**Cost-optimized Automated Variance Reduction**
**for Highly Angle-dependent Radiation Transport Analyses**

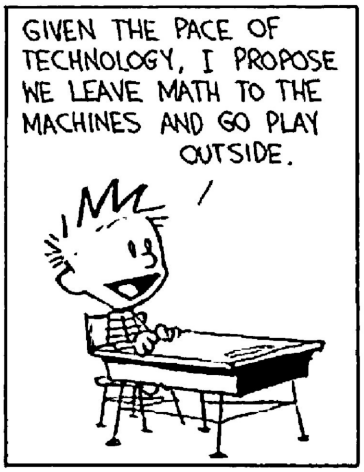
by

Joel A. Kulesza

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Nuclear Engineering and Radiological Sciences)
in the University of Michigan
2018

Doctoral Committee:

Assistant Professor Brian C. Kiedrowski, Co-Chair
Dr. Clell J. Solomon, Jr., Los Alamos National Laboratory, Co-Chair
Professor Edward W. Larsen
Professor William R. Martin
Professor Robert M. Ziff




*from Calvin and Hobbes, Nov. 25, 1992, by Bill Watterson*

Joel A. Kulesza

jkulesza@umich.edu

ORCID iD: 0000-0002-2669-6339

**Dedication**

For my parents, wife, and children—their love, encouragement, and support made this possible.

**Acknowledgements**

My deepest appreciation goes to my advisors, present and past: Prof. Brian Kiedrowski, Dr. CJ Solomon, and Prof. Ed Larsen. I truly appreciate their wisdom, thoughtfulness, and time as we have worked together. I am also indebted to my other committee members, Prof. Bill Martin and Prof. Bob Ziff, for their willingness to selflessly serve, their insightful questions, and the critical review they performed of this work.

My thanks to Mr. Roger Martz for introducing me to both Dr. Solomon and Los Alamos National Laboratory and for acting as an unwavering advocate on my behalf. My thanks to Prof. Kiedrowski and Dr. Solomon for the foresight and flexibility that allowed me to complete this work at Los Alamos National Laboratory. My thanks to Dr. Avneet Sood and Ms. Donna Pimentel for diligently making every accommodation I could hope for while I completed this work at Los Alamos National Laboratory. My thanks to Dr. Tim Burke, Dr. David Dixon, and Dr. Cory Ahrens for acting as sounding boards. I also appreciate Dr. Tom Booth's willingness to briefly suspend his retirement to meet and discuss this work.

I appreciate Ms. Peggy Jo Gramer's and Dr. Garnette Roberts's ceaseless work to ensure that all needs, academic and personal, were addressed during my time as a doctoral student.

My gratitude also goes to mentors from earlier in my career, Mr. Stan Anderson, Mr. Arnie Fero, and Dr. Gianluca Longoni, for instilling strong engineering principles and for encouraging me to follow a path that lead me to this point.

The following organizations provided support for this work and/or the accompanying degree, without which neither would have been possible to pursue: the University of Michigan, the U.S. Nuclear Regulatory Commission, the American Nuclear Society, ASTM International, the Consortium for Nonproliferation Enabling Capabilities, and Los Alamos National Laboratory.

In particular, the completion of this work was supported by the U.S. Department of Energy National Nuclear Security Administration (NNSA) Advanced Simulation and Computing (ASC) Program. It was also supported by the NNSA under Award Number(s) DE-NA0002576 and in part by the NNSA Office of Defense Nuclear Nonproliferation R&D through the Consortium for Nonproliferation Enabling Capabilities.

Finally, it would be unfair to not recognize my wife, Minyoung, for agreeing to take part in this adventure with me. 내게 힘이 되어 준 당신에게, 내 모든 사랑을 바칠 수 밖에.

## Preface

This dissertation, containing 94 figures and 36 tables, was assembled using LyX and typeset using LaTeX via LuaLaTeX with the Computer Modern type face. The figures were prepared using Python™ with matplotlib and/or TikZ. Either legacy or XML-based VTK files often stored the data used for plotting. Before producing the final renditions shown herein, most data files were interrogated using ParaView. References were managed with JabRef, processed using BibTeX with natbib, and are shown using the IEEEtranN style (slightly modified).

Countless hours were spent in Vim manipulating source code, input, and output. Much of the data processing was performed with NumPy and/or SciPy. Jupyter and *Mathematica*® notebooks were frequently used. The GNU and Intel® compilers were used with debugging/profiling performed with the TotalView® and ARM Forge (formerly Allinea Forge) tools.

This colophon and the accompanying software details are included to credit the indirect contributions to this work made by the community of developers and maintainers for these software products.

**Table of Contents**

**List of Figures**

**List of Tables**

## List of Appendices

**List of Acronyms and Terms**

| | |
|---|---|
| angular moment | A functional expansion in Legendre polynomials or spherical harmonic functions used to transition between discrete and continuous directional representations |
| COBYLA | Constrained optimization by linear approximation |
| contribution | A history makes a contribution to a tally every time it registers with a tally; a history can make multiple contributions |
| computational cost | An arbitrary measure of the effort required to perform Monte Carlo calculation by considering both the time taken and level of uncertainty in the result |
| COVRT | Cost-optimized Variance Reduction Technique, a discrete ordinates-based deterministic solver |
| FOM | Figure of merit |
| FTE | Future time equation |
| HSME(s) | History-score moment equation(s) |
| HSPDF(s) | History-score probability density function(s) |
| MADS | Mesh Adaptive Direct Search |
| MCMC | Markov-chain Monte Carlo |
| moment | The statistical moment corresponding to a particular order HSME solution, which represents the Monte Carlo tally score moments |
| NOMAD | Nonlinear optimization by mesh adaptive direct search |
| PDF | Probability density function |
| score | A history has associated with it a single score that can be used to calculate scoring moments; the score is the sum of the history's contributions |
| SLSQP | Sequential least squares quadratic programming |
| $S_N$ | Discrete ordinates |
| statistical moment | See moment |

## List of Algorithms

## List of Symbols

| | |
|---|---|
| $\mathcal{A}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}'$ | absorption kernel giving the probability of being absorbed in phase space $\boldsymbol{P}$ and transitioning to phase space $\boldsymbol{P}'$. For this work, absorption is considered terminal so there is no transition. |
| $\alpha_{\mathrm{IO}}$ | DXTRAN inner sphere importance relative to outer sphere |
| $\mathcal{B}_{\mathrm{DX}}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}'$ | DXTRAN transport kernel |
| $\beta(\boldsymbol{x})$ | Analyst-defined position-dependent DXTRAN rouletting parameter, defined continuously |
| $\beta_c$ | Analyst-defined DXTRAN rouletting parameter, defined for each Monte Carlo geometry cell $c$, defined constantly within each cell |
| $D(\boldsymbol{R})$ | Detector response function over phase space |
| $D_{\mathrm{I}}$ | Integrated detector response function—the quantity of interest |
| $\eta$ | Direction cosine |
| $E$ | particle kinetic energy |
| $\mathcal{E}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}'$ | emergence kernel giving the probability of emerging in phase space $\boldsymbol{P}'$ as a result of having survived a scattering collision in initial phase space $\boldsymbol{P}$. For this work, emergence applies only to scattering but it could also be used to model fission emergence with different constituent probability density functions in direction and energy. |
| $\Gamma_{\mathrm{DX}}$ | set of points interior to a DXTRAN region |
| $\delta\Gamma_{\mathrm{DX}}$ | set of points on the outer boundary of a DXTRAN region |
| $\mathcal{K}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}'$ | collision kernel giving the probability of colliding in phase space $\boldsymbol{P}$, surviving, and transitioning to phase space $\boldsymbol{P}'$. See also emergence kernel, $\mathcal{E}$. |
| $\lambda(\boldsymbol{p}, \boldsymbol{p}')$ | Optical distance between points $\boldsymbol{p}$ and $\boldsymbol{p}'$ |
| $\mu$ | Sample mean or direction cosine, depending on context |
| $\mu_{\mathrm{MC}}$ | Monte Carlo-calculated sample mean |
| $\widehat{\mu}$ | Deterministically calculated sample mean |
| $M_m$ | Tally scoring moment, $m^{\mathrm{th}}$ moment |
| $\widehat{M_m}$ | Deterministically calculated tally scoring moment, $m^{\mathrm{th}}$ moment |
| $\boldsymbol{n}$ | outward-directed unit normal vector for a given point on the surface, $\partial V$, of the system |

| | |
|---|---|
| $\omega_n$ | Quadrature weight corresponding to direction with cosines $\mu_n, \eta_n, \xi_n$ |
| $\boldsymbol{\Omega}$ | particle direction-of-flight |
| $\boldsymbol{\Omega}_{\mathrm{DX}}$ | Initial DXTRAN particle direction of flight |
| $\boldsymbol{P}$ | weight-augmented phase space under consideration within the current context, most generally $(\boldsymbol{x}, \boldsymbol{\Omega}, E, t, w)$, but for this work: $(\boldsymbol{x}, \boldsymbol{\Omega}, E, w)$ |
| $p_{\mathrm{ff}}(\boldsymbol{p}, \boldsymbol{p}')$ | Probability of straight-line free flight between points $\boldsymbol{p}$ and $\boldsymbol{p}'$ |
| $\boldsymbol{p}$ | DXTRAN initiation point |
| $\boldsymbol{p}_{\mathrm{c}}$ | Spherical DXTRAN region center point |
| $\boldsymbol{p}_{\mathrm{DX}}$ | DXTRAN particle creation point on the surface of the DXTRAN region |
| $\boldsymbol{p}_c$ | Point on DXTRAN region surface closest to $\boldsymbol{p}$ along a straight-line trajectory |
| $p(\mu)$ | Analog scattering (or source) emergence PDF |
| $\widetilde{p}(\mu)$ | Non-analog DXTRAN scattering emergence PDF |
| $\Phi_m$ | Angular moment of the $m^{\mathrm{th}}$ statistical moment, used to calculate the scattering source |
| $\psi(\boldsymbol{P}, s)$ | History-score probability density function |
| $\Psi_m(\boldsymbol{P})$ | History-score moment equation, $m^{\mathrm{th}}$ moment |
| $\widehat{\Psi}_m(\boldsymbol{P})$ | Deterministically calculated history-score moment equation, $m^{\mathrm{th}}$ moment |
| $q(\boldsymbol{R})$ | Combined source term consisting of the scattering and inhomogeneous sources |
| $Q(\boldsymbol{R})$ | optional inhomogeneous source within the system under consideration |
| $\boldsymbol{R}$ | traditional phase space under consideration within the current context, most generally: $(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)$ |
| $R_{\mathrm{MC}}$ | Relative standard fractional uncertainty in the sample mean |
| $\mathcal{S}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}'$ | Surface-crossing kernel giving the probability of emerging in phase space $\boldsymbol{P}'$ as a result of initial phase space $\boldsymbol{P}$ given that a surface was crossed at $\boldsymbol{P}$ |
| $\sigma^2$ | Sample variance |
| $\sigma^2_{\mathrm{MC}}$ | Monte Carlo-calculated sample variance |
| $\widehat{\sigma}^2$ | Deterministically calculated sample variance |
| $\sigma_{\mu,\mathrm{MC}}$ | Monte Carlo-calculated standard deviation of the sample mean |
| $\Sigma_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E' \to E)$ | double-differential macroscopic scattering cross section |
| $\Sigma_{\mathrm{t}}(\boldsymbol{x}, E)$ | macroscopic total cross section |
| $\mathcal{T}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}w'$ | transmission kernel giving the probability of free-flight transition from one spatial position to the next without undergoing collision, i.e., the free- flight probability from $\boldsymbol{x}$ to $\boldsymbol{x}'$ |

| | |
|---|---|
| $\mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}w'$ | Free-flight transport kernel with DXTRAN truncation |
| $v$ | speed of a particle corresponding to its kinetic energy through the non-relativistic relationship $v = \sqrt{2E/m}$ where $m$ is the particle's mass |
| $V$ | interior volume of the convex system under consideration |
| $\partial V$ | two-sided surface of the convex system under consideration |
| $w$ | Computational weight of a particle, unity for strictly analog calculations, used to preserve fairness of non-analog games |
| $w_{\mathrm{DX}}$ | Initial computational weight of a DXTRAN particle |
| $\boldsymbol{x}$ | particle position within the spatial domain |
| $\xi$ | Direction cosine |
| $\zeta$ | Uniformly sampled random number between zero and one |

**Abstract**

Monte Carlo variance-reduction techniques that directly bias particle direction have historically received limited attention despite being useful for many radiation transport applications such as those in which radiation travels through large regions with a low probability of colliding. One such technique is known as DXTRAN (short for deterministic transport) in the MCNP Monte Carlo radiation transport code. Until now, effectively applying DXTRAN in calculations is largely based on empirical observations of computational performance. Optimal DXTRAN parameters are identified through manual iteration. This work develops new mathematical descriptions of the DXTRAN variance-reduction process and demonstrates a new automated variance-reduction method that applies these mathematical formulations to determine the optimal application of DXTRAN in a given problem.

