to D2: $+4.44565 \times 10^{2}$

14. **Other results.** For reference purposes, this directory contains output files from a single iteration (on a Linux machine running Portland Group Fortran with `DEBUG=yes`) with 1000 flights. These output files contain the string `1K`. The primary purpose of these files is to serve as a known, reproducible point of contact for testing subsequent code modifications. It also demonstrates that these conservation checks hold even if the code is run with relatively few flights. To duplicate these files, you will need to stop the BGK iteration process after the first (zeroth, actually) iteration. The easiest way to do this is to change the number of iterations, near the top of `flighttest.web`, from:

```
@m bgk_max 5
```

to

```
@m bgk_max 0
```

You may notice that the tallies used for the conservation checks predominantly utilize collision estimators while the standard (zone-resolved) tallies are based on track length estimators. The reason is that the former will explicitly demonstrate conservation of mass, momentum, and energy since they are computed using the instantaneous test particle attributes. The track length estimators will generally provide more accurate values for the code output, but will exhibit conservation only in a statistical sense. You can demonstrate this by replacing the estimator lists in each of the "total" tallies with the lists from the corresponding zone-resolved tallies.

3.7.3 definegeometry2d and defineback Examples

The `definegeometry2d` (for setting up geometries, obviously) and `defineback` (for specifying the corresponding sources and plasma background) are sufficiently flexible and general that specific examples are useful in illustrating the ways that they can be used.

Gas Puff Imaging Example

This example illustrates the use of the DG and Carre codes with `definegeometry2d`, and also demonstrates the 3-D capabilities of `definegeometry2d`. It is based on the DEGAS 2 simulation of the NSTX Gas Puff Imaging (GPI) GPI shot 112811, documented in [62].

To run the 2-D version of this example, follow these steps:

1. **Switch to the example directory.** If the main DEGAS 2 directory sits within your home directory:

   ```
   cd ~/degas2/examples/He_GPI
   ```

2. **Copy the `degas2.in` file into your working directory.** E.g., if you are working on a 64-bit Linux machine,

   ```
   cp degas2.in ../../LINUX64
   cd ../../LINUX64
   ```

3. **Prepare reference data.** This example uses the standard reference data files in the data directory. Hence, this step can be skipped if those files have not been altered since you downloaded the code.

   ```
   gmake datasetup
   ./datasetup
   ```

4. **Prepare problem data.**

   ```
   gmake problemsetup
   ./problemsetup
   ```

5. **Generate geometry file.** Here, we outline briefly the preliminary steps leading up to the creation of the objects referred to in the `definegeometry2d` input file. Users interested in learning more about that process should consult the DG User's Manual (in the `dg_carre/DG/DOC` directory; especially the DEGAS 2 specific sections under "Customizing DG").

   The starting points for the DG code are a specification of the vacuum vessel shape [(R, Z) coordinates; `nstx_walls` in this case] and an equilibrium (values of the poloidal flux on a rectangular R, Z grid, `g112811.00250.equ`, already converted from the EFIT format to DG's preferred file format). Once these are loaded into DG, the user proceeds to graphically generate the input for the Carre code. Carre is run separately outside of DG to create a nearly orthogonal 2-D mesh with one coordinate that follows the flux surfaces. Although DEGAS 2 doesn't really need such a mesh, having one facilitates specifying plasma parameters (typically a function of a flux variable). That mesh is then loaded into DG and the user proceeds to connect it to the vacuum vessel points forming the "DG polygons" (see the entry in the definegeometry2d manual). This process results in three files: the DG file (`nstx_25.dg`), the mesh (`nstx.carre.021`) and DG's objects (`nstx_25.dgo`). Only the latter two are read in by `definegeometry2d`.

   There are two `definegeometry2d` input files here, one for the 2-D (axisymmetric: `nstx_25_dg2d.in`) case and one for 3-D (`nstx_25_dg3d.in`). The results contained in the `examples/He_GPI` directory are for the much quicker running 2-D case. Note that many of the 3-D lines appear in the 2-D file, but are commented out. Both runs utilize just the portion of the Carre mesh directly in front of the gas manifold ("puffer" in the `definegeometry2d` input file); this is done using the additional arguments to the `sonnet_mesh` keyword. One would establish the values of these indices during the creation of the DG polygons. The 3-D run then only simulates a portion of the torus (arguments to the `bounds` and `y_values` keywords). These steps keep the problem size and run time down.

   The emulation of the GPI camera view is done with the code in `gpicamera.web`. Prior to compiling `definegeometry2d`, the user should copy this file to `usr2ddetector.web` (see the brief explanation at the top of the default version of the file `def2ddetector.web`). This is actually just a placeholder 5×5 pixel version of the camera. The full resolution image is created during post-processing with the `postdetector` routine.

```
cd ../src
cp gpicamera.web usr2ddetector.web
cd ../LINUX64
gmake definegeometry2d
./definegeometry2d ../examples/He_GPI/nstx_25_dg2d.in
```

6. **Generate background file.** The detailed specification of the plasma parameters is actually done with the code in `nstxgpi2.web`. Like `gpicamera.web`, this is a user-defined subroutine (`get_n_t`) that needs to be compiled into the driver code. This routine will read the plasma density and temperature profiles obtained with the NSTX Thomson scattering system, `ts_112811_base.TXT` and map them on to the entire geometry assuming that the plasma density and temperature are constant on flux surfaces. On the Carre-generated mesh, this is straightforward. Outside of it, where there are only triangles, an approximate procedure is used.

   The `plasma_file` keyword in the `defineback` input file has two parameters used by the `nstxgpi2.web` `get_n_t` routine. The second is the name of the Thomson scattering data file mentioned above. The first argument provides the subroutine with the location of the separatrix in the Carre mesh subset used by `definegeometry2d`. Specifically, the separatrix is assumed to be between the cells `iz_sep` and `iz_sep+1`. If one knows the number of "cells" used in DG's "Create Surface(s)..." dialog for the core (`n_cells`) and the minimum `iz` used in the mesh subset (specified via the `sonnet_mesh` keyword in `definegeometry2d`), `iz_min`, the value of this is: `iz_sep = n_cells - iz_min + 2.`

   The second piece of input to `defineback` is the specification of the neutral sources. In the GPI simulations, this is a gas puff from the polygon representing the gas manifold, a vertical line in the 2-D case. The physical locations of the 2-D sources are provided by the "stratum" and "segment" indices in the file `source_nstx_26`. The "stratum" number corresponds to the "stratum" keyword used in `definegeometry2d`. In general, the simplest way to determine the segment numbers is to use the "$\star$" option (see the defineback documentation) to select all of the segments associated with a stratum and then manually identify those desired for the problem at hand. In this problem, a constant flux source is used since the sizes of the segments vary. The overall magnitude of the source is arbitrary in this problem, but physically realistic. The source file for the 3-D case

(source_nstx_26_3d) is similar, but also provides the toroidal location of the source segments (Segment_iy). These were chosen so that the source points fall along a nearly straight line at the same angle with respect to horizontal as the actual gas manifold.

```
cd ../src
cp nstxgpi2.web usr2dplasma.web
cd ../LINUX64
gmake defineback
./defineback ../examples/He_GPI/nstx_30_db.in
```

7. **Set the number of flights.** The background file bk_nstx_30.nc contains the specification of the number of flights to be run. Open this file in an editor (Emacs will be the most convenient) and find the line:

   ```
   source_num_flights = 100 ;
   ```

   The results in this directory were obtained with 100,000 flights. However, the 3-D production run used 2,000,000.

8. **Set up the tallies.**

   ```
   gmake tallysetup
   ./tallysetup
   ```

9. **Run the code.**

   ```
   gmake flighttest
   ./flighttest
   ```

10. **Check the text output.** The file density.out can be directly compared with the one in the examples/He_GPI directory. The numbers in here have only five digits of precision. If an exact match is not obtained, something is amiss. These results were obtained with Triangle V. 1.6.

11. **Check the binary output.** You can use the matchout utility to do this. The largest differences should be $\sim 10^{-10}$ or smaller.

12. **Generate graphical output.** For the 2-D simulation,

```
cp ../examples/He_GPI/geometry.inp .
gmake geomtesta
./geomtesta
```

The currently recommended approach is to compile the code with the `GRAPH_FILE=SILO` or `SILO_HDF5` option and use VisIt (built with Silo support) to view the Silo format file generated by `geomtesta`. The primary output variables in this problem are the emission rate of the helium 5877 Å (`he_5877`) viewed by the GPI camera and the neutral helium density, `spHeden`.

For 3-D simulations, two example input files for `geomtesta` are provided. Keep in mind that these need to be copied to the run directory and renamed `geometry.inp` in order for `geomtesta` to find them.

13. **Generate camera image and associated data.** The camera simulated by the main code uses only a 5×5 mesh in this problem so as to not lengthen the run. The generation of actual images suitable for comparison with experiment is deferred to postprocessing with the `postdetector` code. In principle, `postdetector` can be parallelized (although recent changes to the code have yet to be propagated into the MPI-specific sections), further mitigating the time expense for producing the simulated camera image.

    While `postdetector` has been generalized with regard to species (e.g., He or D_2), there are still application specific settings in the code. These are primarily the camera resolution and specification of the target plane; note that these same quantities appear in the preamble of `gpicamera.web`. The orientation of the simulated image can also vary with application; see subroutine `wrdatap` in `postdetector.web`.

    Open `postdetector.web` in an editor and find the section of code just above the start of subroutine —fill_views_master—, and set the application to "NSTX":

    ```
    @m APP NSTX
    ```

    As was the case in compiling `definegeometry2d`, the file `usr2ddetector.web` should be a copy of `gpicamera.web`. Note that the camera resolution parameters in `usr2ddetector.web` are overridden by those in `postdetector.web` and are not needed here.

```
gmake postdetector
./postdetector
```

The camera image and other data are written to HDF format files (this code has not yet been adapted for use with the Silo format). Here are the contents of the most useful files:

emtt Total emission rate of the 5877 Å line [in W/(m^2 ster)]. This is the simulated camera image.

tane Electron density (in m^{-3}) at the target plane (intersection of the camera view and the idealized gas puff sheet).

tate Electron temperature (in eV) at the target plane.

tarr Major radius (in m) at the target plane.

tarz Vertical coordinate (in m) at the target plane.

tns1 Neutral helium density (in m^{-3}) at the target plane.

Note, however, that the identification of the "target plane zones" is an automated optimization process that assumes a relatively fine toroidal discretization. As such, its results are rather difficult to interpret in the 2-D case.

The quantities in the other files require much more space to explain. The interested reader can learn more in [63]

14. **Three-dimensional simulation.** The process for producing the analogous three-dimensional simulation is the same as above. The only modifications are:

    (a) The input file for `definegeometry2d` is `nstx_25_dg3d.in`. Note that `definegeometry2d` will require much more time (on the order of an hour) to run.

    (b) The pointer to the source file in the `defineback` input file `nstx_30_db.in` must be changed to point to `source_nstx_26_3d`.

    (c) Many more flights are required for the main code, e.g., 2,000,000, and the run time is correspondingly longer.

    (d) One of the two 3-D input files should be used for `geomtesta`.

    The resulting camera image corresponds to Fig. 4(a) in [62]. However, the latter also incorporated the effects of camera vignetting, which are not accounted for by the `postdetector` routine.

NCSX Example

This example illustrates the use of `definegeometry2d` with input files generated with a relatively simple stand-alone code. The same concepts could be used to manually create `definegeometry2d` input files. This particular simulation provided neutral penetraton estimates used in the design of plasma facing components for the National Compact Stellarator Experiment [64]. Since the bulk of the recycled neutrals leave the surface with a cosine distribution (i.e., peaked about the normal to the material surface), the greatest penetration will be achieved in the same poloidal plane as the recycling surface. Consequently, toroidally axisymmetric (i.e., a single plasma cross section at a particular toroidal angle) neutral transport calculations can place an upper bound on the neutral penetration distance.

In scoping problems such as this, satisfactory results can be obtained by assuming that the plasma parameters are constant on flux surfaces (of course, if one does know the plasma variation along a flux surface, this assumption can be relaxed). The DEGAS 2 geometry can then be constructed out of flux surface shapes, e.g., from an equilibrium calculation of some sort. The basic approach is to define a "wall" corresponding to each flux surface and then connect adjacent walls to form the "polygons" required by `definegeometry2d` (see its documentation for detailed descriptions of these objects). More specfically, each pair of adjacent flux surfaces is used to construct two polygons (e.g., a lower and upper half) so that the interior of both is topologically well defined.

This objective can be in principle be achieved by manual manipulation of the coordinate data. However, the process is sufficiently straightforward, not to mention tedious, that a code can be constructed that generates the `wallfile` needed by `definegeometry2d`, as well as the polygons created out of those walls that form the bulk of the `definegeometry2d` input file.

The starting point in this case was a 3-D equilibrium file generated by the VMEC code, containing Fourier representations of the NCSX flux surfaces. An example code distributed with the VMEC file was first modified to evaluate these surfaces at a set of input toroidal angles (actually just two, with one represented here). A uniform poloidal angle mesh (an arbitrary choice) was then defined so that a given flux surface could be represented by a set of discrete points. There were many more flux surfaces in the VMEC equilibrium than were needed for this application. The set of surfaces was reduced (again arbitrarily) so that the maximum distance between two adjacent surfaces was 5 cm. A "wall" was created out of each of these surfaces, with the constituent points being the R, Z coordinates

of the surface at each of the points along the poloidal angle mesh. Because the relative location of these walls was known, the generation of the corresponding polygons was a simple matter. The code also defined walls and polygons filling the gap between the VMEC surfaces and the vacuum vessel and connecting the vacuum vessel to the universal cell. Details such as these depend on the particular application and are, thus, not worth describing here.

Another crucial task that must be begun at this stage is establishing the mechanism that will allow `defineback` to map the plasma parameters onto these polygons. The first, and generally essential, step to doing this is to label each polygon with a "stratum" number. Note that DEGAS 2's internal geometry specification, contained in the geometry netCDF file, has no record of the polygons used by `definegeometry2d`; it only knows about zones. The stratum labels are used in conjunction with the geometry's "sectors", but they are of little use within the plasma volume. For this reason, `definegeometry2d` also generates its `polygon_nc_file` based on the 2-D geometry class (see the geometry2d class documentation). Via the `g2_polygon_stratum` and `g2_polygon_zone` arrays in this class, the user can connect these zones back to the polygons specified on input to `definegeometry2d`. In this example, this objective is accomplished by the subroutine in the `ncsxplasma.web` file compiled into `defineback`. Be aware that the polygons in the `polygon_nc_file` file are derived from, but not identical to, those input to `definegeometry2d`. Specifically, polygons broken up with `definegeometry2d`'s `breakup_polygon` will remain as individual polygons in the `polygon_nc_file`. But, those processed with `trianglulate_polygon` or `triangulate_to_zones` are represented by the resulting triangles within the `polygon_nc_file` and, thus, correspond to multiple polygons there. The stratum labels are propagated to the triangle polygons in the process, however, permitting them and the zone numbers to be related to the original polygons.

In this example, the plasma profiles are specified as a function of the square root of the normalized toroidal flux. Consequently, the geometry setup code also writes out a file, `radiiz1li383`, containing the toroidal flux corresponding to each stratum. Because each pair of adjacent flux surfaces is used to construct two polygons, these two polygons are assigned the same stratum label in the `definegeometry2d` input file. The names of the files containing the plasma profiles, the effective stratum radii, and the `polygon_nc_file` are input to `defineback` and passed to the routine in `ncsxplasma.web`.