Specifically, this work includes the first known deduction and application of the integral transport kernels for both biasing with DXTRAN particle production and the associated free-flight transport with truncation of the initiating particle. This work applies these new DXTRAN transport kernels to derive expressions for the mean tally response in Monte Carlo transport calculations involving DXTRAN. These expressions are then used to rigorously prove, for the first time, that the technique is unbiased. This work also derives equations for the variance and associated computational cost of Monte Carlo calculations involving DXTRAN, which are solved using the deterministic discrete ordinates method.

To verify the derivations developed in this work, fourteen 1-D and seven 2-D test case calculations are made. Within the 2-D test cases, a variety of scenarios are examined that lead to highly angle dependent solutions where other variance-reduction techniques that do not directly bias particle direction are challenged. For the 1-D cases, when DXTRAN is solely used, the Monte Carlo and deterministically calculated mean and variances agree within 1.4%. For the 2-D cases, the agreement is generally well within 10% and never worse than 13%, which is consistent with prior analyses for other variance-reduction techniques.

Of the verification test cases, six 1-D and six 2-D test cases are processed using an automated optimization workflow to determine optimal DXTRAN variance reduction parameters. As long as a non-trivial change in FOM is predicted by the optimizer, the optimizer identifies improved DXTRAN parameters relative to the initial guess in all but one case. For the 2-D test cases, a coarse angular quadrature is used to permit the optimization iterations to run quickly; however, the relative change in computational cost as a result of varying DXTRAN size, position, and rouletting parameters is adequately captured.

This work provides a method that could augment a strictly variance-reducing hybrid radiation transport method (e.g., FW-CADIS) to improve the efficiency of highly angle-dependent radiation transport analyses.

# Chapter 1

# Background & Introduction

The goal of the work within this dissertation is to automatically generate optimized Monte Carlo variance-reduction parameters for radiation transport analyses with a strong directional dependence, for which existing approaches are deficient. Deterministic transport methods are used in conjunction with optimization techniques to inform the usage of a particular variance-reduction technique called DXTRAN (short for deterministic transport), which directly permits angular biasing. This dissertation develops a theoretical framework to predict the Monte Carlo statistical variance and computational cost associated with the DXTRAN technique and describes the associated algorithms implemented within a deterministic radiation transport solver. This new method is demonstrated for a variety of 1-D and 2-D test cases.

This chapter provides: motivation for the work, enumerating some of the applications and challenges for problems with strong directional dependence; a discussion of the relevant computational radiation methods; the current state of hybrid radiation transport techniques; the current state of techniques to deterministically predict Monte Carlo variance and calculation time; a development of the DXTRAN technique; and a summary of the novel components of this work.

## 1.1 Motivation

The focus of this work is on automatically generating optimized Monte Carlo variance-reduction parameters for highly angle-dependent radiation transport analyses. Radiation transport analyses can be characterized by various transport characteristics such as being attenuation/scattering dominated or streaming dominated. Attenuation/scattering-dominated analyses typically feature large spatial regions containing material that is optically thick, which leads to small mean free paths between particle collisions. Streaming-dominated problems feature large spatial regions containing material that is optically thin leading to large mean free paths between particle collisions. These two characteristics are often found together where there are a combination of attenuating/scattering and streaming regions. In these situations, it can be challenging to calculate the particle population at a region of interest, for example a radiation detector, as a result of some distant radioactive source. This is because particles can either (a) pass through the attenuating/scattering material with low probability or (b) bypass the attenuating/scattering material in free flight though the streaming regions with low probability. Those particles that travel through the streaming region(s), in free flight or with just a few collisions, will tend to be more energetic than those particles that pass through the attenuating/scattering media. Furthermore, the direction of travel for particles in the streaming regions can be highly anisotropic, traveling along the dominant direction of the streaming region. Such analyses are

considered highly angle-dependent because of this anisotropy along a preferred, but low-probability, flight path.

Examples of such highly angle-dependent problems can be found on several scales. On a small scale, a radioactive source and detector are positioned in adjacent rooms separated by a shield but with some penetrations through the shield. Radiation from the source might reach the detector by penetrating through the shield or bypassing the shield through the penetration, either way with low-probability. On a larger scale, radioactive material may be found in an urban environment that also has radiation detectors that are meant to detect the presence of the material. In this scenario, the buildings act as shields and the streets and alleyways between buildings act as streaming paths. On a larger scale one might consider the case of calculating radiation dose to satellites where planetary features act as shields. In all cases, particle collisions with nearby structures (i.e., walls, buildings, or planets) can cause particles to change direction toward the region of interest (i.e., the detector or satellite). However, the probability of particles changing direction toward the region of interest is low. This low probability is largely driven by the small solid angle of the target region as viewed from the collision point. Furthermore, the probability of having a collision in the streaming region and emerging in a direction toward the region of interest is low. Because of these low probabilities, these analyses can be considered highly angle dependent.

Regardless of the spatial scale of the scenario of interest, there are several methods and tools available to predict the behavior of radiation within them. Two common methods to analyze these scenarios are described next.

### 1.2 Radiation Transport Methods

The two major computational methods to perform radiation transport calculations are stochastic or deterministic. Stochastic methods used in Monte Carlo radiation transport computer codes tend to be the preferred approach to analyze highly angle-dependent radiation transport scenarios because such codes can precisely and continuously represent the stochastic behavior of computational particles without implicit discretization error. However, Monte Carlo calculation results include statistical uncertainty by virtue of using random sampling of the computational particle's phase-space evolution from creation to destruction. Alternatively, deterministic methods are available that solve radiation transport equations over discretized elements of phase space. Such deterministic methods include the point kernel approach, the method of characteristics, and the discrete ordinates ($S_N$) method. This work focuses solely on the $S_N$ method because it is the most widely used deterministic method applied to radiation shielding scenarios that contain significant scattering. Details on Monte Carlo and $S_N$ radiation transport methods are described next in greater detail.

#### 1.2.1 Monte Carlo

Typically, the Monte Carlo method follows computational particles through a random walk from creation (e.g., in the radioactive material acting as a radioactive source) to destruction (e.g., by leaking from the system or getting absorbed). The behavior of the particles during the random walk is governed by randomly sampling probability density functions (PDFs) that govern phase-space state transitions and the random walk forms a Markov chain where the future state depends only on the current state. If a computational particle interacts with a region of interest during its random walk, its contribution to the region of interest is counted

(tallied). An example of a region of interest is a radiation detector active volume. The physical detector has some response to physical particles entering and interacting within it. This response is modeled by the Monte Carlo calculation as a tally such that computational particles that interact with the tally contribute score to the tally consistent with how a physical particle would interact in the physical detector volume. At the conclusion of the calculation, the statistical properties of the tally are assessed to provide an average (mean) response along with the associated statistical uncertainties calculated from the statistical moments of the tally score distribution that arise from the inherent randomness of the method.

#### 1.2.1.1 Analog Monte Carlo

Analog Monte Carlo calculations are those that use PDFs that represent particle behavior consistent with the laws of nature (as best as they are understood and representable). In analog calculations, the only method to reduce tally statistical uncertainty is to increase the number of computational particles (statistical samples) used in the calculation. However, if the computational particles have a low probability of reaching the tally, then the calculation will require a large number of computational particles to characterize the resulting behavior with an acceptably low uncertainty. This is often computationally intractable. An example of such a scenario is that of a well-designed radiation shield separating radioactive material and a detector. The shield, by design, prevents particles from reaching the detector so it will take many computational particles to simulate the behavior of physical particles with an acceptably low statistical uncertainty.

#### 1.2.1.2 Non-analog Monte Carlo

Another method to reduce tally statistical uncertainty is to bias the statistical sampling processes used during the random walk such that the mean tally response is preserved but higher statistical moments are modified. Such biasing results in non-analog transport and is implemented through variance-reduction techniques (often referred to as variance-reduction "games"). Variance-reduction techniques modify the PDFs associated with various phase-space state-change operations in a fair way such that, on average, the statistical sampling performed by each computational particle does not result in a change in the mean tally response but the uncertainty and/or the calculation time needed for a fixed uncertainty level is reduced. Fairness is preserved by augmenting the computational particle with a statistical weight to compensate for modified sampling as a result of variance reduction. Statistical weight is discussed further in Section 2.4.4.

In general, variance-reduction techniques can only reduce the tally uncertainty or the computational time in a mutually exclusive manner. Reducing the variance results in a longer computational time and vice versa. Because of this balance, variance-reduction parameters are usually assigned to variance-reduction techniques to minimize the product of the uncertainty and associated computational time rather than only one quantity or the other. This minimized product can be referred to as a computational cost and can act as a measure of the Monte Carlo calculation's inverse performance (low cost corresponds to high efficiency) relative to calculations using the same computer software and hardware that are identical except with different variance-reduction parameters.

Historically, a radiation analyst interested in minimizing the computational cost of a Monte Carlo radiation transport analysis can select from one or more variance-reduction techniques and can then select parameters to tune the selected techniques. However, the ability to achieve a minimized computational cost is limited

by the analyst's skill, experience, and time available to explore the variance reduction parameter space. Furthermore, the criteria used to minimize the Monte Carlo computational cost are generally subjective and based on heuristics. As the complexity of the Monte Carlo calculation increases, it can be impractical for an analyst to fully explore the available combinations of variance-reduction techniques and tuning parameters to effectively reduce the computational cost.

### 1.2.2 Discrete Ordinates ($S_N$)

As an alternative to Monte Carlo methods for radiation shielding analyses, the $S_N$ method discretizes the angular domain by dividing the directions of particle travel on the unit sphere of phase space into a finite number of discrete directions with associated quadrature weights. The $S_N$ method is most commonly applied with a discretized spatial mesh and using the multi-group approximation to discretize the energy domain. As such, it can be used to globally characterize particle behavior. The $S_N$ method and accompanying discretization are often limited by the level of spatial, angular, and energy mesh refinement versus available computer resources.

Discrete ordinates methods can perform well for deep-penetration shielding problems with attenuating material that induces significant particle scattering. This is because particle attenuation through a material is treated in an approximate but deterministic manner. However, $S_N$ methods can have difficulty with streaming- or absorption-dominated problems. In such problems, $S_N$ methods often exhibit non-physical "ray effects." Ray effects are artificial peaks and valleys in the solution because particles are only allowed to travel along discrete directions of flight. Generally, ray effects are overcome by wisely selecting an appropriate angular quadrature and/or using a large number of directions (using a fine angular quadrature). For deep penetration problems, ray effects are ameliorated by "smoothing" introduced by collisions that reduces the variation between the ray-effect peaks and valleys. However, streaming regions cannot capitalize on this behavior because there are far fewer collisions that implicitly perform smoothing. Furthermore, if a streaming path is not directed along one of the quadrature directions, then the effect of streaming through that path may not be accurately represented. Thus, $S_N$ methods tend to be challenged in problems with strong directional dependence.

## 1.3 Variance-reducing Hybrid Radiation Transport Methods

As discussed in Section 1.2, both Monte Carlo and discrete ordinates methods have drawbacks and strengths. Monte Carlo calculations can take an intractable number of computational particles and/or an unreasonable amount of analyst effort to achieve statistical convergence. Discrete ordinates calculations may require an intractably fine space, angle, and/or energy mesh to mitigate discretization error. However, increasingly the two methods are being combined (often referred to as hybrid radiation transport methods) in ways that attempt to capitalize on the strengths of both methods while minimizing the drawbacks of each.

In particular, deterministic calculations are increasingly used to generate problem-wide variance-reduction parameters for Monte Carlo calculations. Most commonly, this has been done using discrete ordinates calculations in various ways to define boundaries for the Monte Carlo "weight window" variance-reduction technique (with additional information on weight windows available in the MCNP manual [1]). These hybrid transport approaches seek to minimize the Monte Carlo statistical uncertainty but do not directly consider the associated computational cost. Alternatively, methods have been developed to directly bias Monte

Carlo particle direction but those methods are limited to either discrete Monte Carlo particle directions or multi-group Monte Carlo calculations. A review of prior work in these areas of hybrid radiation transport follows.

### 1.3.1 Burgart and Stevens

In 1970, Burgart and Stevens described a method to perform angle-energy biasing for Monte Carlo calculations by modifying the transmission and collision kernels [2]. The transmission kernel is modified using a direction-dependent exponential transform. The collision kernel is modified by discretizing the Monte Carlo directions of flight into an $S_N$-like grid and computing the importance of each direction. A total of 30 directions are used with an importance function calculated using a 1-D adjoint $S_N$ calculation.

The method is then compared to a prior analysis by Straker [3], a hypothetical infinite medium composed of an infinite-mass isotropic scatterer, and the SNAP-2 LiH shield experiment [4]. Generally Burgart and Stevens's method performed better than those it was compared against; however, in some cases ray effects became apparent. Pointing to the specificity of the method, Burgart and Stevens concluded by noting that "much work in the area of data handling, fitting, and synthesizing of the importance function remains to be done, particularly for cylindrical and spherical geometries."

### 1.3.2 Schmidt

Later in 1971, Schmidt described a method to perform angle-energy biasing for 1-D planar Monte Carlo deep penetration calculations using a semi-analytic method [5]. The Monte Carlo calculations require particles to travel only along discrete directions. The method constructs an analytic expression for a planar 1-D space-, energy-, and angle-dependent importance function from several parameterized exponential functions. The parameters for the exponential functions are calculated using 1-D $S_N$ adjoint calculations. While constructing the importance function several simplifying assumptions and arithmetic approximations are made including (1) assuming at most linear anisotropic scattering, (2) making an arbitrary simplification of the importance function to a less complicated form, and (3) simplifying an exponential distribution by an approximate algebraic expression.

The resulting importance function is then used to modify the collision process and perform the corresponding weight correction. Schmidt then demonstrated the method compared to several other Monte Carlo calculations that analyze the SNAP-2 experiment [4]. In general, Schmidt's method produced a lower variance than the other Monte Carlo calculations it was compared to. However, Schmidt did not attempt to find parameters that minimize variance and/or computational cost. Moreover, many of the simplifying assumptions and arithmetic substitutions made throughout Schmidt's work do not appear to be systematic.

### 1.3.3 Tang, Hoffman, and Stevens

In 1977, Tang, Hoffman, and Stevens described a method to perform angle biasing in 3-D multi-group Monte Carlo calculations [6] using importance parameters determined from adjoint 2-D $S_N$ calculations. This work follows from a 1976 technical report by the same authors [7]. In both documents the authors perform a cylindrical straight-duct streaming analysis through a concrete shield using various combinations of source, importance-splitting and rouletting, exponential-transform, and collision-emergence biasing. This work is

similar to the earlier work by Burgart and Stevens [2] in that it also requires the Monte Carlo particles to travel along discrete directions.

### 1.3.4 Vergnaud, Nimal, et al.

Beginning in 1985, the ability to perform space-, angle-, and energy-biasing in the TRIPOLI Monte Carlo code was described by Vergnaud, Nimal, and others [8–13]. The method uses deterministic collision-probability calculations to determine biasing parameters. The biasing methods described require multi-group Monte Carlo calculations and were demonstrated for a variety of 1-, 2-, and 3-D geometries.

### 1.3.5 Abotel, Larsen, and Martin & Baker and Larsen

In 1991–1993, Abotel, Larsen, and Martin [14–16] and Baker and Larsen [17] made several publications on a local exponential transform technique applied to multi-group Monte Carlo calculations. The method generates local biasing parameters with an exponential form to represent the local importance. The method permits directly biasing in angle through the collision emergence kernel by permitting angle-dependent local biasing functions.

The work by Abotel, Larsen, and Martin used 2-D adjoint diffusion calculations to generate biasing parameters that act to reduce variance local to a region of interest. The method is then demonstrated for several 1-D and 2-D test cases with one or two energy groups and either isotropic or linear anisotropic scattering behavior. The work by Baker and Larsen instead generated biasing parameters from a low-order solution to the forward transport equation. In this way, the maximum variance in the Monte Carlo calculation is reduced. It is also demonstrated for several 2-D multi-group calculations.

### 1.3.6 Van Riper et al.

In 1997, Van Riper and others published a report on the Automatic Variance and Time of Analysis Reduction (AVATAR) application to perform automatic variance reduction in Monte Carlo calculations [18]. AVATAR performs an adjoint 3-D discrete ordinates calculation on a spatial mesh independent of the Monte Carlo geometry and identifies Monte Carlo weight window boundaries based on the behavior of the adjoint flux. AVATAR can produce either scalar or angle-dependent weight windows. Scalar weight windows are calculated by inverting the scalar adjoint flux solution and applying an arbitrary normalization. The angle-dependent weight windows use the scalar adjoint flux to approximate an angular adjoint flux by applying information theory to control the population of particles with respect to direction of travel. However, this approach cannot bias particles into those directions of travel directly. Despite its name, AVATAR does not explicitly or directly address the time of the Monte Carlo calculation.

### 1.3.7 Turner and Larsen

Also in 1997, Turner and Larsen developed the Local Importance Function Transform (LIFT) method [19–21]. This method approximates a zero variance radiation transport solution by a transformed transport problem that contains local biasing parameters obtained from diffusion, simplified spherical harmonics ($P_N$), and/or $S_N$ deterministic calculations. The work biases distance to collision and post-collision emergence direction and energy. As such, the method can directly bias particle direction. However, it was only shown to work for multi-group Monte Carlo calculations. In addition, this work acknowledged difficulty in the presence of voids

and low-density regions, characteristic of angle-dependent scenarios, which cause a reduction in the efficiency of the LIFT method relative to results from other methods methods such as AVATAR. Note that this work follows from the work by Abotel, Larsen, and Martin & Baker and Larsen [Section 1.3.5].

### 1.3.8 Wagner and Haghighat

In 1998, Wagner and Haghighat developed the Consistent Adjoint-driven Importance Sampling (CADIS) method [22, 23]. The CADIS method operates roughly similar to AVATAR; however, it produces scalar weight window boundaries *and* generates Monte Carlo forward source biasing parameters consistent with the weight window boundaries. It does this by first solving the adjoint particle transport equation using the Monte Carlo tally as the adjoint source. CADIS then computes the response of interest by calculating the inner product of the adjoint solution and the forward source. The weight window boundaries are then set such that the center of the weight window is the response normalized by the scalar adjoint transport solution. In concert, a biased forward source is defined as the adjoint-weighted forward source normalized by the response. This treatment ensures that the biased source and weight windows are consistent.

As a result of this process, CADIS attempts to consistently populate phase space important to a Monte Carlo tally at two levels: by way of source sampling and weight window population controls. However, CADIS has no capability to directly bias particle direction because it uses only scalar adjoint quantities. Instead, it must rely on analog collision emergence to serendipitously direct particles in important directions.

### 1.3.9 Cooper and Larsen

In 2001, Cooper and Larsen described a method for global Monte Carlo particle transport problems that attempts to distribute particles uniformly throughout the system [24] (based on Cooper's Ph.D. research in [25]) to achieve uniform statistical uncertainty. A unique element to this work is the use of a forward deterministic transport solution to define weight window boundaries. In addition, recognizing that $S_N$ and $P_N$ transport solutions are computationally expensive, this work uses quasi-diffusion deterministic calculations (which use a transport-correction term to close the $P_1$ equations). However, this work only considered isotropic scattering for mono-energetic analyses. Because scalar weight windows are used, this work cannot be used to directly perform direction biasing in the resulting Monte Carlo calculation.

### 1.3.10 Becker, Wollaber, and Larsen

In 2007, similar work to Cooper and Larsen's is described by Becker, Wollaber, and Larsen [26]. As with the work by Cooper and Larsen, this method is meant for calculations in which particle population behavior is of interest at all physical locations in the system. This work also followed from work by Wollaber and Larsen [27] and formed the foundation of Becker's Ph.D. dissertation later in 2009 [28]. Ultimately, this work provides a method that allows direct angle biasing by transforming the scattering emergence operator like Turner and Larsen's, but it is also only applied to multi-group Monte Carlo analyses.

### 1.3.11 Wagner, Blakeman, and Peplow

Also in 2007, Wagner, Blakeman, and Peplow extended the CADIS method developed by Wagner and Haghighat to include forward weighting (designated as FW-CADIS) [29]. The FW-CADIS method is described later in more detail and compared with CADIS in [30]. CADIS is intended for Monte Carlo

calculations with a single tally region, but FW-CADIS is meant for multiple tally regions, up to and including mesh-based track-length tallies that encompass the full problem geometry. Thus, FW-CADIS can be used as a global variance-reduction scheme.

The objective of FW-CADIS is to have uniform reduction in statistical uncertainty for all Monte Carlo tallies. FW-CADIS accomplishes this by first solving the forward radiation transport equation and computing the scalar response of the system. The adjoint transport equation is then solved with all Monte Carlo tallies of interest used to define the adjoint source weighted by the forward-calculated scalar response. From this point, the FW-CADIS method behaves the same as CADIS. Like CADIS, FW-CADIS uses scalar weight windows and cannot bias particle direction directly.

#### 1.3.12 Munk and Slaybaugh

In 2016 and 2017, Munk and Slaybaugh reported on a modification to CADIS and FW-CADIS methods designated as CADIS-Ω and FW/CADIS-Ω, respectively [31, 32]. These new methods use contributon flux information to construct weight windows best suited for deep penetration problems. This work modifies CADIS and FW-CADIS by computing the scalar contributon flux (the angle-integrated product of the forward and adjoint flux solutions) normalized by the scalar forward flux. The normalized contributon flux is used in place of the adjoint flux to then follow the steps necessary for CADIS and FW-CADIS. As such, this approach attempts to capture spatial and directional importance information using the scalar contributon flux. This work performed best in strongly anisotropic scenarios with preferential particle flow paths but only if the flow paths are not comprised of air. That is, anisotropy is handled efficiently but significant collisions are necessary to populate important directions (as with CADIS and FW-CADIS).

### 1.4 Computational-cost Optimizing Hybrid Radiation Transport Methods

Discrete ordinates radiation transport calculations have also been used to solve for Monte Carlo statistical moments and computational time directly using the history-score moment equations (HSMEs) and the future-time equation (FTE), respectively. Using deterministic methods to obtain the solutions of these equations to develop optimal Monte Carlo variance-reduction parameters forms a hybrid radiation transport method, which is central to this dissertation.

The HSMEs are the statistical moments of the history-score probability density function (HSPDF), which gives the probability of a history contributing partial scores to a tally throughout phase space. As such, the HSMEs can be used to predict the Monte Carlo tally first statistical scoring moment (the expected score) and the second scoring moment (used with the first moment to find the population variance). Higher scoring moments can be calculated, but this has not been done in practice. Similarly, the future-time probability density function (FTPDF) gives the probability of a history contributing partial computational time throughout phase space. The FTE is the first moment of the FTPDF and gives the expected time contributed to the calculation by a history as a function of phase space.

When variance reduction is used, the solutions to the second-moment HSMEs and the FTE change as a result (the first moment, the mean value, is preserved). Thus, the effect of a set of variance-reduction parameters on the Monte Carlo tally uncertainty and computational time can be predicted. As described in Section 1.2.1.2,

these two quantities can be used to characterize the Monte Carlo computational cost. If the variance-reduction parameters are varied the change in the computational cost can be found. Accordingly, the variance-reduction parameters can be optimized to find the set that produces a minimized Monte Carlo computational cost.

In some form, the HSME and FTE-based methods have been studied since the 1960s. However, the author is not aware of any software tools that are used routinely to solve the HSMEs and/or the FTE. The HSMEs and the FTE are not typically solved because doing so is computationally expensive and can include subtleties not typical in traditional forward or adjoint radiation transport equation solvers. Retrofitting existing radiation transport codes to solve the HSMEs and the FTE can (a) be a non-trivial effort and (b) lead to calculation times that are unacceptably long. It is for these two reasons that the author suspects the prevalence of HSME and FTE-based methods is not greater despite the apparent benefit of being able to predict Monte Carlo computational cost directly.

A review of prior work in this area of hybrid radiation transport follows.

### 1.4.1 Coveyou, Cain, and Yost

One of the earliest publications on the solution of the integral adjoint equation to inform Monte Carlo importance function biasing is given by Coveyou, Cain, and Yost in 1967 [33]. In this paper, the authors focus on radiation shielding analyses with non-multiplying media. Furthermore, particle splitting is not considered. The authors go on to develop an integral history-score probability function (potential value function in their parlance). In this formulation, the authors present the transport kernel notation used since: for kernel $K(x, y)\,\mathrm{d}y$, if $x$ then $y$ with probability density $K(x, y)\,\mathrm{d}y$.

With these tools, the authors present a framework to deterministically predict the *a priori* Monte Carlo population variance. The authors continue by demonstrating that biased sampling (with appropriate compensatory weight modification) does not modify the first moment solution but will have an effect on the second moment. However, the authors claim that it is impossible to realistically assess the effect of the biased sampling on the overall sampling procedure. This point will be returned to in Sec. 1.4.11. Nevertheless, the authors conclude with derivations for population variance in source biasing, survival biasing, and importance function biasing and a sample analysis to demonstrate the validity of the work for analog, next-, and last-event estimators.

### 1.4.2 Amster and Djomehri

Later in 1967, Amster and Djomehri provides a direct linkage between the solution of the first moment equation to the adjoint solution of the transport equation [34]. The objective of this work was to present a method for determining, in advance, whether a proposed Monte Carlo method using a Markov process would be "worthwhile" from a statistical variance standpoint. The authors begin with a derivation of integral transport equations that are not necessarily adjoint to the forward transport equation. These are shown to be equivalent to integro-differential equations that are adjoint to the forward transport equation. This derivation establishes the nomenclature for much of the later work in this area. The derivation of the adjoint equations then leads to a three-step analysis process to evaluate the first and second moments for a given Monte Carlo method:

1. Form the integro-differential transport equation of interest,
2. Determine its adjoint formulation,
3. Specify an appropriate adjoint source using the history-score-moment equations and solve for, in order, the first statistical moment and the second statistical moment.