One will also need to be able to tell `defineback` where to place the neutral sources. This usually amounts to manually identifying the "strata" corresponding

to those sources within the `definegeometry2d` input file, and then following a simple procedure for deducing the specific "segments" of those strata that are to be used.

To run this example, follow these steps:

1. **Switch to the example directory.** If the main DEGAS 2 directory sits within your home directory:

   ```
   cd ~/degas2/examples/NCSX
   ```

2. **Copy the `degas2.in` file into your working directory.** E.g., if you are working on a 64-bit Linux machine,

   ```
   cp degas2.in ../../LINUX64
   cd ../../LINUX64
   ```

3. **Prepare reference data.** This example uses the standard reference data files.

   ```
   gmake datasetup
   ./datasetup
   ```

4. **Prepare problem data.**

   ```
   gmake problemsetup
   ./problemsetup
   ```

5. **Generate geometry file.**

   ```
   gmake definegeometry2d
   ./definegeometry2d ../examples/NCSX/dg2dinz1li383
   ```

6. **Generate background file.** As was described above, the detailed specification of the plasma parameters is done by the `get_n_t` routine in `ncsxplasma.web`. It requires as input the three files described previously (the effective radii for each stratum, the plasma profiles, and the `polygon_nc_file`), and the plasma density and temperature scrape-off lengths (the plasma parameters are only given on closed flux surfaces).

The remainder of the `defineback` input file contains the specification of the neutral sources. In this example, this is a recycling source at a limiter. The corresponding "stratum" was established as part of the process of creating the input files for `definegeometry2d`. A general, and relatively quick, technique for determining the source segment numbers is to use the "$\star$" option (see the defineback documentation) to select all of the segments associated with a stratum and then manually identify those desired for the problem at hand. The source strength is arbitrary in this problem, although a physically realistic value was selected.

```
cd ../src
cp ncsxplasma.web usr2dplasma.web
cd ../LINUX64
gmake defineback
./defineback ../examples/NCSX/z1li383_db.in
```

7. **Set the number of flights.** The background file `bk_ncsx.nc` contains the specification of the number of flights to be run. Open this file in an editor (Emacs will be the most convenient) and find the line:

   ```
   source_num_flights = 100 ;
   ```

   The results in this directory were obtained with 10,000 flights.

8. **Set up the tallies.**

   ```
   gmake tallysetup
   ./tallysetup
   ```

9. **Run the code.**

   ```
   gmake flighttest
   ./flighttest
   ```

10. **Check the text output.** The file `density.out` can be directly compared with the one in the `examples/NCSX` directory. The numbers in here have only five digits of precision. If an exact match is not obtained, something is amiss. These results were obtained with `Triangle` V. 1.6.

11. **Check the binary output.** You can use the matchout utility to do this. The largest differences should be $\sim 10^{-10}$ or smaller.

12. **Generate graphical output.** For the 2-D simulation,

```
cp ../examples/NCSX/geometry.inp .
gmake geomtesta
./geomtesta
```

3.8 Run Control Parameters

Several run control parameters are contained in the sources class. Ideally, these would be set by the user in a dialog box just prior to the run. For now, they must be changed manually by editing the background netCDF file. The following subsections describe the features, roughly in order of usefulness to the average user:

3.8.1 Time Dependence

The use of time dependence in DEGAS 2 is described above in Sec. 2.9. For completeness, we only note here the principal controlling parameters in the sources class. The switch `so_time_dependent` will be `FALSE` (0) for steady-state runs and `TRUE` (1) for the time dependent case. For the latter, the time interval for the run will be from `so_time_initial` to `so_time_final`; both have units of seconds. There is also a source-group dependent parameter `so_time_varn` that controls the time variation of tahat group.

3.8.2 Checkpoints

By default, the DEGAS 2 output netCDF file is written only at the end of the run. Each source group can be broken up into an integral number of pieces with an intermediate output file being written after each such checkpoint. The array `so_chkpt(grp)` contains the number of checkpoints for source group `grp`. If `so_chkpt(grp)` is 0, no file is written. If 1, an intermediate output file will be written at the end of the source group, and so on.

The intermediate output file does not contain post-processed results and cannot be used with post-processing utilities. The raw data there are suitable only for a restart of the code.

3.8.3 Restarting

The two main uses of the restarting capability are

1. To complete an interrupted (and checkpointed) run without having to restart from the beginning,

2. To increase the number of flights in a run to improve statistics.

In both cases, the user needs to change the value of the flag `so_restart` from its default value of `FALSE` (represented by 0 in the background netCDF file) to `TRUE` (1). The code will expect to find an existing output file, with the name specified in the `degas2.in` file. This can be either an intermediate output file from a checkpoint or a completed one generated at the end of a run.

The restarted run will extend the number of flights to the value specified in the background netCDF file (by `so_nflights`). A few different situations may arise:

1. If the number of flights is the same as those in the output file, no additional flights are run. This feature permits the data in an intermediate output file to be post-processed into a "completed" output file.

2. If the previous checkpointed run was interrupted, the restarted run resumes from the point of the last checkpoint and completes the specified number of flights. The results should match those that would have been obtained if the run were to have run to completion on the first try.

3. If the user wishes to increase the number of flights for the *last* source group of a completed run, the code will compute the required number of flights. The results will be the same as if the flights were all done at once.

4. If the user wishes to increase the number of flights in *more than one* source group, the code will extend the run accordingly. However, the results *will not* exactly match those obtained in a single run of the same length. The reason is that by default a run uses a continuous random number chain for all source groups. In a restart, that chain is resumed at its end. Hence, the first source groups will be using random numbers for these new flights different from those they would have had in a single run. Note that the resulting run should still be *statistically equivalent* to the single run. The "spaced seeds" option has been added to provide a workaround to this minor shortcoming of the restart procedure.

3.8.4 Seed String

Prior to version 2.6 of DEGAS 2, the initial random number seed was hardwired in `flighttest.web`. The user can now change the initial seed (and, hence, the entire random number chain) via `so_seed_string` in the background netCDF file.

The preferred mechanism for specifying the initial seed is as a character string. This is transformed into the appropriate integer representation using routines in `random.web` (see Sec. 2.6). The default value is "12". Reasons to use different values include

1. An unexpected, but perhaps not statistically significant, result has been obtained. A repeat of the run with a different random number seed will aid in establishing the validity of the result,

2. The DEGAS 2 code package contains several tools for statistically analyzing results (e.g., `matchout` and `matcheir`). By changing the random number seed, the effectiveness of these statistical comparisons can be evaluated.

3.8.5 Spaced Seeds

If the user anticipates needing to extend a multiple source group run to a larger number of flights (e.g., to achieve a desired variance) and insists that the results match those of a single run, the flag `so_spaced_seeds` should be set to `TRUE` (1) in the initial run. With this option enabled, the code will set the initial seed for each source group to be 100000000 (set by `so_seed_spacing`) flights apart. Each source group in a restarted run will then be able to "pick up where it left off", up to a total of 100,000,000 flights.

To minimize the chances of compromising the integrity of the random number chain, `so_spaced_seeds` should be left at its default `FALSE` (0) value, unless the user truly needs to be able to extend the run and match the single run results.

3.8.6 Direct Sampling

The default (random) method of sampling the initial particle source in DEGAS 2 leads to a source distribution that differs from the ideal, input distribution by a fraction of order $1/\sqrt{N}$ (N being the number of flights). Yet, if we were to take those same N flights and divide them up by hand amongst the source segments,

we would obtain an error of order $1/N$. Such an approach is impractical in general as it requires specifying the number of flights once and for all at the beginning of the run (a restart would not be possible) and the possibility of overlap in the initial source positions would arise if the number of flights were large enough.

An equally effective alternative can be developed based on hashing algorithms described by Knuth[58]. The idea is to replace the pseudo-random number ξ usually used in the source sampling process with

$$\xi = (x + i/\phi) \bmod 1, \tag{3.17}$$

where x is a single, fixed random number, i is the flight number, and

$$\phi^{-1} = \frac{\sqrt{5} - 1}{2} \tag{3.18}$$

is the golden ratio. This value yields the desired distribution properties (as can be verified using `sourcetest`), but minimizes the possibility of overlap by ensuring that ξ is far from low order rational numbers.

The sampling method used is governed by the flag `so_sampling`. The default value of `so_random` yields the usual random sampling procedure. This is the recommended value for most applications. The new, direct sampling method is selected by setting `so_sampling` to `so_direct`. This method was installed to see if it would lead to more efficient iterations in coupling DEGAS 2 to plasma transport codes.

3.9 Adding Reactions and PMI

3.9.1 Overview of Data Format

The same, basic approach is used to store all reaction- and PMI-related data in DEGAS 2. First, the data for each reaction or PMI is stored in its own netCDF file. That file contains:

1. Name of reaction,
2. Number of dependent variables,
3. Organization of data for each dependent variable ("table" or "fit"),
4. Rank of each dependent variable (number of independent variables),

5. Character name for each dependent and independent variable,

6. Number of values of each independent variable (for tabular data),

7. Data table, a single one-dimensional array. Indices into this array are computed on the fly based on the number of values used for each independent variable.

For more details, see the description of the cross section class. The corresponding information for PMI is in the description of the PMI format class.

The only general tool available for creating new data files is `ratecalc` (see Sec. 3.4.2 and the ratecalc file itself). Example input files for `ratecalc` can be found in the `data` directory. Two other routines `reactionwrite` and `pmiwrite` (again see Sec. 3.4.2) read existing text files and write the data out into netCDF files. These two routines can be used as examples for generating specific data files.

At run time, all of the reaction data files specified in the problem input file (see Sec. 3.5.8) (and only those) are read in. The reaction rates (for PMI, the corresponding quantity is termed the "yield") and the descriptions of their independent variables and other characteristics are compiled into one set of arrays. The information is organized in a similar manner to that of the original data files, but with an additional index corresponding to the problem reaction number. Additional processing is done to replace character strings with integer indices, to make for more efficient searches. Again, all of the actual reaction rate data values are stored as a single 1-D array with macros used to provide simple access to individual entries for a particular reaction. The reaction rate is used in computing the current mean free path of the flight and in choosing amongst the flight's possible reactions when a collision is indicated.

All other data present in the reaction data files is compiled into a set of "handling" arrays, indicating that their primary purpose is to control the specification of collision product velocities for that reaction. In addition to an index for problem reaction number, there is one for dependent variable number. The collision routines will search these data arrays for required information and will stop code execution (in `DEBUG` mode, anyway) if they are not found. The flow of data through the various DEGAS 2 classes is shown in Fig. 3.2.

The data format, evaluation routines, and scoring mechanisms are designed so that specific quantities can be scored just by adding the appropriate data to the data file and creating a corresponding tally with that quantity as the dependent variable.

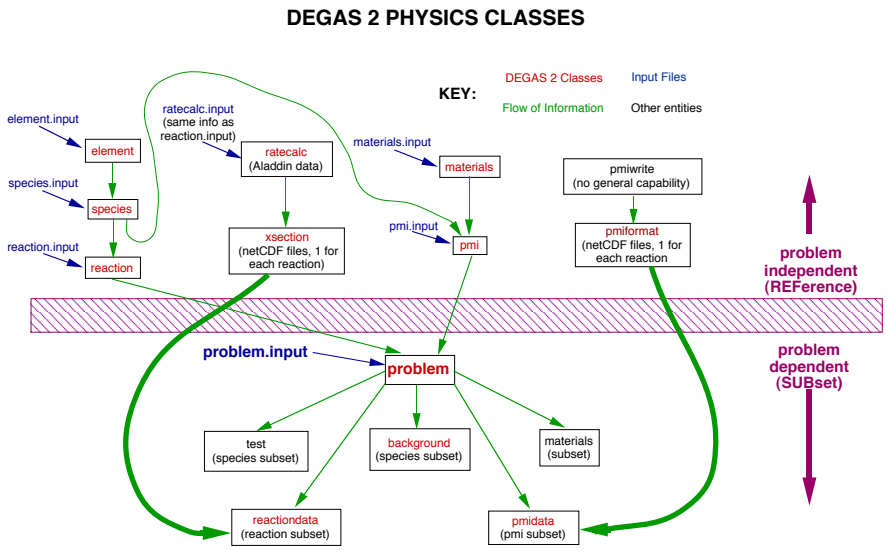

Figure 3.2: Flow of atomic and surface physics data through various DEGAS 2 classes during preprocessing.

One aspect unique to the PMI is that specification of the outgoing velocity distribution necessarily involves a "fit" since standard FORTRAN will not permit a function (in this case, the function which interpolates the data tables) to return a vector.

3.9.2 An Example: Bateman Format Data

A particularly involved example of a PMI is the specification of "Bateman format" data[3, 53]. A few uses of this for reflection processes are currently in the code. A large set of data for PMI in an improved version of this format will soon be added.

For each incident energy and polar angle pair, $(E_{\rm in}, \theta_{\rm in})$, we have a conditional distribution for the product atom

$$P_{E_{\rm in},\theta_{\rm in}}(v,\alpha,\phi)v^2 dv\ \sin\alpha d\alpha\ d\phi, \tag{3.19}$$

where α and ϕ are the outgoing polar and azimuthal angles, respectively.

This distribution is sampled from three separate 1-D distributions. First, the outgoing velocity v_0 is specified by

$$f^1_{E_{\rm in},\theta_{\rm in}}(v) = \int\int P_{E_{\rm in},\theta_{\rm in}}(v,\alpha,\phi)\sin\alpha d\alpha\ d\phi. \tag{3.20}$$

The outgoing polar angle α_0 obeys the distribution

$$f^2_{E_{\rm in},\theta_{\rm in}}(\alpha) = \int P_{E_{\rm in},\theta_{\rm in}}(v_0,\alpha,\phi)d\phi. \tag{3.21}$$

And, finally, the outgoing azimuthal angle ϕ_0 is taken from

$$f^3_{E_{\rm in},\theta_{\rm in}}(\phi) = P_{E_{\rm in},\theta_{\rm in}}(v_0,\alpha_0,\phi). \tag{3.22}$$

The inverse cumulative distribution, $F(\omega) = G^{-1}(\omega)$, with

$$G(x) = \omega = \int_0^x f(y)\ dy, \tag{3.23}$$

is specified, say, 5 values of $0 < \omega < 1$. For example, for a normally incident $\theta_{\rm in}) = 0$, ($E_{\rm in} = 1$ eV H atom being reflected off of Fe, the data are

$R_N(E_{\rm in}, \theta_{\rm in}) =$

```
8.27750E-01
```

$F^1(\xi) =$

```
1.99146E+00  2.25513E+00  2.32691E+00  2.36544E+00  2.40806E+00
```

Interpolation into this array with a random number ξ yields $\sqrt{E_{\rm out}}$, and thus v_0.

$F^2(\eta, \xi) =$

```
5.16532E-01  6.99576E-01  8.11503E-01  8.99906E-01  9.67300E-01
3.70481E-01  6.11799E-01  7.63020E-01  8.72539E-01  9.61285E-01
3.80010E-01  6.18834E-01  7.63825E-01  8.75375E-01  9.61256E-01
3.57688E-01  5.95520E-01  7.51752E-01  8.59766E-01  9.57489E-01
3.48497E-01  5.17881E-01  6.53833E-01  7.79846E-01  9.23956E-01
```

A 2-D interpolation into this table with the same first random number ξ and a second one η gives $\cos\theta_0$.

$F^3(\zeta, \eta, \xi) =$

```
-9.54674E-01 -5.93616E-01 -4.55439E-03  6.11607E-01  9.54292E-01
-9.39719E-01 -5.34726E-01  7.70975E-02  5.91124E-01  9.38723E-01
-9.47749E-01 -5.99118E-01 -5.34413E-03  5.58645E-01  9.59176E-01
-9.52932E-01 -5.09963E-01  1.11845E-01  6.68895E-01  9.59592E-01
-9.63812E-01 -6.32861E-01  2.84474E-02  6.49311E-01  9.65024E-01

-9.49486E-01 -5.27460E-01  4.71202E-02  5.40823E-01  9.43092E-01
-9.18789E-01 -5.34890E-01  3.87104E-03  5.97719E-01  9.43635E-01
-9.35546E-01 -5.66353E-01 -6.17420E-02  5.65376E-01  9.28863E-01
-9.39133E-01 -5.88361E-01  4.03247E-02  5.36441E-01  9.41553E-01
-9.42917E-01 -5.66390E-01  4.88304E-03  5.72368E-01  9.58254E-01

-9.59566E-01 -6.35221E-01 -7.50576E-02  5.72627E-01  9.59374E-01
-9.43053E-01 -6.23133E-01 -8.35145E-02  5.13894E-01  9.33565E-01
-9.42753E-01 -6.40605E-01 -1.50300E-02  5.51493E-01  9.32335E-01
-9.28625E-01 -5.03451E-01  3.69856E-02  6.04412E-01  9.51354E-01
-9.47567E-01 -5.75171E-01  6.78900E-02  6.17675E-01  9.61149E-01

-9.42125E-01 -6.25972E-01 -6.74072E-02  5.10579E-01  9.26111E-01
-9.24288E-01 -5.62514E-01 -6.67559E-02  5.49679E-01  9.57450E-01
-9.45958E-01 -5.88903E-01  6.07203E-02  5.98492E-01  9.45981E-01
-9.64714E-01 -6.17340E-01 -2.49212E-02  5.61052E-01  9.49261E-01
-9.59291E-01 -6.47473E-01 -8.40439E-02  4.58677E-01  9.33637E-01

-9.43142E-01 -5.25077E-01  4.79891E-02  6.22436E-01  9.53027E-01
-9.43214E-01 -5.90395E-01  2.32011E-02  5.42675E-01  9.35411E-01
-9.40994E-01 -4.99927E-01  7.81558E-02  6.19613E-01  9.44418E-01
-9.35018E-01 -5.38589E-01  8.60652E-02  6.47157E-01  9.51505E-01
-9.49873E-01 -5.38587E-01  8.93784E-02  6.39298E-01  9.66431E-01
```

A third random number ζ is used with the other two in a 3-D interpolation of this table to obtain $\cos\phi_0$. The re-use of these random numbers in interpolating these three separate dependent variables means that DEGAS 2 has to treat the random numbers as fixed independent variables, just like electron density or ion temperature, rather than generating the random numbers on the fly in the interpolation process. Hence, the appearance of `1st_random_number`, `2nd_random_number`, and `3rd_random_number` in the independent variables list (see the PMI format class).

So, each file for such PMI contains

1. $R_N(E_{\rm in}, \theta_{\rm in})$,
2. $F^1(\xi, E_{\rm in}, \theta_{\rm in})$,
3. $F^2(\eta, \xi, E_{\rm in}, \theta_{\rm in})$,
4. $F^3(\zeta, \eta, \xi, E_{\rm in}, \theta_{\rm in})$.

Again, all of the data values are stored in a single 1-D array with the array behavior specified by macros (see the PMI format class).

3.9.3 Reaction Processing Routines

Need to create "set products" and "products" routines, as well as collision and track-length routines if new type.

3.10 Defining Radiation Detectors

The "detector" class contains the objects needed to assemble synthetic diagnostics for filterscopes, imaging cameras, and other diagnostics that effectively integrate light emission along a chord through the problem volume. In the detector class, each such chord is referred to as a "view".

All of the views in the problem are characterized by:

1. Two points,
    - The first is taken as the starting point,
    - The second fixes direction.

2. Angular halfwidth (i.e., each view consists of a cone with apex at the starting point),

3. An averaging algorithm, used to simulate the finite spatial resolution obtained with real detectors. The two techniques, each having 2-D and 3-D variants, available are

    - Uniform weighting (i.e., square),
    - Circular weighting.

4. A binning variable,

    - Only examples are "none" and "wavelength,"
    - To use "wavelength", would also need to specify a minimum, maximum, and the number of wavelength bins.

The views thus defined can be separated into "groups". E.g., each group might represent a different diagnostic. The actual variables are defined and described in the detector class.

The contribution from the light emitted in each zone to a view is usually precomputed so that the signal can be quickly determined during or after the run from the volumetric emission. The array used to store these data potentially has size (number of zones) $\times$ (number of chords). However, the code only retains non-zero entries, resulting in an array of modest size since each chord only runs through a handful of zones. Should the user find processing of the chord signals too computationally demanding or that even the compressed array is inconveniently large, a subset of the chords can be used during the main run of the code and the full array generated only in postprocessing. The latter can be done in parallel, if need be.

The first of the three following subsections, Sec. 3.10.1, describes the computation of these signal contributions. The second subsection, Sec. 3.10.2, outlines the structure of the subroutine used to set up the detector views and groups of views. The third subsection, Sec. 3.10.3, describes one approach to computing the detector signal in post-processing.

3.10.1 Signal Computation

There are two situations:

1. Symmetric (2-D) geometry in which the detector chord is perpendicular to the ignorable coordinate. E.g., the chord would be in the x - z (planar symmetry) or R - z (cylindrical symmetry) plane.

2. Symmetric geometry in which the detector chord has a component along the ignorable component or a 3-D geometry. For simplicity, we will refer to this as the "3-D" situation.

In the 2-D case, an effective average over a 3-D cone is computed by exploiting the assumed symmetry in the ignorable coordinate. The 2-D viewing cone is divided into 30 subchords / degree. The code expects, but does not insist (a warning will be printed), that the apex of the cone does not fall inside a plasma or vacuum zone. Each subchord is tracked from the apex to the start of a plasma or vacuum zone. From there, the distance the subchord traverses through each zone is computed. The subchord is terminated when it reaches a solid zone; this is the actual end point. The resulting contribution made to the view by zone j, call it $f(z_j)$, is for $i = 2N + 1$ uniformly weighted subchords

$$f(z_j) = \sum_{i=-N}^{N} d_{i,j} w_i / V_j, \tag{3.24}$$

where $d_{i,j} =$ is the distance along subchord i through zone j, and V_j is the volume of zone j. The relative weight of each subchord is w_i with

$$w_i^{\text{uniform}} = \frac{1}{4\pi(2N+1)}, \tag{3.25}$$

and

$$w_i^{\text{circular}} = \frac{1}{2\pi^2 N\sqrt{1-(i/N)^2}}. \tag{3.26}$$

The units of f are $\mathrm{m}^{-2}/\mathrm{st}$.

In the 3-D case, each chord is modeled as a pyramid (for uniform weighting) or cone (for circular weighting) subtending a solid angle on the order of δ^2, where δ is the specified halfwidth expressed in radians. More precisely, the solid angles are

$$\Omega^{\text{uniform}} = 4 \arctan \frac{\sin^2 \delta}{\sqrt{1 + 2\sin^2 \delta}} \to 4\delta^2, \tag{3.27}$$

and

$$\Omega^{\text{circular}} = 2\pi(1 - \cos\delta) \to \pi\delta^2, \tag{3.28}$$

where the arrows indicate the value in the $\delta \ll 1$ limit.

The subchords are used to perform a 2-D integration over this solid angle. The direction of each subchord is defined relative to that of the detector chord; call the corresponding unit vector $\hat{\delta}_{\parallel}$. Two vectors perpendicular to this $\hat{\delta}_{\perp,1}$, $\hat{\delta}_{\perp,2}$ are determined in a somewhat arbitrary manner with $\hat{\delta}_{\perp,2} = \hat{\delta}_{\parallel} \times \hat{\delta}_{\perp,1}$.

The velocity vector sent to the tracking algorithm in the uniform case is

$$\vec{v}_{j,k}^{\,\text{uniform}} = \hat{\delta}_{\parallel} + \frac{(j-1/2)-n}{n} \sin\delta\hat{\delta}_{\perp,1} + \frac{(k-1/2)-n}{n} \sin\delta\hat{\delta}_{\perp,2}, \tag{3.29}$$

where the subchords are labeled by the indices j and k, both ranging from 1 to $2n$; currently $n = 6$. The weighting for each subchord is

$$w_{j,k}^{\text{uniform}} = \frac{\sin^2\delta}{4\pi\Omega^{\text{uniform}}n^2} \left|\vec{v}_{j,k}^{\,\text{uniform}}\right|^{-3/2}. \tag{3.30}$$

In the circular case, the azimuthal angle's 2π radians are discretized into $\phi_j = (j-1/2)\pi/n$, with $j = 1 \to 2n$. The polar angle is given by $\theta_k = (k-1/2)\delta/n$, with $k = 1 \to n$. Currently, $n = 6$. In terms of these angles, the velocity vector sent to the tracking algorithm is

$$\vec{v}_{j,k}^{\,\text{circular}} = \hat{\delta}_{\parallel} + \sin\theta_k(\cos\phi_j\hat{\delta}_{\perp,1} + \sin\phi_j\hat{\delta}_{\perp,2}), \tag{3.31}$$

and the weight for each subchord is

$$w_{j,k}^{\text{circular}} = \frac{\delta}{4\Omega^{\text{circular}}n^2} \sin\theta_k. \tag{3.32}$$

The resulting contribution to the chord by each zone is again given by Eq. (3.24) in both cases.

These zone contributions can then be used during execution of the main code or during post-processing to compute, say, the H_α signal S_α in the absence of recombination (see Sec. 2.10) as

$$S_\alpha = \sum_j N_1(z_j) \left[\frac{N_3}{N_1} A_{23} E_{23}\right] f(z_j), \tag{3.33}$$

where N_1 is the number of neutral H atoms in zone j. The bracketed quantity is interpolated from the input data file, with E_{23} being the energy of the Balmer-α transition, and other quantities defined in Sec. 2.10. Note, however, that the code is completely general regarding wavelength and origin of radiation. Other line emission rates can be scored in an analogous way if the appropriate data are present in the input data files.

3.10.2 The Detector Setup Subroutine

The subroutine used to set up the detector views and groups of views should be named `detector_setup`; all DEGAS 2 geometry definition routines (`boxgen`, `readgeometry`, and `definegeometry2d`) call this subroutine. The default, null implementation is contained in `def2ddetector.web`. A user developed `detector_setup` replacement for this routine should be placed in a file called `usr2ddetector.web` in the `src` directory. Example implementations of this routine provided with the code are `btopdetector.web` (a 1-D array) and `gpicamera.web` (a 2-D imaging camera; see also its documentation).

The suggested structure of `detector_setup` is:

1. For each view,
   - Set the two points defining the viewing chord,
   - Set the algorithm and half-width values.
   - Use these as arguments to the `detector_view_setup` subroutine (in `dediagsetup.web`).
   - This returns the contribution of each zone in the problem to that view's signal, $f(z_j)$ in Sec. 3.10.1. This quantity is copied into the `de_zone_frags` array.
2. Assemble these views into one or more groups, defining each with a call to `de_grp_init` (in `dediagsetup.web`).
   - If the detector group is to compute the spectrum of light emission, the requisite parameters are also passed to `de_grp_init`.

3.10.3 Detector Computation in Post-Processing

The `postdetector` (see also its documentation) code provides an example of detector signal computation during post-processing. `postdetector` provides additional data, such as the plasma parameters, along the same detector chords not available in the main output. The resulting detector based data are output directly to netCDF or HDF format files, simplifying subsequent processing. `postdetector` can also be used to compute the complete signal if a low resolution detector signal was used with `flighttest` or to

As is noted in Sec. 3.7.3, the version of `postdetector.web` distributed with DEGAS 2 was developed for simulating the camera in Gas Puff Imaging

experiments. Device and experiment specific information (namely a specification of the "target plane"; see gpicamera.web) appear here. Moreover, it explicitly calls the `gpi_views` subroutine invoked by `detector_setup` in `gpicamera.web`. Both there and in `postdetector`, the coordinates of a particular pixel are passed to `gpi_views`; it returns the end points and contributions of each zone to the signal, $f(z_j)$. In this way, the detailed information required to deterimine the view endpoints can be confined to a single subroutine, `gpi_views`.

3.11 Diagnostic Sectors

As was noted in Sec. 2.7, sectors can be defined for purely diagnostic use. The zones adjacent to the sector surface can be of any type. A given sector may be used to specify more than one diagnostic (e.g., one may track particle current, another might tally particle energy).

Diagnostic sectors are defined in a manner closely paralleling that of detectors. Diagnostic sector groups are used to identify sectors of similar functionality. The default sectors defined by DEGAS 2's geometry setup codes (such as `definegeometry2d`, `readgeometry` and `boxgen`) are used to construct the following default diagnostic groups:

Wall and Target Counts Includes all wall and target sectors. No independent variable is associated with this diagnostic group; it is only capable of summing a particular test particle property (which will be specified by a corresponding tally, particle mass, for example).

Exit Counts Includes all exit sectors. No independent variable is associated with this diagnostic group either.

Wall and Target Energy Spectrum Specifies the particle energy as the independent variable. As in the "Counts" groups, some as-yet-unspecified particle property will be tracked. The difference is that the tally will be divided up into the energy bins defined for this group.

Wall and Target Angle Spectrum Similar to the "Energy Spectrum" group, but the independent variable is the angle of incidence of the particle.

The particle properties to be associated with these diagnostic groups for the purpose of defining tallies are directionally dependent. For instance, the particle

current leaving and entering a zone may be specified separately. Details will be given in Sec. 2.3.3.

Additional diagnostic groups can be defined as needed; this is especially straightforward with `definegeometry2d`.

3.12 Output File

The output netCDF file (`outputfile` in `degas2.in`) contains the tally data in four arrays:

1. Sorted by the neutral source group, prior to post-processing,
2. Summed over neutral sources, prior to post-processing,
3. Sorted by neutral sources, after post-processing,
4. Summed over neutral sources, after post-processing.

All have standard deviation data, but their interpretation is not guaranteed for the latter two since the impact of post-processing (especially the addition of post-processed scores to a tally) is not accounted for. If the output file is the result of a checkpoint dump (Sec. 3.8.2), keep in mind that post-processing has not been done. Navigating these large arrays is extremely tedious and not recommended except as a last resort. Instead, the user should utilize `outputbrowser`. There is a separate output array for data to be passed to 2-D fluid plasma code.

A couple of text output files, largely of historical origin, are generated. The `density.out` file has been described already in Sec. 3.7.1. The `sources.out` and `testdata.out` files are unformatted files used to transfer data to the UEDGE code. All of the information in them is accessible through `outputbrowser`, so no further description of them will be given.

3.13 Other Documentation

The file `classes.web` contains an extensive description of what the code looks like internally. Most of the common variables (i.e., the contents of the `hweb` header files) are documented here as well. Try following this link if you want to have a look at it now.

Chapter 4

EIRENE Benchmark

4.1 Introduction

Since UEDGE was first coupled to EIRENE and since EIRENE is currently the most widely used neutral transport code in the field, the first step in coupling DEGAS 2 is to perform a benchmark against EIRENE. A simple slab, "single-null" geometry and plasma generated by UEDGE is used as input to both codes. An initial effort was performed without recombination. Essential information from that documentation has been included here for completeness.

This chapter describes a second round of benchmark exercises in which recombination is accounted for. The plasma $T_{e,i} \sim 1$ eV near plate in this case. For reasons to be explained below, several numerical differences between the codes are accentuated under these conditions. We will describe here the isolation and elimination of most of these differences, enabling us to get agreement of densities and plasma sources to within 5%. In the process we have developed a means for quantitatively comparing the code results.