The authors then conclude with an application of the process using one-group diffusion theory in an infinite, homogeneous, non-multiplying, isotropically scattering medium with a distributed source for collision estimators. Note that the HSME nomenclature developed in this work ($M_1$ and $M_2$) is not used for this work because of the desire to distinguish between the HSMEs themselves and the inner product of the HSME with the forward source.

### 1.4.3 Booth and Amster

The latter-half of 1977 and 1978 saw significant work in the area of deterministically predicting the statistical error in Monte Carlo calculations. Booth and Amster submitted the first of three papers [35] virtually in parallel on this area. This work generalized the collision-estimator theory in [34] to treat two successive collisions allowing one to determine the statistical error for a track-length estimator. Because the estimator is no longer point-wise, the authors have introduced another quantity commonly used in later work, including this work, for transmission (i.e., non-collision) probability for a particle to move from one element of phase space to another. Next, the authors present a comparison of variance between collision and track-length estimators, the development of deterministically calculated expected track-length values, and conclude with an example.

### 1.4.4 Everett and Cashwell

In 1978, Everett and Cashwell address the computational cost of variance reduction in [36]. This work uses the simple transport problem of estimating transmission through a planar slab. First, analytic expressions for cost are developed using three methods:

1. No splitting,
2. Splitting each source particle into two or more particles with appropriate weight correction, and
3. Splitting each collided particle in the slab into two or more particles with appropriate weight correction.

The resulting expressions show that splitting is only sometimes justified (e.g., when the thickness of the slab is large relative to the amount of times a particle is split at each collision). Further, this work calculates the optimal position to perform particle splitting and the number of times to split a particle. This is the first work that considered the computational expense of variance reduction versus the expected reduction in variance.

### 1.4.5 Lux

Meanwhile in 1978, Lux independently proved some general theorems regarding Monte Carlo variance behavior [37]. In [37], Lux incorporates particle weight into the derivations so that non-analog Monte Carlo behavior can be considered. He then goes on to analyze three variance-reduction methods (not described further because they are not relevant to the current work) to demonstrate the usefulness of the foregoing derivations. Lux also published [38] as a response to [34, 35] to provide a generalization of the equations derived therein. Note that Lux also describes HSME-based methods extensively in his book with Koblinger [39, Chapter 5].

### 1.4.6 Sarkar and Prasad

In 1979, Sarkar and Prasad (separately but in parallel with Booth and Cashwell) derived integral equations in [40] to determine the expected statistical error in non-analog Monte Carlo calculations. In addition, generalizations to accommodate multiplying systems were made. They went on to analyze exponential transform by deriving the first and second moments and comparing the integral equation results to Monte Carlo calculations. As a result of this analysis, they showed that the biasing parameter that maximizes efficiency may not necessarily coincide with the lowest variance. Note that Booth also addressed the exponential transform some years later for tri-directional travel ($\mu = 0, \pm 1$) in [41].

### 1.4.7 Booth and Cashwell

Also in 1979, Booth and Cashwell (separately but in parallel with Sarkar and Prasad) derived integral equations in [42] to calculate efficiencies of particle transport problems. This work also treats multiplying (even supercritical) systems and time-dependent analyses. This work concludes by comparing collision and track-length estimators, with and without implicit capture, for calculating the expected number of collisions a particle and its daughters make in time interval 0 to $t$. Like Sarkar and Prasad, this work showed that there is a balance between reduced variance and increased computational time required to maximize the overall efficiency of a biased Monte Carlo calculation.

### 1.4.8 Juzaitis

In 1982, additional work was published by Juzaitis (following from his Ph.D. dissertation [44]) to predict the cost of splitting a Monte Carlo particle [43]. This work is novel relative to [36] because it brought together the calculation of the tally first and second moments and the expected time per particle history to predict Monte Carlo efficiency using deterministic numerical methods. The work in [36] is strictly analytic. Furthermore, this work treats more-general situations than [36] (e.g., it permits particle scattering and multiple positions that induce splitting). Regardless, this work comes to similar conclusions as [36].

### 1.4.9 Dubi and Burn

While not directly tied to the lineage and development of the HSMEs, in the 1980s Dubi, and later in the 1990s Burn, developed and extended a related technique they named the direct statistical approach (DSA). Dubi's work was focused on exponential transform [45, 46] and importance splitting [47–52]. Burn extended Dubi's importance splitting work in generality [53, 54] and to other variance-reduction techniques [55, 56]. The work by these authors is notable here because the DSA is the forward analog of the adjoint-like HSMEs.

### 1.4.10 Mikhailov

Again not directly tied to the lineage of the HSMEs, in the 1990s Mikhailov published a number of papers and one book, in particular, that address minimization of computational costs for Monte Carlo methods [57]. His approach does not use the HSMEs nor does his work numerically verify the theory presented; however, his publications are of value by providing alternative theoretical grounding in techniques to optimize computational cost.

### 1.4.11 Solomon, Schultis, Booth, and Sood

In 2010, the history-score-moment equations were generalized by Solomon, Schultis, Booth, and Sood to treat weight-dependent variance-reduction techniques (most notably: weight windows) as described in [58, 59]. Weight-dependent transport kernels for weight windows were derived (and other transport kernels for importance splitting, rouletting, weight cutoff, and implicit capture were re-derived). In addition, the FTEs were developed to estimate the computational cost for transport and these variance-reduction techniques in the MCNP code. With that, [58, 59] also demonstrate the effectiveness of this approach and a tool created to carry it out.

Methods to solve the HSMEs and FTE were implemented in the Cost-optimized Variance-reduction Technique (COVRT) software. COVRT is a discrete ordinates-based multi-group radiation transport solver that operates on a Cartesian spatial grid using scattering source iterations to iteratively calculate its statistical-moment and future-time solutions. COVRT then combines the solutions to predict the resulting computational cost, varies the variance-reduction parameters to identify the set that produces the lowest predicted MCNP computational cost, and reports the final parameters to the user to apply to an MCNP calculation.

The main challenge faced when developing COVRT was how to manage the additional component of phase space, weight, as a result of analyzing weight-dependent variance-reduction techniques. COVRT's use of computer memory for temporary storage limited it to small problems and the large system of equations required long calculation times.

COVRT could not directly analyze angle-biasing variance-reduction techniques at the outset of this work. However, it is the only tool known to solve the HSMEs and FTE for other variance-reduction techniques, so it is extended in this work to include the DXTRAN HSMEs and FTE. When DXTRAN is incorporated into COVRT for this work, simplifications are made to the DXTRAN process (Sections 2.4.6) to make it weight independent to overcome some of the aforementioned challenges.

The work in [58, 59] is what this work most closely follows from. The lineage of these techniques can be directly traced back from [58, 59] to [42, 60] and to [34].

## 1.5 Angle-biasing with MCNP DXTRAN Regions

None of the hybrid transport approaches described in Sections 1.3 or 1.4 directly angle bias continuous-direction and continuous-energy Monte Carlo calculations. Early work would use either continuous-energy or multi-group Monte Carlo calculations and directly bias post-collision direction of flight, but the particles were

restricted to traveling in discrete directions [2, 5, 6]. Later work applied direct angle biasing but only for multi-group Monte Carlo calculations [8–17, 19–21, 24–26, 28]. Otherwise, direction of flight is only indirectly influenced by population control methods [18, 22, 23, 29, 30]. The work by Munk and Slaybaugh [31, 32] calculates an angular contributon flux; however, it then integrates over direction and normalizes by scalar forward flux so angular information is lost early in the process.

However, one variance-reduction technique that does permit directly biasing particle direction of flight toward a region of interest, and includes post-collision sampling about the region, is the MCNP DXTRAN technique. The MCNP DXTRAN technique is usable in continuous-direction and continuous-energy Monte Carlo calculations. The history and operation of DXTRAN and similar techniques is described next and is followed by examples of some applications where such techniques have been applied.

### 1.5.1 History of DXTRAN

Unfortunately, the genesis of the DXTRAN variance-reduction technique is unclear. There is no seminal document describing its development or initial implementation into a Monte Carlo code. The first mention of DXTRAN is in the MCNP-A manual [61] from November 1979, but there is no mention of DXTRAN in an earlier revision of this document [62] from July 1978. It is not clear whether this is because DXTRAN was implemented between the two revisions or if the later document was more comprehensive in its coverage.

Tom Booth (Los Alamos National Laboratory, retired) recalls that DXTRAN existed in MCNP's immediate predecessor code MCN [63] and surmises that Ed Cashwell, perhaps in concert with Heikki Kalli, were the originators of the technique during their work on point detector methods [64]. Unfortunately, neither individual could be contacted to confirm this. None of the related publications by Cashwell and Kalli [65, 66] support this recollection. A Ph.D. dissertation from 1987 [67, page 49] implies that Leland Carter developed DXTRAN. A publication by Carter from 1980 uses a modified version of DXTRAN [68]; however, it doesn't suggest that Carter developed the technique and Carter couldn't be reached for comment. As such, the best reference material for DXTRAN is the MCNP theory manual [1] with the details of its creation seemingly lost to time.

In the MCBEND code [69, 70], a technique similar to DXTRAN is referred to as "forced flight." In a research version of the EGSnrc code [71, 72], a technique similar to DXTRAN is referred to as "forced scattering." The approaches used in the MCNP, MCBEND, and EGSnrc codes are functionally the same though the codes provide different capabilities to specify the related geometry and tune the resulting behavior. For the remainder of this work, this variance-reduction method is referred to as DXTRAN because there is no agreed-upon term for such a variance-reduction technique and because it is more terse. Furthermore, as DXTRAN and other Monte Carlo features and capabilities are described, those specific to the MCNP code will be focused upon. This is a practical limit on scope.

### 1.5.2 DXTRAN Overview

DXTRAN biases angular sampling by directing computational particles toward a region of interest following particle emergence from a source or collision event. In addition, DXTRAN biases spatial sampling by projecting the particle to the surface of the region of interest. In this way, DXTRAN increases sampling

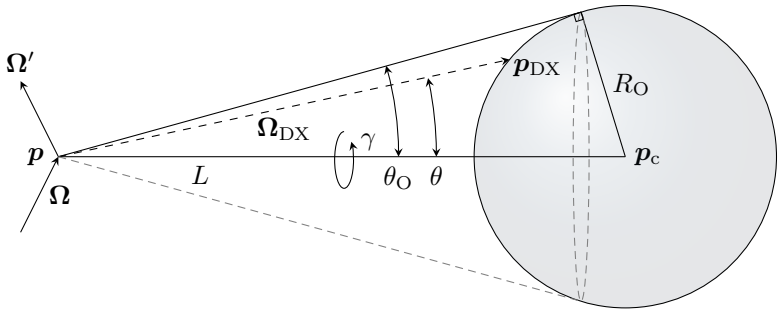



**Figure 1.1:** Simplified Spherical DXTRAN Process Diagram

along user-specified directions and in user-specified regions. It does this by operating after each valid source emission and non-absorptive collision. Either event is valid if there is a non-zero probability of the particle traveling toward the DXTRAN region. An example of an invalid event is a mono-directional source that is not directed toward the DXTRAN region.

Once a valid event is identified at spatial point $\boldsymbol{p}$ and the DXTRAN process is initiated, a computational particle (the DXTRAN particle) is created. The initial direction of the DXTRAN particle, $\boldsymbol{\Omega}_{\mathrm{DX}}$, is sampled from the available directions from $\boldsymbol{p}$ directed toward the DXTRAN region. For a spherical DXTRAN region, the possible directions are represented by a cone with its apex at $\boldsymbol{p}$ and the circular base is the solid angle subtended by the sphere as shown in Fig. 1.1. A projection is then made along $\boldsymbol{\Omega}_{\mathrm{DX}}$ and the DXTRAN particle's initial position, $\boldsymbol{p}_{\mathrm{DX}}$, is where the projection intersects the DXTRAN region. This process thus *directly* moves computational particles toward and into region of interest to increase sampling.

Significantly more detail on the MCNP DXTRAN process is available in Section 2.4.6 including caveats to the method and the simplifications made to it for the purpose of this dissertation.

### 1.5.3 Applications of DXTRAN Regions

In the author's experience, and confirmed by a review of openly available literature, the DXTRAN variance-reduction technique is not widely used. Instead, it is used only in situations that it was specifically designed for: highly angle-dependent scenarios. This can be compared with other variance-reduction techniques such as weight windows, which can and are used more broadly (e.g., to indirectly perform angle biasing) as described in Section 1.3.

DXTRAN use is limited for several reasons. First, scenarios that require DXTRAN regions are generally limited. Often, indirect angle biasing is sufficient as demonstrated by techniques like CADIS and FW-CADIS. Second, because of the dearth of guidance on using DXTRAN many practitioners are not comfortable using it. Some of the guidance that does exist recommends against using DXTRAN regions to perform biasing in deep-penetration problems. This is because DXTRAN regions can draw computational particles toward the region of interest (the DXTRAN region) and can sample locally; however, if particles do not propagate through another method to an important region of phase space, then that region is inadequately sampled. Minimal information is available for how to overcome this inadequate sampling.

With those caveats noted, there are numerous classes of problems that DXTRAN has been successfully applied to. For medical physics applications, DXTRAN has been used for both boron neutron capture therapy planning [73, 74] and planning and optimizing dose to medical staff [75]. Experimental facility analysis has been performed with DXTRAN to characterize neutron time of flight [76] and experimental beam port neutron flux environments [77, 78]. DXTRAN has also been used in neutron generator device analyses [79] and for performing spacecraft nuclear propulsion system shielding analyses [80]. Finally, inertial confinement fusion reactor design analyses have been performed with DXTRAN [81]. Some analyses using DXTRAN without publicly available documents include estimating commercial nuclear reactor polar crane operator dose, performing radioactive material triangulation in an urban environment, and determining energy deposition to satellite components from radiation arising from terrestrial or cosmic sources.

In all of the above scenarios, the DXTRAN parameters such as size and position are based on user skill and experience. The work in this dissertation is the first attempt to develop methodical, computational-cost optimized, guidance for the application of DXTRAN regions. The MCNP code permits multiple DXTRAN spheres to be used in a single calculation. Furthermore, DXTRAN spheres can be wholly nested within each other since version 6.0 Beta 2 of the MCNP code [82]. To limit scope of this work, only a single DXTRAN region is considered; however, analyzing multiple DXTRAN regions and/or wholly nested DXTRAN regions provides a natural extension of this work for the future.

### 1.6 Novelty of Current Work

The DXTRAN technique has received limited attention in the prior literature, despite it being useful for many relevant transport applications. Additionally, effectively applying DXTRAN in calculations is largely experiential and based on empirical observations of computational performance. This dissertation develops new mathematical descriptions of the DXTRAN variance-reduction process (Section 3.7) and demonstrates a new automated variance-reduction method that applies these mathematical formulations to determine the optimal application of DXTRAN in a given problem.

Specifically, this work includes the first known deduction and application of the integral transport kernels for both biasing with DXTRAN particle production (Section 3.7.1) and the associated free-flight transport with truncation of the initiating particle (Section 3.7.2). This dissertation applies these new DXTRAN transport kernels to derive expressions for the mean tally response in Monte Carlo transport calculations involving DXTRAN. These expressions are then used to rigorously prove, for the first time, that the technique is unbiased (Section 3.7.4). This dissertation also derives equations for the variance and associated computational cost of Monte Carlo calculations involving DXTRAN, which are solved using the deterministic $S_N$ method (Section 3.7.5). The DXTRAN transport kernels have unique non-local aspects, which are present in neither analog transport nor any of the other variance-reduction techniques that have been analyzed in prior work, that require significant modification to the conventional $S_N$ transport sweeps. This deterministic $S_N$ solution is then used in an automated variance-reduction scheme that optimizes the placement of DXTRAN regions and the associated spatial biasing parameters. Prior to this work, no deterministic method could provide information relevant to optimizing the use of DXTRAN.

The deterministic $S_N$ solver used for this work operates on a Cartesian spatial mesh. Prior to this work, the Monte Carlo code used only permitted DXTRAN regions defined using spheres so a direct comparison with

the $S_N$ results is not practical. To permit a direct comparison, a method to use DXTRAN regions defined by arbitrary convex polyhedra in the Monte Carlo is developed and verified (Appendix B). No widely available Monte Carlo code exists that permits DXTRAN regions defined using such shapes and such capability has not been demonstrated before.

### 1.7 Outline of Remaining Chapters

Having described the motivation for this work, a historical account of related work, and the novel elements of the current work, this section outlines the remainder of this dissertation.

Chapter 2 provides mathematical concepts and brief background in transport theory and related numerical solution methods. It begins by heuristically deriving a forward neutral-particle transport equation. It then derives alternative adjoint and integral transport formulations. Next, the energy, space, and angle discretizations that allow a deterministic solution to the aforementioned transport equations are given. Monte Carlo particle transport concepts are introduced and variance-reduction techniques applied in this work are discussed.

Chapter 3 begins with previously derived HSMEs by reviewing their analog transport kernels and how they are used in the resulting HSME derivations for analog calculations. The relationship between the analytic HSMEs and Monte Carlo random walks is then described. The chapter concludes with a deduction of the DXTRAN biasing kernel and a derivation of the first two DXTRAN HSMEs (necessary to calculate tally mean and uncertainty).

Chapter 4 describes the basis and development of the FTE similar to the HSMEs in Chapter 2. It continues by discussing how these, along with the HSMEs, can be used with numerical optimization techniques to determine the computational cost (inverse figure of merit) of a Monte Carlo calculation. Next, several suitable (gradient-free, bounded, and constrained) optimization techniques evaluated for use in this work are described.

Chapter 5 describes how the equations from Chapters 3 and 4 are encoded in software. The basic algorithms and program flow are described for a standalone Fortran computer code. In addition, a Python-based framework to interface a Fortran shared object with an external optimization package is described. This chapter provides the linkage between the theory and derivations in Chapters 3 and 5 and the deterministic solution techniques in Chapter 2.

Chapter 6 defines and gives the results for a series of verification test cases to show that the solutions to the HSMEs are correctly obtained using the implementation in Chapter 5. Fourteen 1-D test cases and seven 2-D test cases are used. In these test cases, a variety of forward sources (boundary and distributed), materials (pure absorber and 20% absorber), energy groups (one and two), and tallies (leakage, current, and expected track length) are exercised.

Chapter 7 uses a subset of the test cases described in Chapter 6 to test the optimization described in Chapter 4. In total, twelve optimization test cases are used (six 1-D and six 2-D). Using the results of these optimization calculations, as well as knowledge gained throughout the course of conducting this work, general guidance for using DXTRAN regions is offered at the conclusion of the chapter.

Chapter 8 provides a summary of this work. It also summarizes possible near- and long-term directions that future work could take.

A series of appendices give supplemental methodological and implementation details that are useful for, but would distract from, the main body of this work. These include the interchange between forward- and adjoint-transport operators (Appendix A), an algorithm to use arbitrary convex polyhedra to define DXTRAN regions (Appendix B), procedures to generate the $S_N$ quadrature sets used in this work (Appendix C), a demonstration on DXTRAN's effect on tally variance (Appendix D), a description of the method to obtain MCNP subroutine calculation times and the detailed time data (Appendix E), and Python scripts that demonstrate how the DXTRAN HSMEs can be solved numerically (Appendix F).

# Chapter 2

# Radiation Transport Fundamentals

## 2.1 Introduction

This chapter provides background on theoretical and computational radiation transport approaches. For the computational components, both deterministic (specifically, discrete ordinates) and Monte Carlo techniques are used to compute quantities of interest that can be characterized by the neutral-particle radiation transport equation. Broadly speaking, four main topics are covered in this chapter:

1. The forward and adjoint integro-differential forms of the time-independent fixed-source radiation transport equation and how those forms can be used to determine an integral response of interest,
2. A way to express the time-independent fixed-source radiation transport equation using an integral form,
3. Discretization methods to deterministically solve the time-independent fixed source radiation transport equation, and
4. Monte Carlo methods and variance-reduction techniques to efficiently simulate particle transport behavior.

These topics are reviewed because they must be used to accomplish the goal of this work: solving a Monte Carlo particle transport simulation efficiently by applying variance reduction (chiefly, topic 4). The Monte Carlo simulation provides stochastic estimates of detector response moments and has an overall calculation time, which can be used together to represent the computational cost of the Monte Carlo calculation. The HSMEs (discussed in Chapter 3), are represented using an integral formulation (topic 2) by considering how statistical moments contribute to the Monte Carlo detectors (thus an adjoint-like representation, topic 1). To solve the HSMEs and FTE, a deterministic solver is used by recasting the integral formulation into a more-traditional integro-differential form, discretizing, and solving (topics 2 and 3).

As such, the forward radiation transport balance equation is first developed heuristically. Next, a review of adjoint and integral radiation transport methods is given. Deterministic transport methods and associated discretization techniques are described next to present the terminology and notation that will be used through the remainder of this work. Next, Monte Carlo techniques are discussed.

When describing the radiation transport equation and its solution in this chapter, $f$ is used to represent the quantity of interest, a particle fluence rate density (hereafter referred to simply as flux), rather than

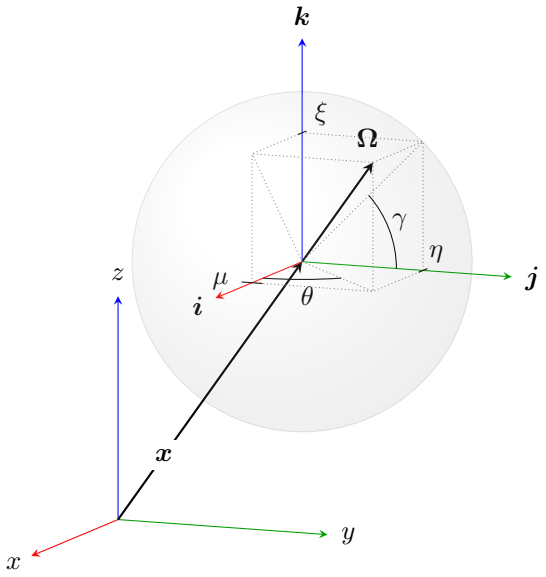



**Figure 2.1:** Position and Angle Phase Space

the more traditional $\psi$, $\phi$, or $\varphi$. This is done because the radiation transport equation can be used to solve for flux directly. However, an integro-differential equation of a similar form can also be used to solve the history-score moment equations (HSMEs) or the future-time equation (FTE) as discussed in Chapters 3 and 4 and implemented later in Chapter 5. Thus, $f$ is used to represent the quantity of interest in the radiation transport equation for this discussion to avoid confusion with symbols used elsewhere.

Supplemental information on radiation transport methods and techniques is available in references such as [39, 83–91].

## 2.2 Transport Theory

This section provides an abbreviated derivation of the linear neutral-particle radiation transport equation. The purpose of this section is to provide supplemental background to the reader such that the subsequent work with the radiation transport equation in this chapter and Chapters 2–5 do not appear to arise without basis. Note that hereafter when the radiation transport equation is referred to its linearity and validity for neutral particles are implied properties.

The goal of radiation transport analyses and calculations is to quantify the behavior of particles in some phase space of interest. This work assumes that

- there is a "large enough" population of particles in a given element of phase space such that statistical fluctuations are insignificant,
- there are no particle-particle interactions,
- particles behave as points (i.e., wave-like behavior is neglected),
- there is no system feedback and no particle-producing precursors,

- there is no fissionable material,
- accelerating forces (e.g., gravity) are neglected,
- correlations between secondary particles are ignored (i.e., conservation laws are only satisfied on average), and
- the atoms in the materials are randomly and isotropically distributed.

As such, this section begins by defining the phase-space of interest, angular particle density, and a typically more useful quantity: angular flux. This section then concludes with a heuristic derivation of the radiation transport equation by considering its role as a conservation equation. The terms used within, and approach to deriving, the radiation transport equation are similar to that used in the University of Michigan Advanced Reactor Theory course notes (course number NERS 543) [90].

### 2.2.1 Phase-space Definition

To fully describe average particle behavior for time-dependent, continuous-energy, three-dimensional, Cartesian geometries seven independent phase-space variables are required: three for space, three for velocity (or one kinetic energy and two direction), and one for time. Note that Cartesian geometry is discussed herein because it is often the most familiar spatial representation, it somewhat simplifies notation and considerations elsewhere, and it underpins the computer code used in this work. Regardless, begin by defining a spatial position $\boldsymbol{x}$ as the vector

$$\boldsymbol{x} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}, \tag{2.1}$$

where $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are unit vectors aligned along the three orthogonal Cartesian axes. Therefore, $x, y, z$ are measures along the axes giving a position in space and have units of length and are three of the independent variables needed. Correspondingly, a differential volume $\mathrm{d}V$ can be expressed as

$$\mathrm{d}V = (\mathrm{d}x)\,(\mathrm{d}y)\,(\mathrm{d}z) \tag{2.2}$$

such that particles can be thought to exist in the region $\mathrm{d}V$ about $\boldsymbol{x}$.

Next, a particle in $\mathrm{d}V$ about $\boldsymbol{x}$ can travel in any direction within the $4\pi$ steradians around $\boldsymbol{x}$. As such, define the particle direction of flight as the dimensionless unit vector

$$\boldsymbol{\Omega} = \Omega_{\mathrm{x}}\boldsymbol{i} + \Omega_{\mathrm{y}}\boldsymbol{j} + \Omega_{\mathrm{z}}\boldsymbol{k}. \tag{2.3}$$

It is often convenient to represent the direction of flight by defining a polar axis (e.g., along $\boldsymbol{i}$) and expressing $\Omega_{\mathrm{x}}$ relative to its angle, $\theta$, relative to the polar axis. More commonly, this is done by defining $\theta$ by its cosine $\mu = \cos\theta = \boldsymbol{\Omega}\cdot\boldsymbol{i}$ so Eq. (2.3) can be rewritten as

$$\boldsymbol{\Omega} = \mu\boldsymbol{i} + \left(\sqrt{1-\mu^2}\cos\gamma\right)\boldsymbol{j} + \left(\sqrt{1-\mu^2}\sin\gamma\right)\boldsymbol{k}, \tag{2.4}$$

where $\gamma$ is the azimuthal direction of flight. This gives $\mu, \gamma$ as two more of the necessary independent variables.