The following sections will:

1. Describe the geometry, boundary conditions, and UEDGE plasma,
2. Qualitatively compare "out of the box" code results,
3. Describe how code results are quantitatively compared,
4. Explain several differences between the codes, eliminating or minimizing them when possible,
5. Characterize the level of agreement between the codes,

6. Go through a performance benchmark and optimization of DEGAS 2.

4.2 Problem Description

The geometry is intended to be a 2-D slab representation of a toroidally symmetric single-null scrape-off layer. The resulting "box" is 1 m long, extending from a mirror boundary at $Z = 0$ to a molybdenum target plate at $Z = 1$ m. Only half of the scrape-off layer is being simulated; hence, the appearance of the mirror. The radial width is 0.05 m. An exit is used to represent the core boundary at $R = -0.01$ m between $Z = 0$ and $Z = 0.75$ m. The outer wall at $R = 0.04$ m is assumed to be made of molybdenum. The section representing the wall adjacent to the private flux region, at $R = -0.01$ m between $Z = 0.75$ m and $Z = 1$ m, is also taken to be made of molybdenum. Finally, a mirror surface at $Z = 0.75$ m extending from $R = -0.01$ m to $R = 0$ creates a private flux region above it. The length of the box in the third dimension is 1 m, although periodic boundary conditions are established at both ends.

The UEDGE mesh has a nonuniform spacing with 64 zones in the Z direction and 32 in R. The geometry and plasma data are provided to DEGAS 2 in a file generated during UEDGE post-processing by the BASIS utility. The electron temperature and density computed by UEDGE with its fluid neutral transport model are shown in Figs. 4.1 and 4.2.

4.3 "Out of the Box" Results

Figures 4.3 and 4.4 show how the atomic deuterium densities computed by EIRENE and DEGAS 2 using the UEDGE plasma compare with roughly standard physics. The word "roughly" is meant to imply that some of the changes to be described later in this chapter have been incorporated into these simulations. For EIRENE, the random number problem (see Sec. 4.5.2) has been remedied. For DEGAS 2, the treatment of reflection used on the molybdenum surfaces uses the EIRENE data, including the attempts to mimic EIRENE's extrapolation behavior (see Secs. 4.5.3 and 4.5.4).

The peak EIRENE density is about 3×10^{20} m^{-3}. For DEGAS 2, the peak is only about 2×10^{20} m^{-3}. In Sec. 4.5, we will list the causes of this discrepancy and describe the techniques we use to eliminate or minimize them.

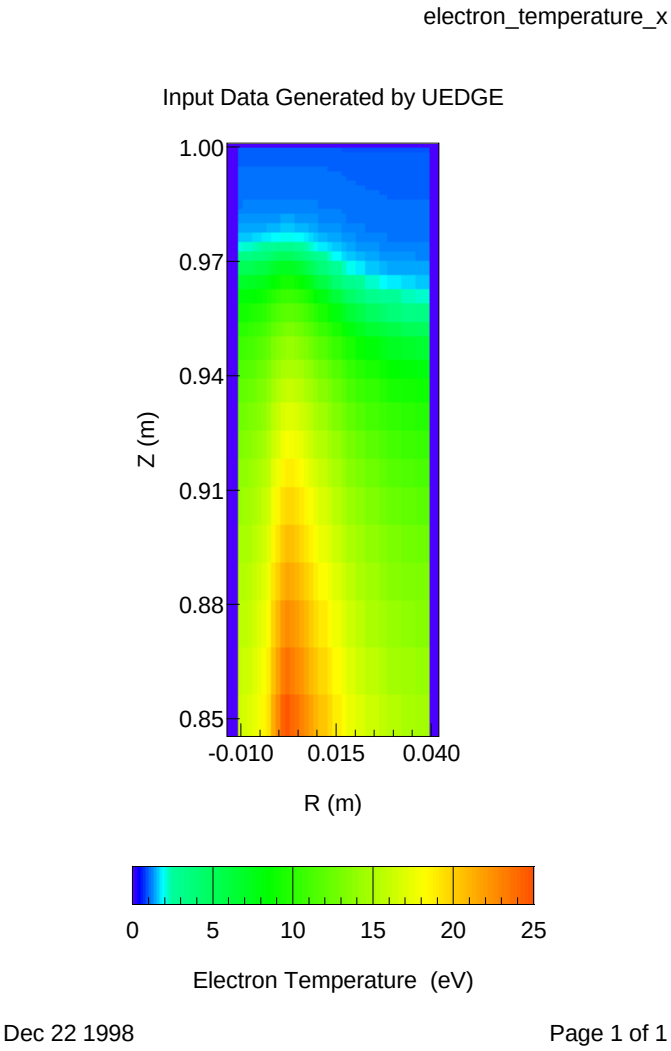

Figure 4.1: Electron temperature near the target ($Z = 1$ m) as computed by UEDGE with its fluid neutral model.

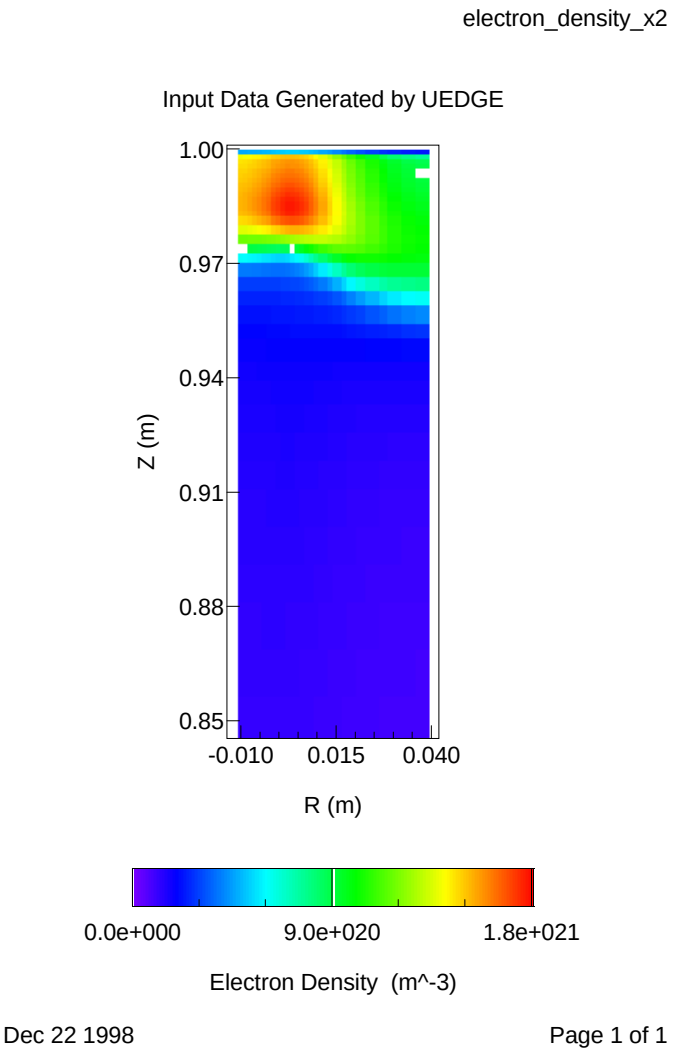

Figure 4.2: Electron density near the target ($Z = 1$ m) as computed by UEDGE with its fluid neutral model.

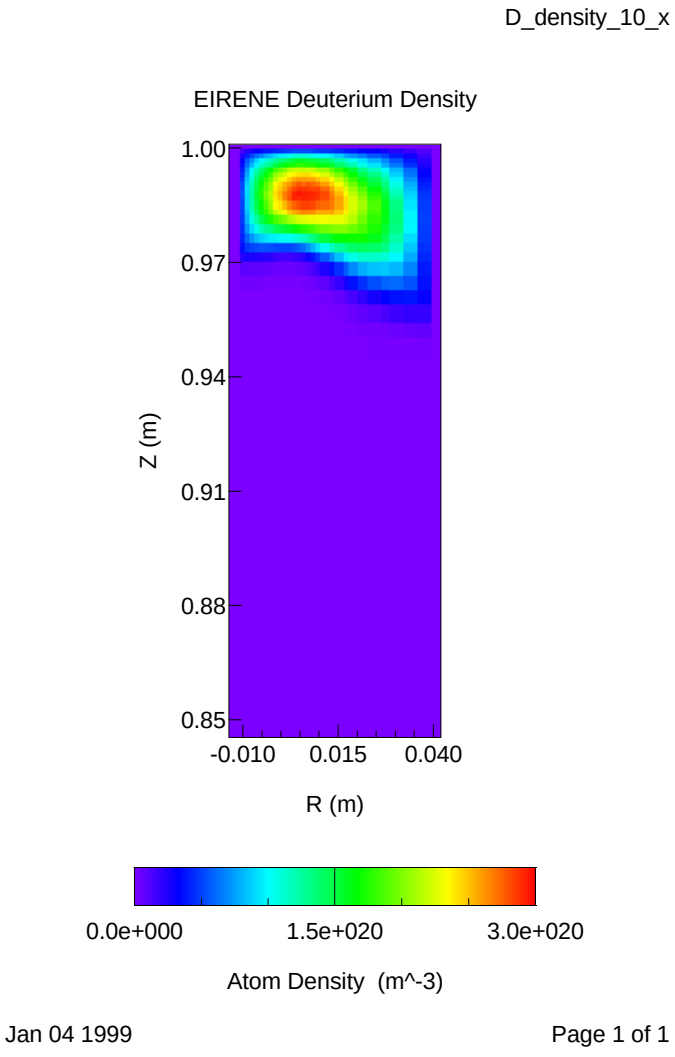

Figure 4.3: Neutral deuterium atom density computed by EIRENE.

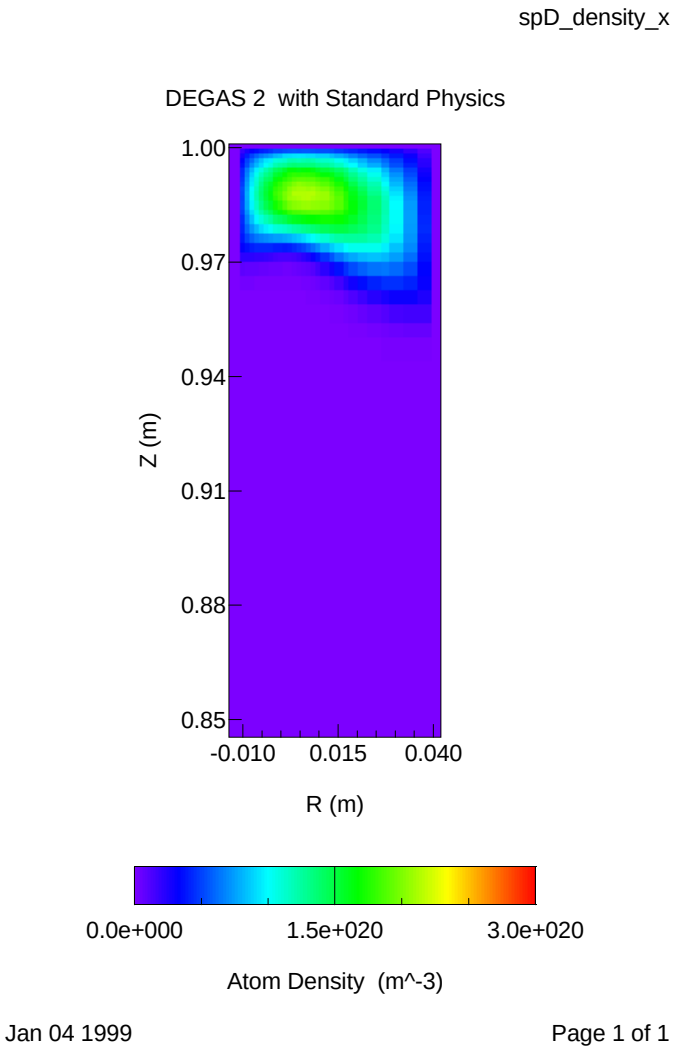

Figure 4.4: Neutral deuterium atom density computed by DEGAS 2 using "standard physics".

4.4 Statistical Basis for Comparisons

Because Monte Carlo codes provide only a statistical description of a result, e.g., the neutral density at a particular point in space and an estimate of its error, comparisons between codes require some care. Differences between codes smaller than the standard error in their results cannot be discerned, as this section will demonstrate. We will describe a quantitative procedure for comparing code results which can be used to tell if the differences cannot be statistical in nature.

The standard error decreases with $1/\sqrt{N}$, where N is the number of flights, and can in principle be made as small as desired. For the problem at hand, we've used 80,000 flights for each of the source groups, resulting in standard errors of a few percent near the plate for simple tallies such as the neutral atom density. The differences between EIRENE and DEGAS 2 remaining after the steps described in Sec. 4.5 have been taken are slightly larger than this, so raising the number of flights further would not be useful.

Both DEGAS 2 and EIRENE compute the recombination contributions to the source terms (e.g., subtracting recombination rate from the ion particle source) directly and add them to the final Monte Carlo results at the end of the run. DEGAS 2 does provide the option for computing these contributions in the standard Monte Carlo way (as a check), but the statistical error is increased substantially. Comparison of the code results is, thus, further complicated because the computed variance estimates cannot be used directly with the final output results. Rather, our statistical comparisons must be made *without* these "post-processing" contributions.

DEGAS 2 has two sets of output arrays:

1. One written before post-processing and containing relative standard deviations,

2. One with the post-processing results and no variance data.

The screen output generated by EIRENE likewise contains results without these post-processing contributions and their relative standard deviations, provided they were requested in the input file. These data along with the first of the two DEGAS 2 output array sets will be analyzed with the procedure outlined here.

The principal basis our statistical comparison is the Central Limit Theorem[2]. Say we have the density score in a particular zone from a run of K flights; call it μ_K. This result is effectively a sample drawn from a parent population; call its

mean m and the standard deviation σ. The objective of the Monte Carlo method is to estimate this mean m.

The Central Limit Theorem then says that the relative error

$$\epsilon_{\rm rel} \equiv \frac{|\mu_K - m|}{\sigma_K}, \tag{4.1}$$

where $\sigma_K = \sigma/\sqrt{K}$ is the standard error, has a Maxwellian distribution. That is, the probability that $\epsilon_{\rm rel} < 1$ is 68.3%, < 2 is 95.4% and < 3 is 99.7%.

To apply this directlyto our situation, we would need to:

- Do many runs of both codes,
- Find values of m for each code which are consistent with those results,
- Compare the m values between codes,
- Repeat the process for each zone.

Such a process is too lengthy to permit extensive comparisons of different variables and input conditions. We propose using instead a simpler, compromise procedure. Say we have a density score from DEGAS 2 $\mu_K^{\rm D}$ and one from EIRENE $\mu_{K'}^{\rm E}$, We begin by *postulating* that they have the same m, If this is true, the random variable $z \equiv \mu_K^{\rm D} - \mu_{K'}^{\rm E}$ has mean $m_z = 0$, Then, if each of the codes' standard deviation estimates $s_K^{\rm D}$ and $s_{K'}^{\rm E}$ are also consistent with σ, one can show that the distribution of z about its mean 0 will have a standard deviation

$$s_{z,K,\rm eff} = \sqrt{(s_K^{\rm D})^2 + (s_{K'}^{\rm E})^2}, \tag{4.2}$$

This is just what one expects from propagation of errors.

We are not done yet in that applying this would still require many runs of both codes. However, we do have many zones in each run. *If* the scores in each zone are uncorrelated with other zones, we could treat the value of

$$\epsilon_{z,\rm rel} = z/s_{z,K,\rm eff} \tag{4.3}$$

in each zone as a separate sample and compile a distribution of $\epsilon_{z,\rm rel}$ by comparing the results in each zone. If we find this distribution to be Maxwellian, we may infer that our postulates are correct, i.e., the codes are giving the same results. If the distribution strongly deviates from a Maxwellian, we can suspect that the codes do not agree.

The extent to which scores in two zones are correlated depends on many factors. Since we do not require precise statistical agreement between the codes (a few percent of systematic error is acceptable), we will assume that the effect of these correlations is too small to prevent this test from detecting significant (more than a few percent) differences.

One other consideration is that the Central Limit Theorem only applies for "large enough K". For a given zone, this effectively means a "small enough relative standard deviation". For these comparisons, both codes produce relative standard deviations near the target of a few percent. If we restrict the comparison to zones with relative standard deviations below some value, say $\sigma_{\max} = 20\%$, we should still have a large number of zones to work with.

Note that the codes output the relative standard deviation, $\sigma/(m\sqrt{K})$. To get σ_K^{D} we multiply the output relative standard deviation by μ_K^{D}.

4.5 Code Differences

4.5.1 Atomic Physics

The common ancestry of EIRENE, DEGAS, and DEGAS 2 is apparent in their atomic physics data. All three codes rely upon the molecular data in the Janev book[16] (see Sec. 4.5.5, however). The reaction rates are explicitly the same in all three codes, as are the dissociation energies, and the electron energy loss rates. Note also that since the scoring of momentum transfer to the plasma in EIRENE is relatively new, its implementation is incomplete: there is no momentum transfer for the molecular dissociation processes. However, given the dominance of the other momentum sources, this should not be a problem. DEGAS 2 tracks all three components of the momentum transfer in all reactions.

This version of EIRENE (with the present input file) uses the reaction rate for charge exchange in the Janev book[16]. No attempt is made to ensure that the sampled background ion velocities are consistent with the charge exchange cross section[3]. Furthermore, the momentum and energy transfer expressions are correct only in the limit of $\langle \sigma v \rangle = \sigma v$ (see Sec. 4.5.6). More recent versions of the code certainly do better than this[17].

The DEGAS 2 standard charge exchange data comes from the Janev-Smith database[18], with consistently computed reaction rates, and energy and momentum transfer rates[17]. The simulation described in Sec. 4.3 used these data. For subsequent runs, data equivalent to those in EIRENE will be substituted.

DEGAS 2 and EIRENE each rely on their own collisional radiative[17] codes for the multi-step electron-impact ionization and recombination of hydrogen data. These two codes have been compared several times in the past. The basic difference is that the EIRENE code, AMJUEL, is based upon the Johnson-Hinnov ionization cross sections[19]. The data used in DEGAS 2 are described in Sec. 2.10.

Figures 4.5 and 4.6 compare the rate data. The AMJUEL ionization rates are noticeably lower for $T_e < 10$ eV. This is likely responsible for most of the differences between the densities in Figs. 4.3 and 4.4. The energy loss rates agree. However, when normalized to the ionization rate, to give the energy lost per ionization, the differences of Fig. 4.5 are factored in.

Figure 4.7 shows that the recombination rates agree well. The reason is that the Johnson-Hinnov[19] recombination cross section data are used in both collisional radiative codes. The energy exchange rates agree for $T_e < 10$ eV, but disagree at higher temperatures. This quantity includes the average energy of a radiatively recombining electron; the expression used to compute it involves[17] $d\langle\sigma v\rangle_{\text{recombination}}/dT_e$. Apparently, the AMJUEL values were obtained by differentiating a fit to $\langle\sigma v\rangle_{\text{recombination}}$. The values used in DEGAS 2, however, were obtained by differentiating closed form expressions for $\langle\sigma v\rangle_{\text{recombination}}$ and should be correct.

As with charge exchange, in Sec. 4.3 DEGAS 2 used the `ehr2.dat` data. For subsequent runs, the AMJUEL data will be used in both codes.

4.5.2 Source Sampling

The ion current distribution to the target is sampled according to the relative fractions of current striking each segment (i.e., each radial zone). Likewise, the probability distribution used in sampling the "birth" zone of a recombining ion is given by the fraction of the total recombination occurring in that zone. Whether or not a code is correctly sampling these distributions can be easily determined. For DEGAS 2, the `sourcetest` utility was written to do just this (see Sec. 3.4.3). Cruder means were used to extract the requisite data from EIRENE.

Figure 4.8 shows that the two codes are able to match the input distribution (to within the error bars). To make the comparison quantitative, we use a χ^2 test[51]. The result is p, the probability that sampled distribution matches the parent distribution. For the 5000 samples used in generating Fig. 4.8, the DEGAS 2 results yield $p = 0.97$. For EIRENE, we get $p = 0.69$. Since p in both cases is not too much smaller than 1, we conclude that the input distribution is being adequately sampled.

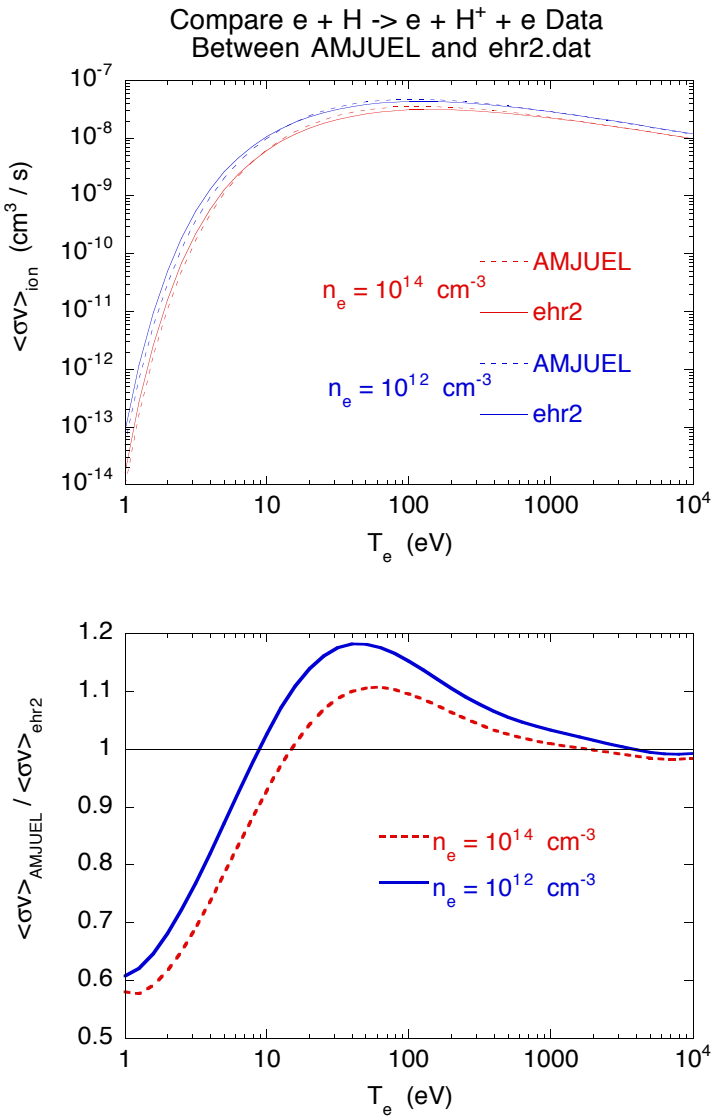

Figure 4.5: Comparison of effective collisional radiative electron impact hydrogen ionization rate from EIRENE's AMJUEL data file and from DEGAS 2's IRLS code (contained in the publicly distributed file `ehr2.dat`) at densities of 10^{18} m^{-3} and 10^{20} m^{-3}. The lower figure shows the ratios of the rates computed by the two codes.

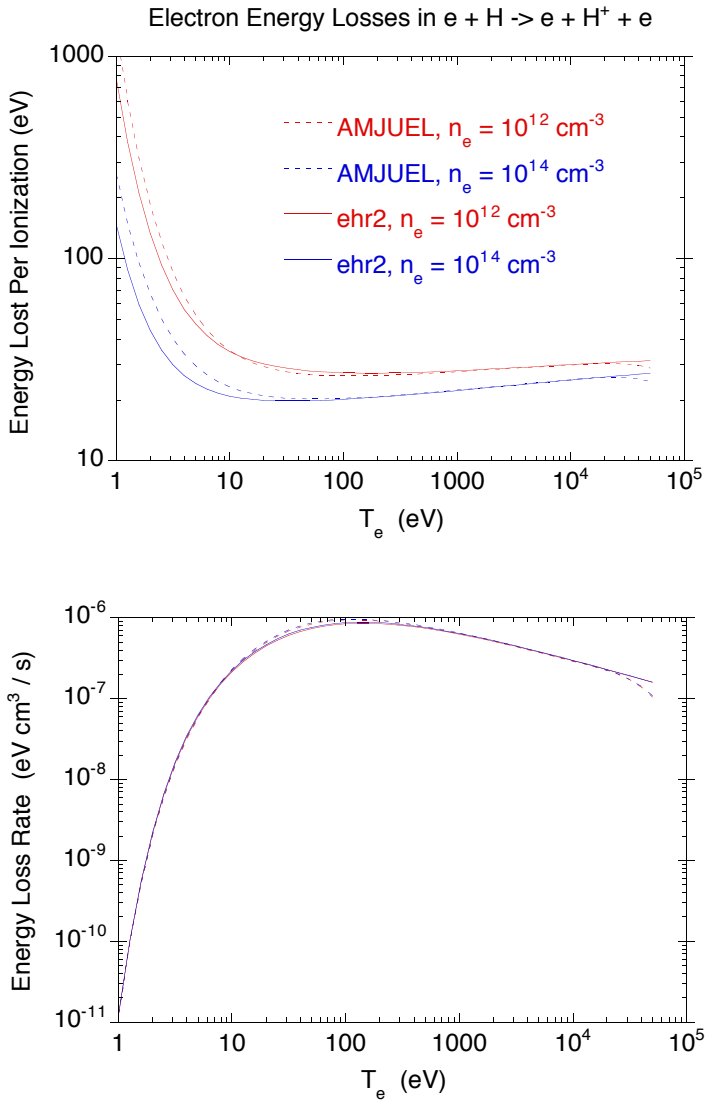

Figure 4.6: Comparison of effective collisional radiative electron energy loss rate from EIRENE's AMJUEL data file and from DEGAS 2's IRLS code (contained in the publicly distributed file `ehr2.dat`) at densities of 10^{18} m^{-3} and 10^{20} m^{-3}. The upper figure shows the energy lost per ionization; the lower gives the power lost.

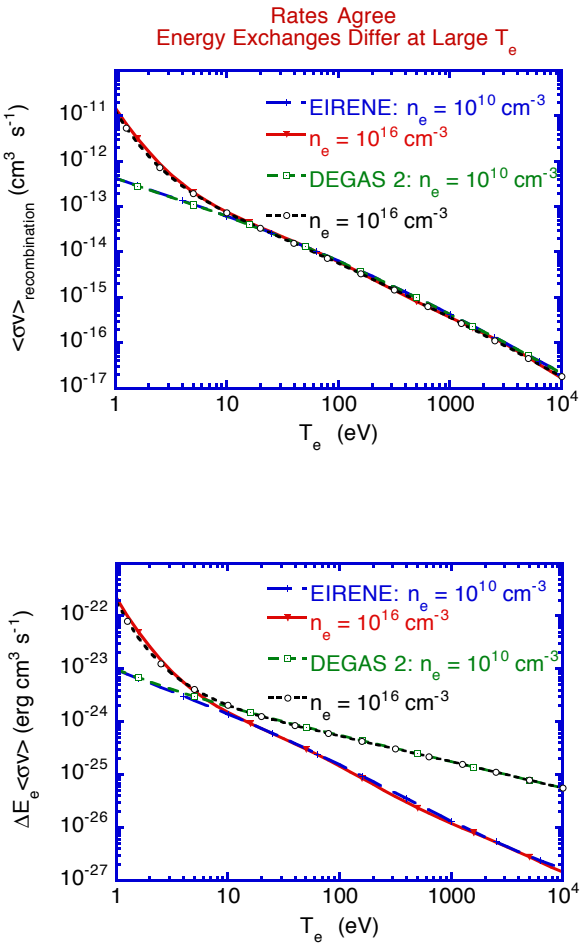

Figure 4.7: Comparison of effective collisional radiative electron recombination rates from EIRENE's AMJUEL data file and from DEGAS 2's IRLS code (contained in the publicly distributed file `ehr2.dat`) at densities of 10^{16} m^{-3} and 10^{22} m^{-3}. The upper figure shows the effective recombination rates; the lower gives the power lost.

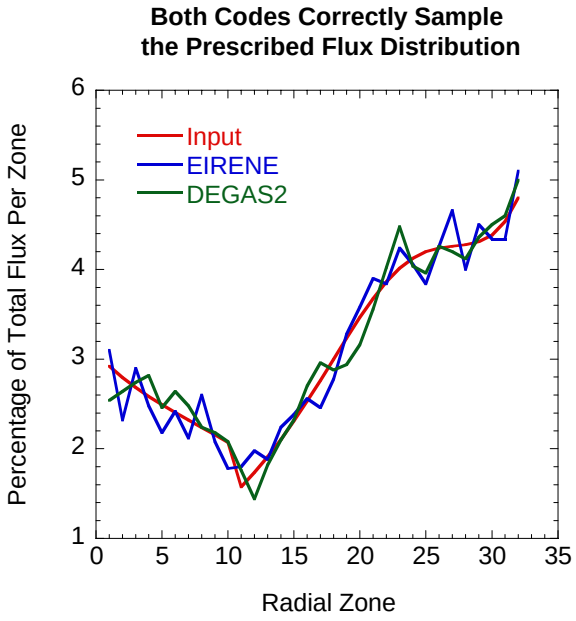

Figure 4.8: Sampled target flux distributions, as a function of radial zone, for DEGAS 2 and EIRENE. The curve labeled "Input" is the ideal result.

We use 50,000 samples of the recombination source. Because of the large number of zones, a simple plot analogous to Fig. 4.8 is unintelligible. Furthermore, not all of the problem zones are adequately sampled to permit the application of a χ^2 test to the whole problem space. Instead, we isolate the 648 zones with highest recombination rate (the lowest temperature region, chosen somewhat arbitrarily). Applying a χ^2 test to samples taken in those zones yields for DEGAS 2 $p = 0.94$. But, for EIRENE $p = 0.21$. Sensing that this is small enough to cause concern, we repaired a known problem with the EIRENE random number generator (as received, the code had random number seeds which differed by unity; the fix is to make one of these integers very different). The result is much improved: $p = 0.79$. All other EIRENE runs described in this chapter utilize this bug fix.

4.5.3 Energy Distribution of Reflected Atoms

The EIRENE surface physics is more detailed than that of the old DEGAS and, hence, DEGAS 2. Consequently, we will not make an overall comparison with different surface physics data. Instead, we have translated the Bateman format data for reflection of D off of Mo taken from EIRENE's `TRIM` input file into a DEGAS 2 format. To be more precise, the Mo data obtained from Fe data using a scaling argument[52].

These data prescribe the reflection coefficient, the outgoing energy, and two outgoing angles as a function of the incident energy and polar angle. The Bateman format is described in detail elsewhere[3]. Briefly, these data specify these parameter values (e.g., the outgoing energy) at intervals of the cumulative distribution function: 0.1, 0.3, 0.5, 0.7, 0.9. One can also establish data points at each end of the distribution. For example, a minimum energy might be the wall temperature; a maximum would be the incident energy. As originally implemented in DEGAS 2, this additional information to extrapolate beyond the ends of the data. Doing this carefully required specifying additional cumulative distribution values at 0, 0.2, 0.4, 0.6, 0.8, and 1.0 (intermediate values being obtained by linear interpolation).