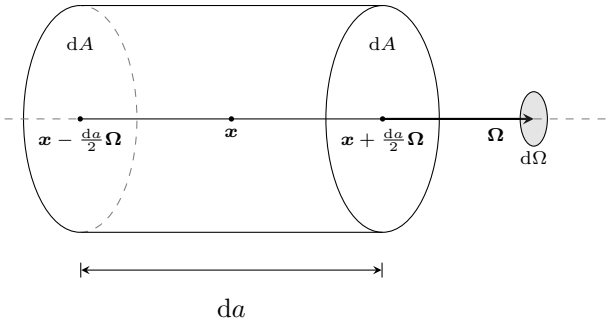



**Figure 2.2:** Differential Control Volume, d$V$

The differential area about $\boldsymbol{\Omega}$ that particles can thought to be flying in, $d\Omega$, can be expressed as

$$\mathrm{d}\Omega = \mathrm{d}\mu\mathrm{d}\gamma \tag{2.5}$$

so particles can be thought to travel in direction d$\Omega$ about $\boldsymbol{\Omega}$. Equation (2.4) can also be rewritten as

$$\boldsymbol{\Omega} = \mu\boldsymbol{i} + \eta\boldsymbol{j} + \xi\boldsymbol{k} \tag{2.6}$$

by making the substitutions

$$\eta = \sqrt{1-\mu^2}\cos\gamma, \tag{2.7a}$$
$$\xi = \sqrt{1-\mu^2}\sin\gamma, \tag{2.7b}$$

where $\mu$, $\eta$, and $\xi$ are referred to as direction cosines. Note that as long as one direction cosine is specified relative to the polar axis, the other two are related to it by Eq. (2.7).

Further, particles can exist in energy intervals d$E$ about $E$ (where energy $E$ is related to a particle's speed through the classic kinematics relationship $E = {}^1\!/_2 mv^2$). Finally, particles can exist in each of these differential components of phase space at any time $t$.

Having defined all components of phase space, the angular particle density $N(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)$ in phase space d$V$ about $\boldsymbol{x}$, d$\Omega$ about $\boldsymbol{\Omega}$, d$E$ about $E$ at time $t$ is the quantity sought.

### 2.2.2 Heuristic Radiation Transport Balance Equation Derivation

To begin, one can recognize that the radiation transport equation is fundamentally a conservation equation over phase space characterizing the various gain and loss mechanisms for angular particle density. As such, one can define a differential control volume d$V$ taken to be a cylinder as shown in Fig. 2.2. The volume's length is $\mathrm{d}a = v\mathrm{d}t$, its cap surface area is d$A$, and its axis is oriented along $\boldsymbol{\Omega}$ (corresponding to the direction of particle flight) where the particles have speed $v = \sqrt{2E/m}$. By defining the control volume in this way, particles only enter and exit through the caps of the cylinder.

Also, one can define the angular fluence rate density or angular flux as $f(\boldsymbol{x}, \boldsymbol{\Omega}, E, t) = vN(\boldsymbol{x}, \boldsymbol{\Omega}, E, t)$. This definition is made because the rate at which particles pass through a unit area is often a more convenient

quantity to obtain the ultimate quantity of interest: the reaction rate of the particles with some material. The angular flux has two interpretations. The first is the rate, per unit area. per unit energy, per unit solid angle that particles cross some differential area oriented normal to the direction of flight. The second is the rate, per unit volume, per unit energy, per unit solid angle that particles generate "path length" within a differential volume, which can be directly related to a quantity of interest, the reaction rate of particles.

Using the differential control volume in Fig. 2.2, one can construct a physical interpretation of the net rate of particles in $\mathrm{d}\Omega\mathrm{d}E$ streaming out of $\mathrm{d}V$ at time $t$ as

$$\begin{aligned}
&= \left[ f\left(\boldsymbol{x} + \frac{\mathrm{d}a}{2}\boldsymbol{\Omega}, \boldsymbol{\Omega}, E, t\right) - f\left(\boldsymbol{x} - \frac{\mathrm{d}a}{2}\boldsymbol{\Omega}, \boldsymbol{\Omega}, E, t\right)\right] \mathrm{d}A\mathrm{d}\Omega\mathrm{d}E \\
&= \left[ \frac{f\left(\boldsymbol{x} + \frac{\mathrm{d}a}{2}\boldsymbol{\Omega}, \boldsymbol{\Omega}, E, t\right) - f\left(\boldsymbol{x} - \frac{\mathrm{d}a}{2}\boldsymbol{\Omega}, \boldsymbol{\Omega}, E, t\right)}{\mathrm{d}a} \right] (\mathrm{d}a\mathrm{d}A)\, \mathrm{d}\Omega\mathrm{d}E \\
&= \left[ \frac{f\left(\boldsymbol{x} + \frac{\mathrm{d}a}{2}\boldsymbol{\Omega}, \boldsymbol{\Omega}, E, t\right) - f\left(\boldsymbol{x} - \frac{\mathrm{d}a}{2}\boldsymbol{\Omega}, \boldsymbol{\Omega}, E, t\right)}{\mathrm{d}a} \right] (\mathrm{d}V)\, \mathrm{d}\Omega\mathrm{d}E \\
&= \frac{\partial f}{\partial a}\mathrm{d}V\mathrm{d}\Omega\mathrm{d}E
\end{aligned}$$

Using the same control volume, one can also calculate the removal of particles caused by collisions in the control volume as

$$\begin{aligned}
&= \Sigma_{\mathrm{t}}\left(\boldsymbol{x}, E\right) \frac{\mathrm{d}a}{\mathrm{d}t} N\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) \mathrm{d}V\mathrm{d}\Omega\mathrm{d}E \\
&= \Sigma_{\mathrm{t}}\left(\boldsymbol{x}, E\right) vN\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) \mathrm{d}V\mathrm{d}\Omega\mathrm{d}E \\
&= \Sigma_{\mathrm{t}}\left(\boldsymbol{x}, E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) \mathrm{d}V\mathrm{d}\Omega\mathrm{d}E,
\end{aligned}$$

where the macroscopic total cross section is a coefficient describing the number of interactions per unit distance traveled per particle. The macroscopic total cross section is assumed to be stationary, instantaneous, independent of direction (i.e., the material is random and isotropic). These two terms, streaming and collision, represent the loss terms of the balance equation.

There are gain terms commonly considered for non-fissile systems. The first gain term constitutes those particles that undergo non-absorptive collision in the control volume and emerge with direction $\boldsymbol{\Omega}$ and energy $E$ that originated within the control volume with any other angle-energy phase space position.

With the assumption that the background nuclei are randomly distributed and oriented, particle scattering is rotationally invariant. That is, during scattering emergence there is an equiprobable distribution of azimuthal emergence directions from scattering about the incident direction $\boldsymbol{\Omega}'$ (i.e., $p\left(\gamma\right) = 1/2\pi$ as shown in Fig. 2.3). As such, following scattering the change in direction from $\boldsymbol{\Omega}'$ to $\boldsymbol{\Omega}$ depends only on the scattering angle $\theta$ between the polar axis $\boldsymbol{\Omega}'$ and $\boldsymbol{\Omega}$ so

$$\Sigma_{\mathrm{s}}\left(\boldsymbol{x}, \boldsymbol{\Omega}', E' \rightarrow \boldsymbol{\Omega}, E\right) = \Sigma_{\mathrm{s}}\left(\boldsymbol{x}, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}, E' \rightarrow E\right). \tag{2.8}$$

In Eq. (2.8), the double-differential macroscopic scattering cross section gives the probability per unit distance traveled per particle that a particle that undergoes a scattering collision at $\boldsymbol{x}$ with initial direction $\boldsymbol{\Omega}'$ and

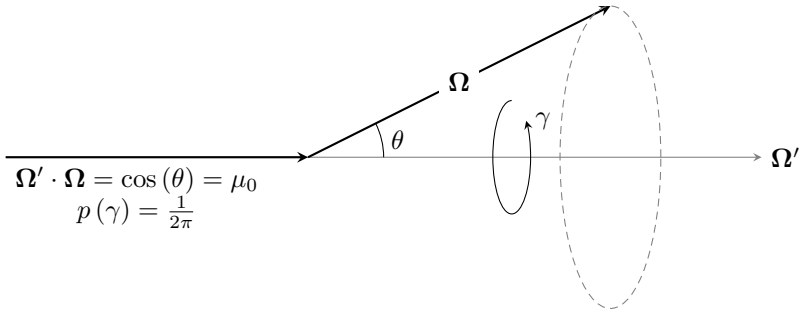



**Figure 2.3:** Scattering Emission with Rotational Invariance

energy $\boldsymbol{E}'$ will emerge in direction $\mathrm{d}\Omega$ about $\boldsymbol{\Omega}$ and $\mathrm{d}E$ about $E$. Thus, the scattering gain term can be expressed as

$$\int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \Sigma_\mathrm{s}\left(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E' \to E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}', E', t\right) \mathrm{d}V \mathrm{d}\Omega' \mathrm{d}E'.$$

This assumes non-absorptive collisions are stationary and instantaneous.

Also, if there is a fixed source term within the control volume, it will introduce particles into the control volume as

$$Q\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E.$$

Thus, the time rate of change of the particle density in the system can be written as

$$\frac{\partial N}{\partial t} = \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \Sigma_\mathrm{s}\left(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E' \to E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}', E', t\right) \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E$$
$$Q\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E - \frac{\partial f}{\partial a} \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E - \Sigma_\mathrm{t}\left(\boldsymbol{x}, E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E \quad (2.9)$$

or, more commonly by remembering that $f = vN$, as

$$\frac{1}{v}\frac{\partial f}{\partial t} + \frac{\partial f}{\partial a} \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E + \Sigma_\mathrm{t}\left(\boldsymbol{x}, E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E =$$
$$\int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \Sigma_\mathrm{s}\left(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E' \to E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}', E', t\right) \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E + Q\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) \mathrm{d}V \mathrm{d}\Omega \mathrm{d}E. \quad (2.10)$$

One can then divide through by $\mathrm{d}V \mathrm{d}\Omega \mathrm{d}E$ to obtain the radiation transport equation:

$$\frac{1}{v}\frac{\partial f}{\partial t} + \frac{\partial f}{\partial a} + \Sigma_\mathrm{t}\left(\boldsymbol{x}, E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right) =$$
$$\int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \Sigma_\mathrm{s}\left(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E' \to E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}', E', t\right) + Q\left(\boldsymbol{x}, \boldsymbol{\Omega}, E, t\right). \quad (2.11)$$

However, it is often more convenient to define streaming in the global spatial coordinate system rather than

along a characteristic ray. One can make this redefinition by applying the chain rule to find

$$\frac{\partial f}{\partial a} = \frac{\partial x}{\partial a}\frac{\partial f}{\partial x} + \frac{\partial y}{\partial a}\frac{\partial f}{\partial y} + \frac{\partial z}{\partial a}\frac{\partial f}{\partial z}, \tag{2.12}$$

where $\partial x/\partial a = \boldsymbol{\Omega}\cdot\boldsymbol{i}$, $\partial y/\partial a = \boldsymbol{\Omega}\cdot\boldsymbol{j}$, and $\partial z/\partial a = \boldsymbol{\Omega}\cdot\boldsymbol{k}$ so one can rewrite

$$\frac{\partial f}{\partial a} = \boldsymbol{\Omega}\cdot\boldsymbol{\nabla} f. \tag{2.13}$$

One must also define boundary and initial conditions. The initial condition can be specified throughout the system as

$$f\left(\boldsymbol{x},\boldsymbol{\Omega},E,0\right) = f_{\mathrm{i}}\left(\boldsymbol{x},\boldsymbol{\Omega},E\right)\ ,\ \boldsymbol{\Omega}\in 4\pi\ ,\ 0\leq E<\infty. \tag{2.14}$$

Meanwhile, to define the boundary condition one may assume a convex system surrounded by vacuum. Doing this restricts one to specifying the boundary condition only for inward-facing directions relative to the outward-facing surface normal $\boldsymbol{n}$ such that $\boldsymbol{\Omega}\cdot\boldsymbol{n}<0$. As such, the boundary condition can be specified as

$$f\left(\boldsymbol{x},\boldsymbol{\Omega},E,t\right) = f_{\mathrm{b}}\left(\boldsymbol{x},\boldsymbol{\Omega},E,t\right)\ ,\ \boldsymbol{x}\in\partial V\ ,\ \boldsymbol{\Omega}\cdot\boldsymbol{n}<0\ ,\ 0\leq E<\infty\ ,\ 0\leq t. \tag{2.15}$$