EIRENE, however, mindlessly extrapolates the data and enforces the minimum and maximum values after sampling. The result is that the sampled maximum and minimum may be noticeably different from what one expects. The first set of benchmark runs indicated that the difference in the handling of the lowest reflected energy atoms was significant. The reason is that the atom density near the target plate (i.e., where ionization is lowest) scales like the inverse of the atom velocity, and hence is impacted strongly by the lowest energy atoms. To eliminate this difference, we reset the lower bound on the energy so DEGAS 2 would mimic

the EIRENE extrapolation. Figure 4.9 shows that the original DEGAS 2 reflected energies for a normally incident 19 eV atom match the input data well (the difference at the lower end is due to the use of logarithmic interpolation in DEGAS 2). The revised DEGAS 2 reflected energies differ from the input, but better match the values obtained from EIRENE.

A second difference becomes significant with the present set of benchmarks including recombination. The temperatures in this plasma are sufficiently low that a significant number of atoms are striking the target with energies below 1 eV. EIRENE assumes that such atoms are always absorbed (and desorbed thermally as molecules in this simulation). DEGAS 2 treats the data literally; the reflection coefficient is about 0.2 at 1 eV. For the initial comparison run (see Sec. 4.3), DEGAS 2 was run in this manner, although the energy and angular extrapolation changes described above and in the next subsection were implemented. For all subsequently discussed benchmark simulations, DEGAS 2 will assume absorption for incident atoms of less than 1 eV energy.

4.5.4 Angular Distribution of Reflected Atoms

The cosines of the angular distribution are sampled in a way analogous to the energy. In this case, the bounds of the sampled parameters are clearer still since they are the cosines of the angles (i.e., 0 and 1). However, during this benchmark exercise, a discrepancy in the near target deuterium density was again connected with the distribution of neutral velocities in the Z direction. With the difference due to energy extrapolation (see Sec. 4.5.3) having been eliminated, the cause this time was traced to the extrapolation of the cosine of the outgoing polar angle, $\cos(\theta_{\rm out})$.

Figure 4.10 shows the sampled cumulative distribution from two DEGAS 2 runs and from an EIRENE run. These data were compiled from all of the ions incident on the target in a particular zone (normal incidence at 6.88 eV). The curve labeled "original" shows the DEGAS 2 treatment extending down to near $\cos(\theta_{\rm out}) \simeq 0$. The "revised" curve demonstrates that our modifications to the DEGAS 2 data successfully mimic the EIRENE extrapolation behavior.

4.5.5 H_2 Dissociation Rate

Significant differences in the electron energy sink were traced to a discrepancy in the H_2 dissociation rate (reaction 2.2.5 in the Janev book[16]) at the lowest electron temperatures, $T_e \simeq 1$ eV. Unfortunately, both the plot and the fit for these

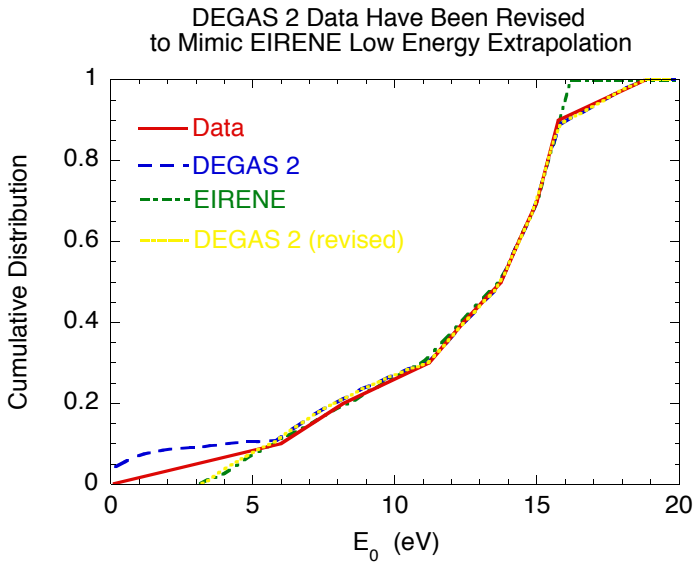

Figure 4.9: Comparison of the cumulative distribution of reflected energies sampled in DEGAS 2 and EIRENE with the input data obtained from EIRENE's `TRIM` file. The "revised" DEGAS 2 curve indicates that the code is now able to mimic the simple extrapolation used in EIRENE. These data are for a 19 eV deuterium atom incident on molybdenum.

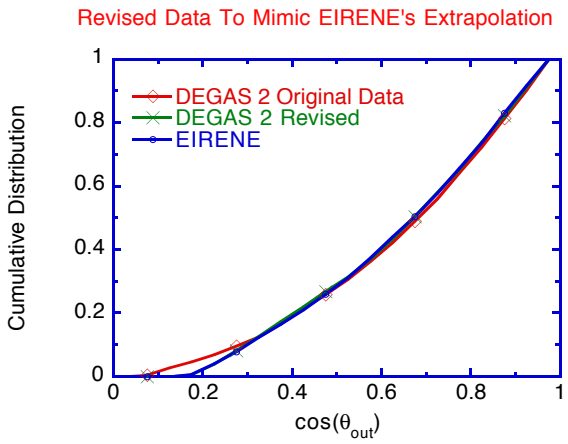

Figure 4.10: Comparison of the cumulative distribution of reflected outgoing polar angles sampled in DEGAS 2 and EIRENE with the input data obtained from EIRENE's `TRIM` file. These distributions were compiled from all of the initial incident *ions* (i.e., at the start of the flight) in a particular zones. All of these ions strike the target normally with 6.88 eV.

data in the Janev book are *wrong*. One needs to refer to the corresponding preprint even to get the correct values for the fit coefficients. EIRENE has those values, but applies them down to the smallest T_e used in the code, 1 eV. However, the fit is valid down to only 1.2 eV. The dissociation rate values used in DEGAS 2 at these lowest temperatures were taken directly from the tabular data provided with the original DEGAS code[3], not from the fit. The point is that these values differ from those computed by the fit (presumably due to the fit not being valid there). No record explaining the origin of the DEGAS values, but we assume that they were taken from the initial data compilation from which the fit was derived. Regardless, the remedy for the objectives of this benchmark is to replace these low T_e dissociation rates with the same values used in EIRENE. The results of Sec. 4.3 have the DEGAS values; runs discussed in subsequent sections will have the EIRENE values.

4.5.6 Estimator Differences

With DEGAS 2 run as in Sec. 4.3, the relative standard deviations for the electron and ion energy sources were substantially larger than those for EIRENE (these differences were most noticeable close to the target). The cause of the higher electron energy source variance was that DEGAS 2 was using a collision estimator for the molecular contributions to that quantity (for historical reasons), while EIRENE was using a track length estimator. Switching the estimator used in DEGAS 2 required a minor change to `tallysetup` (see Sec. 3.4.1). At the same time, we changed the scoring of the electron impact ionization of deuterium contributions to the ion energy and momentum sources from being done "post process" (i.e., computed from the neutral density at the end of the run) to a track length estimator. This allowed the variance contributions from this process to be accounted for in these scores and was essential for comparing with the EIRENE values (see Sec. 4.4).

The main reason for the larger ion energy source variance in DEGAS 2, however, was that charge exchange was being treated by the collision estimator, rather than the track length estimator used in EIRENE. The original reason for choosing the collision estimator in DEGAS 2 was that the charge exchange data file did not have the required integrals of the cross section needed to score the energy and momentum exchanges[17]. We have now worked out the forms these integrals must have to emulate the EIRENE assumption of constant σv and incorporated them into the charge exchange data file.

The runs of Sec. 4.3 do not employ these estimator changes (and utilize a

different charge exchange cross section altogether, as was noted previously), while runs in subsequent sections have these modifications incorporated.

4.5.7 Other Differences

We suspect that there are other differences remaining which will prevent the two codes from being brought into closer agreement. For example, DEGAS 2 uses a logarithmic interpolation in incident energy for the Bateman format reflection data. EIRENE assumes a linear interpolation. The interpolation schemes in DEGAS 2 presently require a uniform spacing, either linearly or logarithmically, between data points (a nonuniform spacing option was tried at one point, but abandoned as too difficult to maintain). Hence, the interpolation technique used by EIRENE cannot be emulated without significant changes to the code. On the other hand, the subtlety of some of the differences which have been detailed in this section suggests that remaining nonstatistical discrepancies in the code results are unlikely to be of physical significance and are consistent with minor numerical differences between the codes.

4.6 Characterize Comparison

The principal means of comparing the code results is now considered to be the quantitative prescription given in Sec. 4.4. We will, however, begin this section with plots to illustrate two points.

4.6.1 Density Differences

In Fig. 4.11, we show the relative error,

$$\epsilon_{\mathrm{rel}} = \frac{n_D^{\mathrm{D}} - n_E^{\mathrm{E}}}{\sigma_{\mathrm{rsd}}^{\mathrm{D}} \sqrt{(n_D^{\mathrm{D}})^2 + (n_E^{\mathrm{E}})^2}}. \tag{4.4}$$

This expression is equivalent to Eq. (4.3) with σ_{rsd} being the relative standard deviation. Within the `geomtesta` post-processor (see Sec. 3.4.1), we do not have access to the EIRENE error estimates (see Sec. 4.4). However, in the case of the deuterium density, the EIRENE and DEGAS 2 relative standard deviations are sufficiently similar that we can treat them as equal for this purpose. Both are on the order of a one to a few percent near the target. The scale in Fig. 4.11 indicates

the number of standard errors between the DEGAS 2 and EIRENE density values. Note that this is a signed quantity. The white zones indicate areas with differences greater than 3 times the standard error; the black areas indicate likewise that the lower bound has been exceeded.

The dominance of the green areas is indicative of the nearly complete statistical agreement (see Sec. 4.6.3). The red, orange, and yellow zones near the target likely point to a remaining systematic discrepancy between the codes, as suggested in Sec. 4.5.7.

4.6.2 Comparison of Momentum Source

As indicated by Figs. 4.12 and 4.13, the deuterium ion parallel momentum sources computed by the two codes are qualitatively (i.e., visually) similar. However, the statistical errors are extremely high (Fig. 4.14), presumably caused by the tight coupling with the background ions through charge exchange (i.e., the net momentum source due to charge exchange is the difference of two nearly equal numbers).

The lack of regions with small relative standard deviations makes comparison of the code results (see Sec. 4.4) problematic since we require a large number of (independent, ideally) zones with "good statistics" (so that the Central Limit Theorem applies) for the analysis. As will be seen in Sec. 4.6.3, relaxing one of these two constraints permits a comparison to be made. In each case, the results are consistent with the two codes being in agreement. The deuterium ion energy source likewise has significant relative standard deviations, although smaller than those shown in Fig. 4.14. The consequences for the stability of a coupling to a plasma transport code may be more serious, however. Future work will assess and remedy this situation, if necessary.

4.6.3 Results of Quantitative Comparison

The procedure outlined in Sec. 4.4 must be amended slightly to accomodate the remaining few-percent systematic differences in some of the code results. The relative error is given instead by

$$\epsilon_{\text{rel}} = \frac{\mu_K^{\text{D}} - \mu_{K'}^{\text{E}}}{\max\left(\sqrt{(s_K^{\text{D}})^2 + (s_{K'}^{\text{E}})^2}, \sigma_{\min}\frac{\mu_K^{\text{D}} + \mu_{K'}^{\text{E}}}{2}\right)}, \tag{4.5}$$

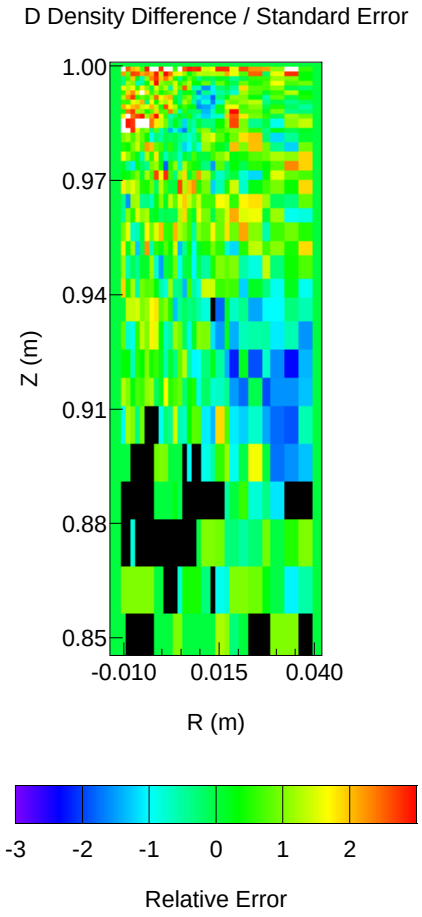

Figure 4.11: Difference between DEGAS 2 and EIRENE deuterium density relative to the standard error estimate.

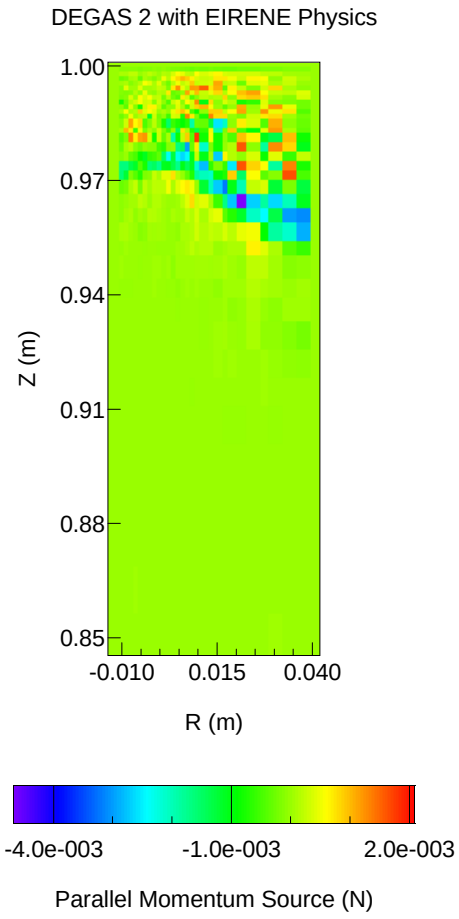

Figure 4.12: Source of deuterium ion parallel momentum computed by DEGAS 2.

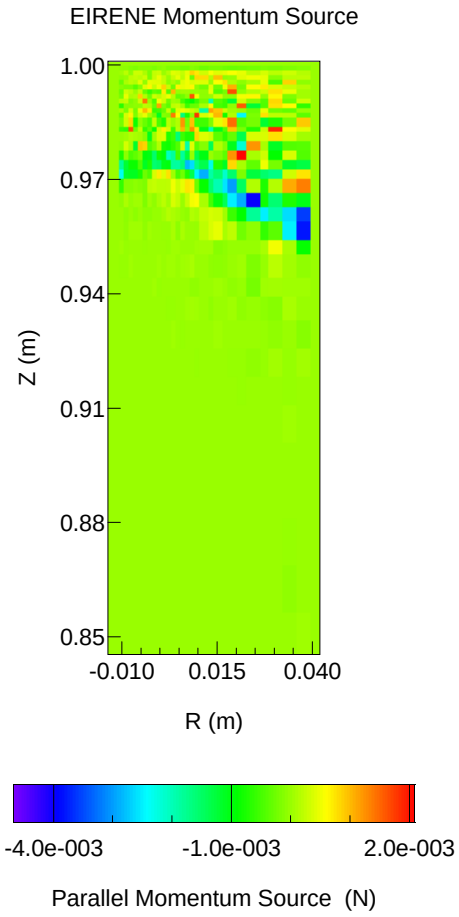

Figure 4.13: Source of deuterium ion parallel momentum computed by EIRENE.

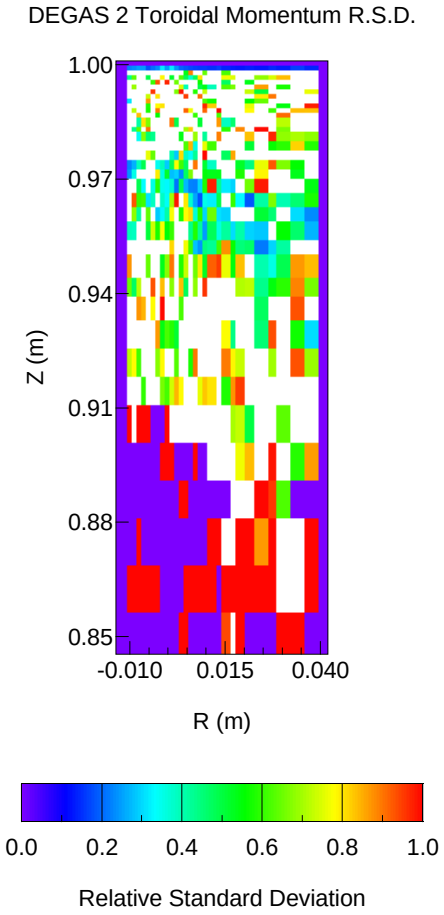

Figure 4.14: Relative standard deviation for the deuterium ion momentum source computed by DEGAS 2.

where again the s_K refers to the standard error for a simulation of K flights (i.e., $s_K = \sigma_{\mathrm{rsd}}\mu_K$). The additional parameter, σ_{min} is associated with the statistical error (expressed as a fraction, like the relative standard deviation).

For all of these except the ion energy and momentum source, $\sigma_{\mathrm{max}} = 20\%$ and σ_{min} is given. For those two, $\sigma_{\mathrm{min}} = 0$ and σ_{max} is given. In the former case, the relative standard deviation is required to be $< 20\%$ to ensure that the Central Limit Theorem is applicable and σ_{min} represents the systematic error. Of the tested values of σ_{min} the one which yields a set of percentages most closely matching the Maxwellian ideal (68%, 95%, 99.7%) is taken as our estimate of the systematic error. For the ion energy and momentum sources, we are unable to get "good statistics" with $\sigma_{\mathrm{max}} = 20\%$; hence, larger values are tested. The relative standard deviations are too large for any systematic error to be detected, so we use $\sigma_{\mathrm{min}} = 0$.

Source:	Plate			Recombination		
	1σ	2σ	3σ	1σ	2σ	3σ
D Density						
$\sigma_{\min} = 7\%$	82%	99.2%	100%	90%	99.4%	99.9%
$\sigma_{\min} = 5\%$	75%	95%	99.8%	82%	98%	99.9%
$\sigma_{\min} = 0\%$	56%	86%	96%	52%	88%	99%
D_2 Density						
$\sigma_{\min} = 0\%$	64%	92%	98%	59%	92%	99.2%
D_2^+ Density						
$\sigma_{\min} = 0\%$	63%	92%	100%	68%	97%	100%
D^+ Ion Source Rate						
$\sigma_{\min} = 7\%$	74%	99.6%	100%	83%	99.3%	100%
$\sigma_{\min} = 5\%$	57%	94%	100%	63%	98%	99.8%
$\sigma_{\min} = 0\%$	38%	73%	93%	38%	66%	85%
Electron Energy Source Rate						
$\sigma_{\min} = 7\%$	92%	99%	99.1%	93%	99%	99.3%
$\sigma_{\min} = 5\%$	79%	98%	99.1%	86%	98%	99.1%
$\sigma_{\min} = 0\%$	38%	72%	91%	45%	78%	94%
D^+ Ion Energy Source Rate						
$\sigma_{\max} = 70\%$	69%	94%	99%	68%	96%	99.7%
$\sigma_{\max} = 50\%$	69%	94%	99.1%	65%	97%	100%
$\sigma_{\max} = 20\%$	68%	94%	100%	61%	98%	100%
D^+ Ion Parallel Momentum Source Rate						
$\sigma_{\max} = 70\%$	72%	88%	99.2%	75%	92%	99.4%
$\sigma_{\max} = 50\%$	74%	97%	100%	76%	97%	99%
$\sigma_{\max} = 20\%$	58%	89%	100%	0%	0%	0%

The two molecular densities fare the best, given nearly Maxwellian percentages without allowing for any systematic error. The atom density, electron energy source (which is given by the atom density times a function of T_e, apart from a small contribution due to molecules), and ion source rate are all roughly consistent with a 5% systematic error. Again, the ion energy and momentum sources are too noisy to permit an estimate of the systematic error. However, these numbers indicate that the two codes do agree within the estimated statistical errors.

4.7 Performance Benchmark and Optimization

Although efficiency was kept in mind during all stages of the design process, no effort was made to optimize the performance of DEGAS 2 along the way. In this section, we describe the benchmark of the code's performance against EIRENE. Speed bottlenecks were discovered through profiling. Improvements eliminating those bottlenecks.were implemented. Some involved significant code revisions; others were effected by simple changes to source or input files. The end result was that the run time for DEGAS 2 was roughly equal to that of EIRENE within the variations normally experienced in a time-sharing environment.

The implemented code revisions were motivated by profiling the code to determine which subroutines were the most time-consuming. At one point, assignments and comparisons of string variables were using significant fractions of the run time. The strings responsible were being used to describe auxiliary data associated with the atomic physics reactions. The code was modified to compile a list of all of these strings at the beginning of the run and to use an integer pointer into that list for the run-time comparisons and assignments. Because these code changes were pervasive, their impact on the run time could not be documented as carefully as the other improvements noted below. Roughly, however, they resulted in a reduction of about 10 seconds per 1000 flights.

4.7.1 Starting Point

The starting assumptions were the same as those associated with the code as run for the initial EIRENE benchmark (specifically, the DEGAS 2 version with the CVS tag `V1_8a`). The "EIRENE physics" set was used to minimize differences due to the number of collisions and to permit direct comparisons of the variances.

For each code configuration, runs were performed at 1000, 2000, and 4000 flights. The time for the 1000 flight case was subtracted from the other two and the results used to estimate the incremental time required to track 1000 flights. The idea is that production runs would be substantially longer than these (with relative standard deviations on the order of 1%) so that the overhead associated with input and output would be negligible. In a few cases, production length runs (e.g., 80,000 flights) were done; for those, the run time was just taken to be the time consumed divided by 80.

These runs were performed on a Sun UltraSPARC 2 running SunOS 5.5.1 and V. 4.2 of f77 with `-O4` optimization. Again, because this computer is a time-sharing system, these run times are only approximate. The estimated error is about

10%. The baseline run time for DEGAS 2 was: 136 seconds per 1000 flights.

4.7.2 Charge Exchange Rejection

The charge exchange cross section depends on the relative ion-neutral velocity. However, collisions are decided upon using a cross section averaged over a Maxwellian distribution. In the baseline, the ion collision partners were chosen so as to be consistent with the actual cross section using a rejection technique, sampling several ions before finding one which is satisfactory. The version of EIRENE used in this benchmark does not do this (neither did the original DEGAS).

For the first change, charge exchange rejection was turned off. A cursory examination of the results showed no significant impact on the neutral density. However, a stronger effect may have occurred elsewhere (e.g., energy transferred to background species). Run time after disabling charge exchange rejection: 119 seconds per 1000 flights.

4.7.3 Reduce Number of Scores

One impressive feature of EIRENE is how thoroughly its default operation has been pared down to the minimum necessary for coupling to the fluid plasma codes. In particular, no data on the variances are kept. DEGAS 2 was written with the philosophy that the mean value for a score is meaningless without a corresponding variance, and the two data values are kept together. For this comparison, a short list of variances was requested in the EIRENE input file. These will be needed later.

As presently distributed, DEGAS 2 by default sets up about 14 scores or tallies. This list was reduced to 7 at this step (later, 3 more will be eliminated leaving the neutral density and the 3 scores representing the particle, momentum, and energy transfer to the background species). Run time after cutting the number of scores to 7: 95 seconds per 1000 flights.

4.7.4 Compression of Scores

The first offender in the profiling exercise was the routine responsible for compiling the scores accumulated during a single flight into the global total. As originally written, both the total and incremental scoring arrays were full-sized (roughly the number of zones times the number of scores times the number of background

species). However, each flight visits only a small fraction of the problem space, and this routine was spending a significant amount of time adding 0 to 0.

These scoring arrays were modified to contain only the non-zero scores, with an array of pointers mapping their contents back to the corresponding locations in the full-sized arrays (which were still used for the final output stage). Run time after implementation of compressed scoring: 49 seconds per 1000 flights.

4.7.5 Other Changes with Minimal Effects

Further reducing the number of scores from 7 to 4 had little impact on the run, probably because of the use of compression at this point.

DEGAS 2 (and EIRENE) employ a 10 term quadratic representation of all surfaces in the geometry. In some problems, such as this one, only the linear terms are needed. The inclusion of the quadratic terms was made a compile-time option so that they could be turned off for fully linear geometries. Earlier tests of the impact of this change showed a reduction of about 4 seconds per 1000 flights. However, in this more systematic series of trials, the improvement could not be quantified with certainty.

4.7.6 Variance-Altering Changes

Thus far, all of the changes made led to the same final results as the run with "EIRENE physics". Most of the subsequent changes, while reducing the run time, also result in increases in the variance of the results. In a given Monte Carlo calculation, the variance is inversely proportional to the number of flights and, hence, to the run time. So, the performance Figure of Merit (FOM) is the variance times the run time. An attempt will be made below to quantify this FOM, but we can also compare qualitatively the variance over the most relevant portion of the problem space (near the target plate) via the relative standard deviation σ_{rsd}. Figure 4.15 shows σ_{rsd} for the deuterium density in the baseline case.

Below we will use σ_{rsd} for the source rate of D^+ due to ionization of neutrals. Although not computed for this baseline run, this σ_{rsd} should be comparable to that shown in Fig. 4.15 except for the first one or two zones adjacent to the plate in which the contributions from D_2 are significant.

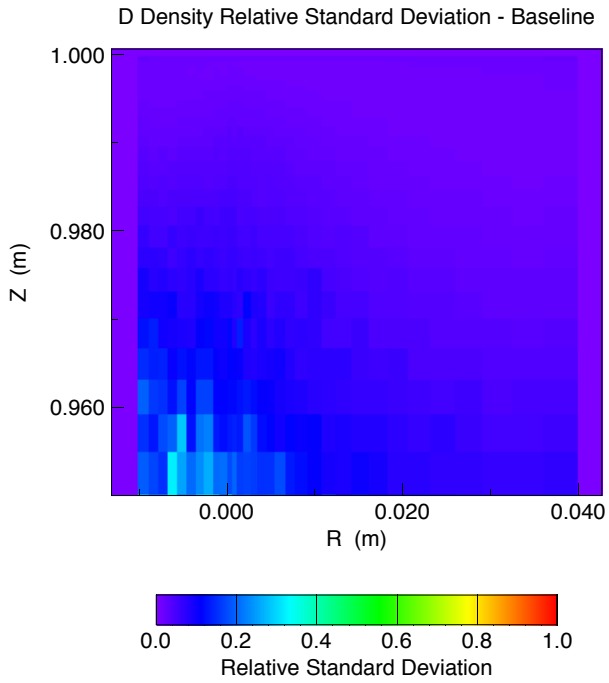

Figure 4.15: Relative standard deviation of the atomic deuterium density in the baseline run of the performance benchmark.

4.7.7 Removal of Suppressed Ionization

The simplest ("analog") Monte Carlo simulation kills off neutrals at their first ionizing collision. However, this makes finding the solution more than one mean free path from the source difficult. The predominant non-analog improvement is to assign a weight to each neutral flight (e.g., initially 1) and reduce that weight at each step to reflect the amount of ionization which should have occurred along the way. As a result, smaller variances are obtained in low probability regions of the problem. Because each flight will thus be tracked for more steps, a run using suppressed ionization will take longer. Whether or not that time is well spent depends on the problem at hand.

These EIRENE runs do not use suppressed ionization. For the purposes of comparison, we turned it off in DEGAS 2 as well. The run time without suppressed ionization dropped to 15 seconds per 1000 flights. Figure 4.16 shows the resulting σ_{rsd}.

4.7.8 Collision Estimator

Some of the scores in DEGAS 2, e.g., the neutral density and the D^+ source rate, are computed via the track-length estimator. The other scores are compiled using data only at collisions (though most of these can also be done by track-length). Since the collision routines get executed regardless of whether or not the resulting data are used in the scores, we felt that it would be interesting to switch all of the estimators to collision (except density) and eliminate the overhead associated with generating the track-length scores. Of course, doing this increases the variance as is shown in Fig. 4.17. The run with using the collision estimator (again without suppressed absorption) needed 10 seconds per 1000 flights.

4.7.9 Russian Roulette for Molecules

Two of the seven reactions involving D_2 and D_2^+ in DEGAS 2 result in two D atoms. DEGAS 2 normally tracks both of these. EIRENE, however, like the original DEGAS, chooses one of the two to follow and "kills" off the other. This is a simple example of a general nonanalog Monte Carlo technique known as "Russian roulette". Whether the use (or non-use) of this technique improves the variance again depends on the problem. For the purposes of this benchmark, Russian roulette was added to DEGAS 2's molecular dissociation routine. Those scores switched to collision estimator in the previous section were reverted to the track-

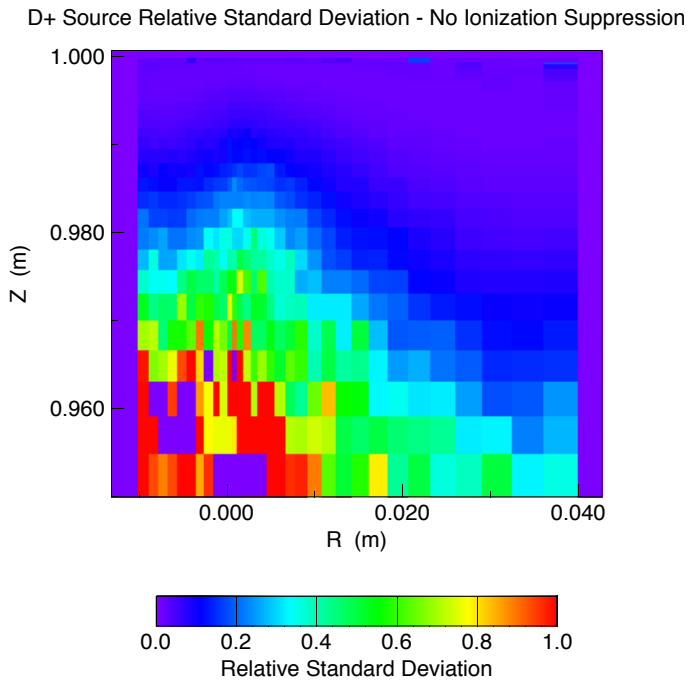

Figure 4.16: Relative standard deviation of the D^+ ion source *without* suppressed ionization.