Thus, one obtains the forward time-dependent radiation transport equation, with boundary and initial conditions, used to model average neutral particle behavior in a stationary convex system without fission surrounded by vacuum,

$$\begin{aligned}
&\frac{1}{v}\frac{\partial f\left(\boldsymbol{R},t\right)}{\partial t} + \boldsymbol{\Omega}\cdot\boldsymbol{\nabla} f\left(\boldsymbol{R},t\right) + \Sigma_{\mathrm{t}}\left(\boldsymbol{x},E\right) f\left(\boldsymbol{R},t\right)\\
&\quad = \int_{4\pi}\mathrm{d}\Omega'\int_{0}^{\infty}\mathrm{d}E'\Sigma_{\mathrm{s}}\left(\boldsymbol{x},\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}',E'\rightarrow E\right) f\left(\boldsymbol{R}',t\right)\\
&\qquad + Q\left(\boldsymbol{R},t\right)\ ,\ \boldsymbol{x}\in V\ ,\ \boldsymbol{\Omega}\in 4\pi\ ,\ 0\leq E<\infty\ ,\ 0<t, &\text{(2.16a)}\\
&f\left(\boldsymbol{R},t\right) = f_{\mathrm{b}}\left(\boldsymbol{R},t\right)\ ,\ \boldsymbol{x}\in\partial V\ ,\ \boldsymbol{\Omega}\cdot\boldsymbol{n}<0\ ,\ 0\leq E<\infty\ ,\ 0<t, &\text{(2.16b)}\\
&f\left(\boldsymbol{R},0\right) = f_{\mathrm{i}}\left(\boldsymbol{R}\right)\ ,\ \boldsymbol{\Omega}\in 4\pi\ ,\ 0\leq E<\infty, &\text{(2.16c)}
\end{aligned}$$

where

$v$ is the speed of a particle corresponding to its kinetic energy through the non-relativistic relationship $v=\sqrt{2E/m}$, where $m$ is the particle's mass,

$f\left(\boldsymbol{R},t\right)$ is the angular flux or similar flux-like quantity (e.g., history-score moments) at phase space $\boldsymbol{R}$ at time $t$,

$\boldsymbol{R}=\left(\boldsymbol{x},\boldsymbol{\Omega},E\right)$ is the time-independent traditional phase space under consideration,

$\boldsymbol{x}$ is the particle's position within the spatial domain, e.g., $\boldsymbol{x}=x\boldsymbol{i}+y\boldsymbol{j}+z\boldsymbol{k}$,

$\boldsymbol{\Omega}$ is the particle's direction of flight unit vector within the angular domain, e.g., $\boldsymbol{\Omega}=\Omega_{\mathrm{x}}\boldsymbol{i}+\Omega_{\mathrm{y}}\boldsymbol{j}+\Omega_{\mathrm{z}}\boldsymbol{k}$,

$E$ is the particle's kinetic energy,

$\Sigma_{\mathbf{t}}(\boldsymbol{x}, E)$ is the macroscopic total cross section,

$\Sigma_{\mathbf{s}}(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E' \to E)$ is the double-differential macroscopic scattering cross section,

$Q(\boldsymbol{R}, t)$ is an optional inhomogeneous source of particles at time $t$,

$f_{\mathbf{b}}(\boldsymbol{R}, t)$ is the associated boundary condition at time $t$,

$f_{\mathbf{i}}(\boldsymbol{R})$ is the initial condition of the system,

$V$ is the interior volume of the convex system,

$\partial V$ is the two-sided exterior surface of the convex system, and

$\boldsymbol{n}$ is the outward-directed unit normal vector for a given point on the surface, $\partial V$, of the system.

For simplicity, this work will only consider the time-independent (steady-state) form of the radiation transport equation. This is obtained by integrating over $0 \leq t < \infty$ to obtain

$$\boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f(\boldsymbol{R}) + \Sigma_{\mathrm{t}}(\boldsymbol{x}, E) f(\boldsymbol{R}) = \int_{4\pi} \mathrm{d}\Omega' \int_{0}^{\infty} \mathrm{d}E' \Sigma_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}, E' \to E) f(\boldsymbol{R}') + Q(\boldsymbol{R}), \tag{2.17a}$$

$$\boldsymbol{x} \in V, \boldsymbol{\Omega} \in 4\pi, 0 \leq E < \infty,$$

$$f(\boldsymbol{R}) = f_{\mathrm{b}}(\boldsymbol{R}), \boldsymbol{x} \in \partial V, \boldsymbol{\Omega} \cdot \boldsymbol{n} < 0, 0 \leq E < \infty. \tag{2.17b}$$

Equation (2.17a) acts as a conservation equation with loss terms (streaming and collision) on the left-hand side and sources (scattering and inhomogeneous) on the right-hand side. As noted before, it is assumed that the system is convex and that outside the system there is only vacuum (so there are no re-entrant particles). Further, there is a specified inward-directed ($\boldsymbol{\Omega} \cdot \boldsymbol{n} < 0$) boundary source (potentially zero).

The right-hand side of Eq. (2.17a) can be combined as

$$q(\boldsymbol{R}) = \int_{4\pi} \mathrm{d}\Omega' \int_{0}^{\infty} \mathrm{d}E' \Sigma_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}, E' \to E) f(\boldsymbol{R}') + Q(\boldsymbol{R}) \tag{2.18}$$

to form a total source term that includes scattering driven by the flux solution and any inhomogeneous sources.

### 2.2.3 Adjoint Radiation Transport

Thus far all discussion in this chapter has focused on the forward form of the radiation transport equation. However, understanding the adjoint radiation transport equation is relevant because the HSMEs solved in this dissertation resemble it and in some cases can be written the same way. This is because the adjoint form can be used to understand the potential contribution to the detector from all points in phase space of the system.

The forward form of the radiation transport equation follows particles from emission from the forward source, through the system, to a detector and/or termination. The response of the detector can then be computed by integrating the flux over the phase space occupied by the detector response function, $D(\boldsymbol{R})$, to obtain the integral detector response:

$$D_{\mathrm{I}} = \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \, D(\boldsymbol{x}, \boldsymbol{\Omega}, E) \, f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \,. \tag{2.19}$$

The radiation transport equation can also be written in an adjoint form, which can also be used to compute the integral detector response. The expected contribution to the particular detector can be thought of as the adjoint flux.

To determine the adjoint flux, one can begin by defining the inner product of two functions as

$$\langle g, h \rangle = \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \, g(\boldsymbol{x}, \boldsymbol{\Omega}, E) \, h(\boldsymbol{x}, \boldsymbol{\Omega}, E) \,. \tag{2.20}$$

Then, using the "migration" terms in Eq. (2.17a) one can use Eq. (2.20) to compute the inner product

$$\begin{aligned} \langle g, \mathcal{M} f \rangle = \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \, g(\boldsymbol{x}, \boldsymbol{\Omega}, E) \Bigg[ & \boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f(\boldsymbol{x}, \boldsymbol{\Omega}, E) + \Sigma_{\mathrm{t}}(\boldsymbol{x}, E) \, f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \\ & - \int_{4\pi} \mathrm{d}\Omega' \int_0^\infty \mathrm{d}E' \Sigma_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}, E' \to E) \, f(\boldsymbol{x}, \boldsymbol{\Omega}', E') \Bigg] \,. \end{aligned} \tag{2.21}$$

It can be shown (see Appendix A) that Eq. (2.21) can be rewritten (neglecting a surface term arising from the divergence theorem) as

$$\begin{aligned} \langle \mathcal{M}^* g, f \rangle = \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \, f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \Bigg[ & - \boldsymbol{\Omega} \cdot \boldsymbol{\nabla} g(\boldsymbol{x}, \boldsymbol{\Omega}, E) + \Sigma_{\mathrm{t}}(\boldsymbol{x}, E) \, g(\boldsymbol{x}, \boldsymbol{\Omega}, E) \\ & - \int_{4\pi} \mathrm{d}\Omega' \int_0^\infty \mathrm{d}E' \Sigma_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E \to E') \, g(\boldsymbol{x}, \boldsymbol{\Omega}', E') \Bigg] \,. \end{aligned} \tag{2.22}$$

In Eq. (2.22) two notable changes have take place relative to Eq. (2.21). First, the streaming operator has been reversed—it is now negative. This means that particles travel opposite to their associated direction (i.e., a particle oriented along $\boldsymbol{\Omega}$ travels in direction $-\boldsymbol{\Omega}$). Second, the direction of scattering, in angle and energy, has been reversed. This means that particles scatter into collisions (and subsequently the reverse direction in energy than in the forward sense).

The result shown in Eq. (2.22) allows one to write the adjoint form of the radiation transport equation by

replacing $g$ with $f^*$:

$$-\boldsymbol{\Omega}\cdot\boldsymbol{\nabla}f^*\left(\boldsymbol{R}\right)+\Sigma_{\mathrm{t}}\left(\boldsymbol{x},E\right)f^*\left(\boldsymbol{R}\right)=\int\limits_{4\pi}\mathrm{d}\Omega'\int\limits_{0}^{\infty}\mathrm{d}E'\Sigma_{\mathrm{s}}\left(\boldsymbol{x},\boldsymbol{\Omega}\cdot\boldsymbol{\Omega}',E\rightarrow E'\right)f^*\left(\boldsymbol{R}'\right)+Q^*\left(\boldsymbol{R}\right), \tag{2.23a}$$

$$f^*\left(\boldsymbol{R}\right)=f_{\mathrm{b}}^*\left(\boldsymbol{R}\right)\ ,\ \boldsymbol{x}\in\partial V\ ,\ \boldsymbol{\Omega}\cdot\boldsymbol{n}>0\ ,\ 0<E<\infty. \tag{2.23b}$$

Note that the boundary source term is defined for all outward directions on the boundary in Eq. (2.23) (cf. Eq. (2.15)).

In Eq. (2.23), the forward source term, $Q\left(\boldsymbol{R}\right)$, has been replaced by an adjoint source term $Q^*\left(\boldsymbol{R}\right)$. For adjoint calculations, the source term is the response of interest so

$$Q^*\left(\boldsymbol{R}\right)=D\left(\boldsymbol{R}\right). \tag{2.24}$$

By defining the adjoint source in this manner, the adjoint flux, $f^*\left(\boldsymbol{R}\right)$, represents the expected contribution to the response as a result of migrating away from the detector region. Thus, the integral detector responses calculated using Eq. (2.17) and Eq. (2.23) are related as

$$D_{\mathrm{I}}=\left\langle f,Q^*\right\rangle=\left\langle f^*,Q\right\rangle. \tag{2.25}$$

Because of the similarity of Eq. (2.23) to Eq. (2.17), the discretizations described in Sections 2.3.1–2.3.3 and the source iteration solution algorithm (Algorithm 2.1) described in Section 2.3.4 also apply to Eq. (2.23). The main modification to Algorithm 2.1 is to reverse the energy group iterations and to transpose the scattering matrix. This accommodates the energy-angle change as a result of scattering. It is important to recognize the relationship of the forward and adjoint quantities and how the integral detector response is calculating using the adjoint flux and to recognize that the adjoint flux represents expected contributions to the detector.

### 2.2.4 Integral Radiation Transport

Thus far, all forms of the radiation transport equation are integro-differential equations (i.e., containing both integral and differential terms for the scattering and streaming terms, respectively). However, an alternative formulation exists that allows one to calculate $f\left(\boldsymbol{R}\right)$ strictly using integration. Understanding this alternative approach is important to this dissertation because the equations solved in this work are path integral formulations. Thus, understanding how a strictly integral form of the particle transport equation behaves, which includes a recursive relationship, provides a useful comparison when discussing how the path integral formulations are used to represent particle random walk behavior. A strictly integral transport equation is developed next. See [84, 89, 92] for more information.