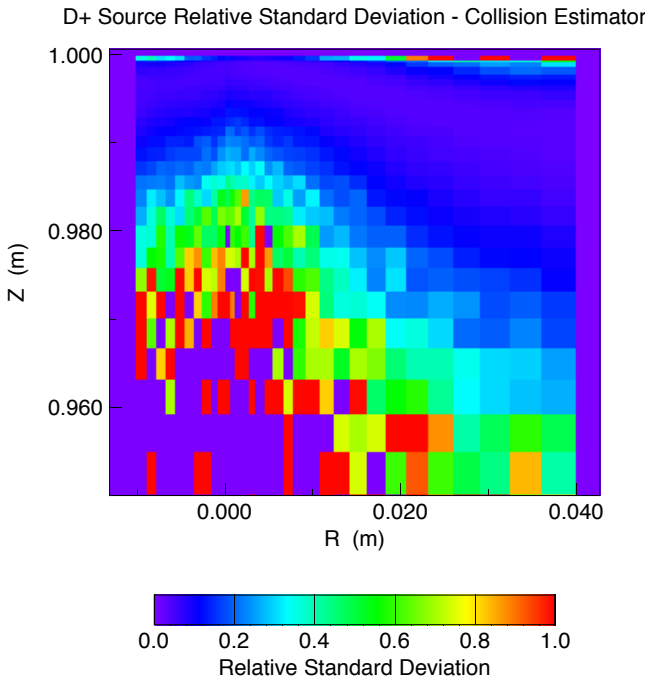

Figure 4.17: Relative standard deviation of the D^+ ion source using a collision estimator and *without* suppressed ionization.

length estimator for this configuration. Figure 4.18 shows the corresponding σ_{rsd}. This run used only 8 seconds per 1000 flights.

4.7.10 Figures of Merit

We now have four different configurations with which to evaluate this figure of merit, variance times run time. Presently, there is no single "answer" in these simulations which we can use to compute the FOM. Somewhat arbitrarily, we have selected a region of the problem spanning the width (radius) of this geometry and encompassing most of the integrated D^+ source (the selection was made by eyeballing the plot; 83% of the integrated source was included). The σ_{rsd} for each configuration was divided by that of the baseline and integrated over this region. We then defined the FOM as the product of the 1000 flight run times (quoted above) and the square (to get the variance) of this ratio.

Configuration	Seconds / 1000 flights	σ_{rsd} ratio	FOM
Baseline	49	1.0	49
No Suppressed Ionization	15	1.9	54
Collision Estimator	10	4.3	185
D_2 Russian Roulette	8	2.3	41

Clearly, using the collision estimator is not a good idea. The effectiveness of suppressed ionization could go either way. However, doing Russian roulette on the molecular products (this run was done without ionization suppression) looks to be the overall winner. Probably not by accident, this configuration is the closest to the default mode of operation for EIRENE.

4.7.11 EIRENE Performance

Of course, the whole point of this portion of the benchmark is to compare the performance of DEGAS 2 against that of EIRENE. Apart from the addition of the variance output to EIRENE, no other modifications have been made to its default mode of operation. However, the compiler optimization was changed from the `-O3` value specified with the IPP-Garching version to the `-O4` which appears to work well for DEGAS 2 with the Sun FORTRAN 77 compiler. Over several runs of length 1000 to 10000 flights, the incremental time for 1000 flights is 12 seconds. The relative standard deviations computed by EIRENE were consistent with those of the corresponding DEGAS 2 configuration (Russian roulette), although a direct comparison was not made here (see Sec. 4.5.6).

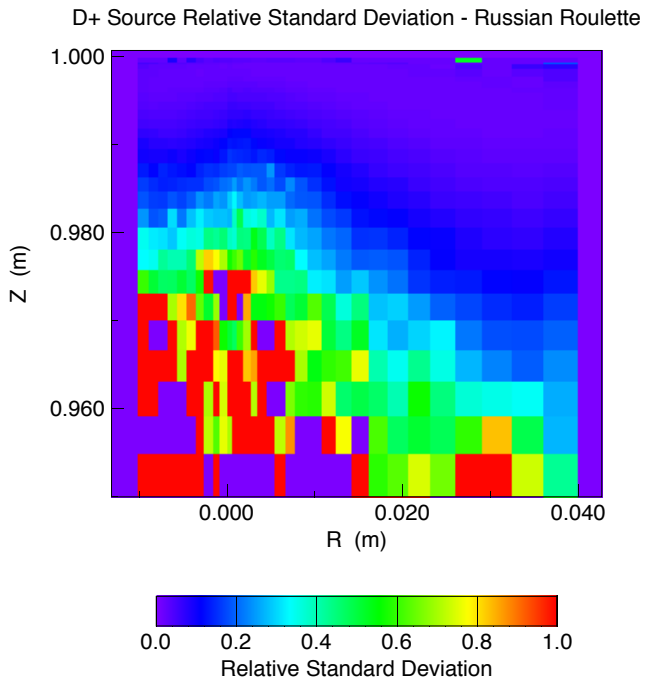

Figure 4.18: Relative standard deviation of the D^+ ion source using Russian roulette for molecules.

Within normal variations, the two codes are thus running at the same speed. The particular DEGAS 2 configuration should be chosen according to the needs of the problem at hand. Keep in mind that the original objective of DEGAS 2 was to be faster than DEGAS; the hope was that the code would yield performance comparable to that of EIRENE while retaining flexibility. That objective has been met. DEGAS 2 is also able to take full advantage of the substantial benefits of parallel processing.

Note, however, that EIRENE's computation of variances is less than optimal. Disabling them in the input file and turning off unneeded tallies reduces the incremental run time to 3 seconds per 1000 flights. At this point, EIRENE is substantially faster than DEGAS 2 since turning off its variance computation (which has been optimized) will not have a significant impact. The performance comparison between the two codes will be revisited as part of a later benchmark utilizing a more realistic divertor geometry.

The advantages of dynamic memory allocation of all run-time arrays, another design feature of DEGAS 2, have also been demonstrated during this benchmark. The run-time sizes were: DEGAS 2 - 7 MB, EIRENE - 142 MB. By reducing the dimensioning parameters in EIRENE to values more appropriate for this input file, the size of the code was reduced from 142 MB to 55 MB.

Chapter 5

Troubleshooting

5.1 Help!

Daren Stotler
Princeton Plasma Physics Laboratory, MS 27
P. O. Box 451
Princeton NJ 08543-0451
(609) 243-2063
E-mail: dstotler@pppl.gov

Bibliography

[1] The development of analytic, numerical, and Monte Carlo methods of solving the time-independent Boltzmann equation describing neutral kinetics is reviewed in:
M. Tendler and D. Heifetz, *Fusion Tech.* **11**, 289 (1987).

[2] The "bible" of Monte Carlo transport techniques is:
J. Spanier and E. M. Gelbard, "MonteCarlo Principles and Neutron Transport Problems" (Addison-Wesley Publishing Company, Reading Massachusetts, 1969).
Other references from the neutron transport field include:
J. M. Hammersley and D. C. Handscomb, "Monte Carlo Methods" (Metuchen, London, 1964).
L. L. Carter and E. D. Cashwell, "Particle-Transport Simulation with the Monte Carlo Method" (National Technical Information Service, Springfield, Virginia, 1975).
Ivan Lux and Laszlo Koblinger, "Monte Carlo Particle Transport Methods: Neutron and Photon Calculations" (CRC Press, Boca Raton, 1991).

[3] The Algorithm and applications of DEGAS are discussed in:
D. Heifetz, D. Post, M. Petravic, J. Weisheit, and G. Bateman, *J. Comp. Phys.* **46** 309 (1982).
The Proceedings from a NATO-sponsored school in Val-Morin, Canada (1984) are an invaluable reference for scrape-off layer physics. In addition to again describing the algorithms and applications of DEGAS, Heifetz covers in his article some of the basics of neutral transport in a plasma: D. B. Heifetz, in *Physics of Plasma-Wall Interactions in Controlled Fusion*, D. Post and R. Behrisch, Eds., (Plenum, New York, 1986), p. 695.

[4] D. P. Stotler et al., *J. Nucl. Mater.* **196-198**, 894 (1992).

[5] D. Reiter et al., *J. Nucl. Mater.* **220–222** 987 (1995).

[6] D. R. Bates, A. E. Kingston, and R. W. P. McWhirter, *Proc. R. Soc. A* **267**, 297 (1962).

[7] R. W. P. McWhirter, in *Plasma Diagnostic Techniques*, R. H. Huddlestone and S. L. Leonard, Eds. (Academic, New York, 1965).

[8] `http://w3.pppl.gov/~krommes/fweb_toc.html` (FWEB home page)

[9] `http://www.unidata.ucar.edu/software/netcdf/` (netCDF home page)

[10] `http://www.hdfgroup.org/` (HDF home page)

[11] `https://wci.llnl.gov/codes/silo/index.html` (Silo home page)

[12] "The EIRENE Code User Manual" (D. Reiter, Forschungszentrum Jülich, 2017), EIRENE home page. information.

[13] K. M. Case and P. F. Zweifel, *Linear Transport Theory* (Addison-Wesley, Reading, MA, 1967).

[14] R. Marchand and M. Dumberry, *Comp. Phys. Comm.* **96**, 232 (1996).

[15] D. P. Stotler, A. Yu. Pigarov, C. F. F. Karney, S. I. Krasheninnikov, B. LaBombard, B. Lipschultz, G. M. McCracken, A. Niemczewski, J. A. Snipes, J. L. Terry, and R. A. Vesey, in *Fusion Energy 1996* (Proc. 16th Int. Conf. Montreal, 1996), Vol. 2 (IAEA, Vienna, 1997) p. 633 (see also the corresponding PPPL Report).

[16] R. K. Janev, W. D. Langer, K. Evans, Jr., and D. E. Post, Jr., *Elementary Processes in Hydrogen-Helium Plasmas* (Springer-Verlag, Berlin, 1987).

[17] D. Reiter, in *Atomic and Plasma-Material Interaction Processes in Controlled Thermonuclear Fusion*, R. K. Janev and H. W. Drawin, Eds., (Elsevier, New York, 1993), p. 243.

[18] R. K. Janev and J. J. Smith, *Atomic and Plasma-Material Interaction Data for Fusion* (Supplement to the journal Nuclear Fusion) **4** (1993).

[19] L. C. Johnson and E. Hinnov, *J. Quant. Spectrosc. Radiat. Transfer* **13**, 333 (1973).

[20] J. C. Weisheit, *J. Phys. B: Atom. Molec. Phys.* **8**, 2556 (1975).

[21] M. Goto, *J. Quant. Spectrosc. Radiat. Transfer* **76**, 331 (2003).

[22] T. Fujimoto, *J. Quant. Spectrosc. Radiat. Transfer* **21**, 439 (1979).

[23] R. J. Kanzleiter, D. P. Stotler, C. F. F. Karney, and D. Steiner *Phys. Plasmas* **7**, 5064 (2000).

[24] P. S. Krstic and D. R. Schultz, *Atomic and Plasma-Material Data for Fusion* **8**, 1 (1998).

[25] P. Bachmann and D. Reiter, *Contrib. Plasma Phys.* **35**, 45 (1995).

[26] H. H. Abou-Gabal and G. A. Emmert, *Nucl. Fusion* **31**, 407 (1991).

[27] A. Dalgarno, *Phil. Trans. R. Soc. London, Ser. A* **250**, 426 (1958).

[28] T. Holstein, *J. Phys. Chem.* **56**, 832 (1952).

[29] J. M. Wadehra, *Phys. Rev. A* **20**, 1859 (1979).

[30] C. F. Barnett, *Atomic Data for Fusion*, Oak Ridge National Laboratory Report ORNL-6086, Vol. 1 (1990).

[31] J. H. Newman, J. D. Cogan, D. L. Ziegler et al., *Phys. Rev. A* **25**, 2976 (1982).

[32] A. C. Reviere, *Nucl. Fusion* **11**, 363 (1971).

[33] J. F. O'Hanlon, *A User's Guide to Vacuum Technology* (John Wiley & Sons, New York, 1989).

[34] A. Niemczewski, Ph. D. Thesis, Massachusetts Institute of Technology, Plasma Fusion Center Report PFC/RR-95-8 (1995).

[35] P. L. Bhatnagar, E. P. Gross, and M. Krook, *Phys. Rev.* **94**, 511 (1954).

[36] P. Welander, *Arkiv Fysik* **7**, 507 (1954).

[37] Chr. May, Ph. D. Thesis.

[38] D. Reiter, Chr. May, M. Baelmans, and P. Börner, *J. Nucl. Mater.* **241**, 342 (1997).

[39] R. J. Kanzleiter, Ph. D. Thesis, Rensselaer Polytechnic Institute (1999).

[40] S. Chapman and T. G. Cowling, *The Mathematical Theory of Non-Uniform Gases* (Cambridge University Press, 1970).

[41] T. F. Morse, *Phys. Fluids* **6**, 1420 (1963).

[42] R. C. Reid, J. M. Prausnitz, B. E. Poling, *The Properties of Liquids and Gases* (McGraw-Hill Book Company, New York, 1987).

[43] *CRC Handbook of Chemistry and Physics, 75th Edition, 1913–1995*, D. R. Lide, Ed., p. 6-239 (CRC Press, Boca Raton, 1994).

[44] C. S. Chang, S. Klasky, J. Cummings et al., *J. Phys.: Conf. Ser.* **125**, 012040 (2008).

[45] P. S. Krstic and D. R. Schultz, *Phys. Rev. A* **60**, 2118 (1999).

[46] H. P. Summers, The ADAS User Manual, Version 2.6, 2002. http://www.adas.ac.uk/

[47] D. P. Stotler, D. J. Battaglia, R. Hager et al., *Nucl. Mater. Energy* **12**, 1130 (2017).

[48] D. R. Willis, *Phys. Fluids* **5**, 127 (1962).

[49] C. Cercignani, *J. Stat. Phys.* **1**, 297 (1969).

[50] C. Cercignani, *Theory and Application of the Boltzmann Equation* (Scottish Academic Press, New York, 1975).

[51] P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences* (McGraw-Hill Book Company, New York, 1969).

[52] W. Eckstein, *Computer Simulation of Ion-Solid Interactions* (Springer-Verlag, New York, 1991).

[53] G. Bateman, "Distribution of Neutrals Scattered Off A Wall", PPPL Applied Physics Division Report # 1 (1980).

[54] D. B. Macmillan, *SIAM J. Appl. Math.* **15**, 264 (1967).

[55] J. Spanier, *SIAM J. Appl. Math.* **14**, 702 (1966).

[56] S. Brandt, *Data Analysis: Statistical and Computational Methods for Scientists and Engineers* (Springer International Publishing, Switzerland, 2014), online.

[57] X-5 Monte Carlo Team, "MCNP - A General Monte Carlo N-Particle Transport Code, Version 5, Volume I: Overview and Theory", Los Alamos National Laboratory Report LA-UR-03-1987 (2008).

[58] D. E. Knuth, *The Art of Computer Programming* (Addison-Wesley, Reading, MA, 1997).

[59] C. S. Pitcher, C. J. Boswell, J. A. Goetz et al., *Phys. Plasmas* **7**, 1894 (2000).

[60] C. S. Pitcher, C. J. Boswell, T. Chung et al., *J. Nucl. Mater.* **290-293**, 812 (2001).

[61] D. P. Stotler, C. S. Pitcher, C. J. Boswell et al., *J. Nucl. Mater.* **290-293**, 967 (2001).

[62] D. P. Stotler, J. Boedo, B. LeBlanc, R. J. Maqueda, and S. J. Zweben, *J. Nucl. Mater.* **363-365**, 686 (2007).

[63] D. P. Stotler, D. A. D'Ippolito, B. LeBlanc, R. J. Maqueda, J. R. Myra, S. A. Sabbagh, and S. J. Zweben, *Contrib. Plasma Phys.* **44**, 294 (2004).

[64] M. C. Zarnstorff, L. A. Berry, A. Brooks et al., *Plasma Phys. Control. Fusion* **43**, A237 (2001).

Index

Contents