To begin, one can use Eqs. (2.11) and (2.18) to obtain

$$\frac{\partial f\left(\boldsymbol{R}\right)}{\partial a}+\Sigma_{\mathrm{t}}\left(\boldsymbol{x},\boldsymbol{\Omega},E\right)f\left(\boldsymbol{R}\right)=q\left(\boldsymbol{R}\right)\ ,\ \boldsymbol{x}\in V\ ,\ \boldsymbol{\Omega}\in4\pi\ ,\ 0\leq E<\infty, \tag{2.26a}$$

$$f\left(\boldsymbol{R}\right)=f_{\mathrm{b}}\left(\boldsymbol{R}\right)\ ,\ \boldsymbol{x}\in\partial V\ ,\ \boldsymbol{\Omega}\cdot\boldsymbol{n}<0\ ,\ 0\leq E<\infty. \tag{2.26b}$$

The streaming term is the rate of change in particle population along the direction of travel. Thus, one can define $\boldsymbol{x} \equiv \boldsymbol{x}' - a\boldsymbol{\Omega}$ to rewrite Eq. (2.26a) as

$$\frac{\partial}{\partial a} f\left(\boldsymbol{x}' - a\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right) + \Sigma_{\mathrm{t}}\left(\boldsymbol{x}' - a\boldsymbol{\Omega}, E\right) f\left(\boldsymbol{x}' - a\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right) = q\left(\boldsymbol{x}' - a\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right). \tag{2.27}$$

Multiply by the integrating factor $\exp\left(-\int_0^a \mathrm{d}a' \Sigma_{\mathrm{t}}\left(\boldsymbol{x}' - a'\boldsymbol{\Omega}, E\right)\right)$, and replace $\boldsymbol{x}'$ with $\boldsymbol{x}$ to obtain

$$\frac{\partial}{\partial a}\left[f\left(\boldsymbol{x} - a\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right) \exp\left(-\int_0^a \mathrm{d}a' \Sigma_{\mathrm{t}}\left(\boldsymbol{x} - a'\boldsymbol{\Omega}, E\right)\right)\right] = q\left(\boldsymbol{x} - a\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right) \exp\left(-\int_0^a \mathrm{d}a' \Sigma_{\mathrm{t}}\left(\boldsymbol{x} - a'\boldsymbol{\Omega}, E\right)\right). \tag{2.28}$$

One can then integrate over $a$ along $-\boldsymbol{\Omega}$ from zero to the distance to the system boundary, $a_{\mathrm{s}}$, to obtain

$$f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right) = \int_0^{a_{\mathrm{s}}} \mathrm{d}a' q\left(\boldsymbol{x} - a'\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right) \exp\left(-\int_0^{a'} \mathrm{d}a'' \Sigma_{\mathrm{t}}\left(\boldsymbol{x} - a''\boldsymbol{\Omega}, E\right)\right). \tag{2.29a}$$

This result indicates that the angular flux at some point $\mathrm{d}V$ about $\boldsymbol{x}$ within a given element of angle-energy phase space equals the production of particles from all positions backwards along the direction $-\boldsymbol{\Omega}$ to the system boundary attenuated by the free-flight probability of reaching position $\boldsymbol{x}$. In turn, the production within the combined source term,

$$q\left(\boldsymbol{x} - a\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right) = \int_{4\pi} \mathrm{d}\Omega' \int_0^{\infty} \mathrm{d}E' \Sigma_{\mathrm{s}}\left(\boldsymbol{x} - a\boldsymbol{\Omega}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E \to E'\right) f\left(\boldsymbol{x} - a\boldsymbol{\Omega}, \boldsymbol{\Omega}', E'\right) + Q\left(\boldsymbol{x} - a\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right) + f_{\mathrm{b}}\left(\boldsymbol{x} - a\boldsymbol{\Omega}, \boldsymbol{\Omega}, E\right) \delta\left(a - a_{\mathrm{s}}\right) \tag{2.29b}$$

along this traversed path is from the scattering source, the inhomogeneous source, and any incident boundary source. The scattering source is, in turn, a function of all streaming into each position $\boldsymbol{x} - a\boldsymbol{\Omega}$. Thus, Eq. (2.29a) represents a recursive integral equation to determine the flux in some phase-space element of interest.

### 2.3 Deterministic Radiation Transport

In only rare circumstances can Eq. (2.17) be solved analytically. Therefore, one approach to solving Eq. (2.17) is to discretize the energy, space, and angle domains (using discrete ordinates) and then solve the resulting system of equations. This approach is deterministic in nature, suffering primarily from truncation errors introduced as each component of phase space is discretized. Other discretizations are possible such as spherical harmonics ($P_N$) and the method of characteristics (MOC); however, for this work only the discrete ordinates discretization is used. The discretization of each of these domains is discussed in the next three subsections with an overview of one of the most simple and common solution algorithms, scattering source iteration, given in Section 2.3.4.

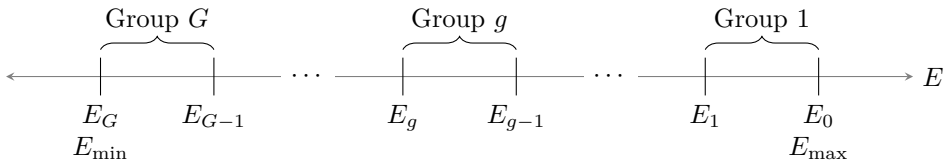



**Figure 2.4:** Multi-group Energy Mesh Bin Structure

### 2.3.1 Discretization of the Energy Domain

For this dissertation, all multi-group cross sections used are specified using artificial values. This discussion shows how multi-group cross sections are computed for completeness even though this work does not do this. The most common technique to discretize the energy domain is the multi-group method where the energy domain of $0 \leq E < \infty$ is subdivided into an energy mesh with $G$ discrete bins (groups) with boundaries

$$0 \leq E_G = E_{\min} < E_{G-1} < \cdots < E_g < E_{g-1} \cdots < E_1 < E_0 = E_{\max} < \infty$$

as shown in Fig. 2.4. The number of energy groups generally depends on the balance between (a) the level of detail needed in the calculation, with more groups able to better represent the entire energy domain by providing greater energy resolution around energy regions of interest (e.g., where resonances exist) and (b) practical computational limitations.

Some reactor physics calculations can be performed with a few groups (often just two groups to represent the fast and thermal energy regimes). However, radiation shielding calculations often require dozens to hundreds of groups to represent the large energy range that particles must traverse as they slow down from the source en route to the detector. To demonstrate the effect of different group boundaries, Fig. 2.5 shows two 47-group representations of the $^{56}$Fe total microscopic cross section for incident neutrons. The first representation is from the HELIOS reactor physics library [93] and the second is from the BUGLE-B7 reactor pressure vessel fluence analysis library [94]. Despite having the same number of energy groups, the effect of different energy bin boundaries is apparent. In particular, HELIOS has just one energy group around 20 keV but the BUGLE-B7 library has several to represent the local resonance and interference behavior. Conversely, the HELIOS library has several groups around 0.1 eV to better represent this region—an important energy regime for thermal fissile systems—while the BUGLE-B7 library has just one group below 0.1 eV because such low energy neutrons cause little damage to reactor structural components.

To construct the multi-group approximation to the radiation transport equation, begin by defining

$$f_g(\boldsymbol{x}, \boldsymbol{\Omega}) \equiv \int_{E_g}^{E_{g-1}} \mathrm{d}E\, f(\boldsymbol{x}, \boldsymbol{\Omega}, E)\,, \tag{2.30a}$$

$$Q_g(\boldsymbol{x}, \boldsymbol{\Omega}) \equiv \int_{E_g}^{E_{g-1}} \mathrm{d}E\, Q(\boldsymbol{x}, \boldsymbol{\Omega}, E)\,, \tag{2.30b}$$

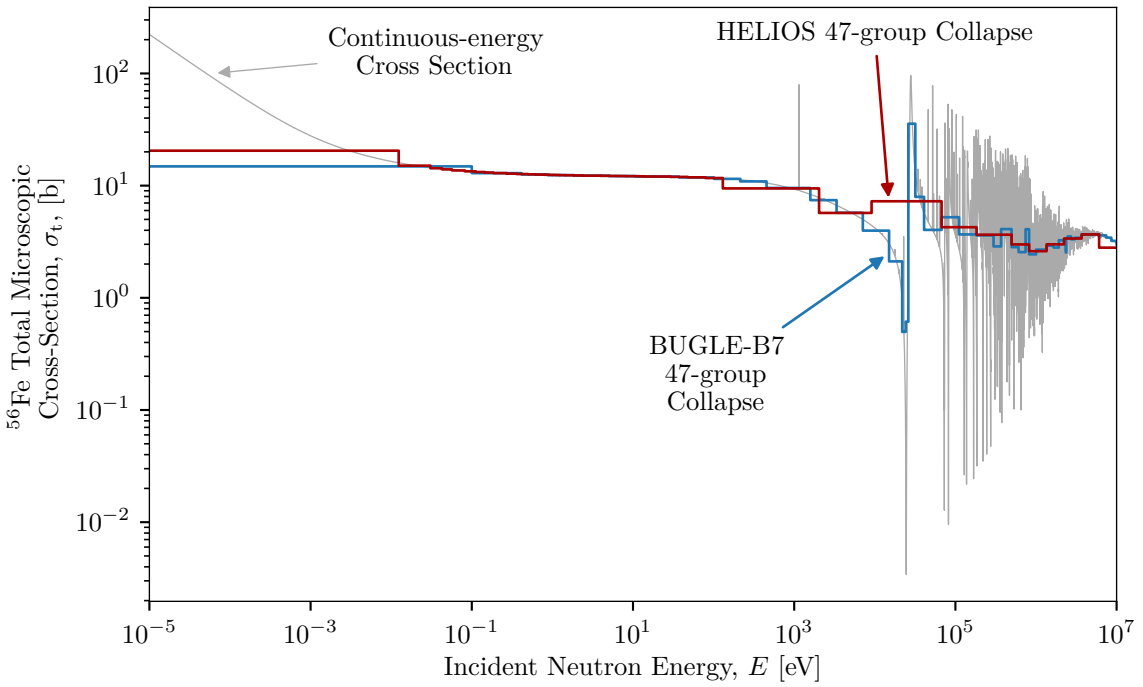



**Figure 2.5:** Demonstration of Varying Group Structure for Multi-group Cross Sections

and integrate Eq. (2.17) over each group $g$ to obtain

$$\int_{E_g}^{E_{g-1}} \mathrm{d}E\, \boldsymbol{\Omega}\cdot\boldsymbol{\nabla} f\left(\boldsymbol{x},\boldsymbol{\Omega},E\right) + \int_{E_g}^{E_{g-1}} \mathrm{d}E\, \Sigma_{\mathrm{t}}\left(\boldsymbol{x},E\right) f\left(\boldsymbol{x},\boldsymbol{\Omega},E\right)$$
$$= \sum_{g'=1}^{G} \int_{E_g}^{E_{g-1}} \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega' \int_{E_{g'}}^{E_{g'-1}} \mathrm{d}E' \Sigma_{\mathrm{s}}\left(\boldsymbol{x},\boldsymbol{\Omega}'\cdot\boldsymbol{\Omega},E\to E'\right) f\left(\boldsymbol{x},\boldsymbol{\Omega}',E'\right) + \int_{E_g}^{E_{g-1}} \mathrm{d}E\, Q\left(\boldsymbol{x},\boldsymbol{\Omega},E'\right),$$
$$\boldsymbol{x}\in V\,,\ \boldsymbol{\Omega}\in 4\pi\,,\ g=1,2,\ldots,G, \quad (2.31a)$$

$$\int_{E_g}^{E_{g-1}} \mathrm{d}E\, f\left(\boldsymbol{x},\boldsymbol{\Omega}',E'\right) = \int_{E_g}^{E_{g-1}} \mathrm{d}E\, f_{\mathrm{b}}\left(\boldsymbol{x},\boldsymbol{\Omega}',E'\right)\ ,\ \boldsymbol{x}\in\partial V\,,\ \boldsymbol{\Omega}\cdot\boldsymbol{n}<0\,,\ g=1,2,\ldots,G, \quad (2.31b)$$

which can then be written as

$$\boldsymbol{\Omega}\cdot\boldsymbol{\nabla} f_g\left(\boldsymbol{x},\boldsymbol{\Omega}\right) + \int_{E_g}^{E_{g-1}} \mathrm{d}E\, \Sigma_{\mathrm{t}}\left(\boldsymbol{x},E\right) f\left(\boldsymbol{x},\boldsymbol{\Omega},E\right) \frac{f_g\left(\boldsymbol{x},\boldsymbol{\Omega}\right)}{\int_{E_g}^{E_{g-1}} \mathrm{d}E\, f\left(\boldsymbol{x},\boldsymbol{\Omega},E\right)}$$
$$= \sum_{g'=1}^{G} \int_{E_g}^{E_{g-1}} \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega' \int_{E_{g'}}^{E_{g'-1}} \mathrm{d}E' \Sigma_{\mathrm{s}}\left(\boldsymbol{x},\boldsymbol{\Omega}'\cdot\boldsymbol{\Omega},E\to E'\right) f\left(\boldsymbol{x},\boldsymbol{\Omega}',E'\right) \frac{f_{g'}\left(\boldsymbol{x},\boldsymbol{\Omega}\right)}{\int_{E_{g'}}^{E_{g'-1}} \mathrm{d}E'\, f\left(\boldsymbol{x},\boldsymbol{\Omega},E'\right)} + Q_g\left(\boldsymbol{x},\boldsymbol{\Omega}\right),$$
$$\boldsymbol{x}\in V\,,\ \boldsymbol{\Omega}\in 4\pi\,,\ g=1,2,\ldots,G, \quad (2.32a)$$

$$f_g(\boldsymbol{x},\boldsymbol{\Omega}) = f_{\mathrm{b},g}(\boldsymbol{x},\boldsymbol{\Omega})\ ,\ \boldsymbol{x}\in\partial V\ ,\ \boldsymbol{\Omega}\cdot\boldsymbol{n}<0\ ,\ g=1,2,\ldots,G. \tag{2.32b}$$

Next, the angular flux is separated into two functions by removing the continuous space-angle dependence from energy as

$$f(\boldsymbol{x},\boldsymbol{\Omega},E) \approx F(\boldsymbol{x},E)\, \digamma(\boldsymbol{x},\boldsymbol{\Omega})\,. \tag{2.33}$$

The function $F(\boldsymbol{x},E)$ represents the energy spectrum with an assumed piecewise-constant spatial dependence, and $\digamma(\boldsymbol{x},\boldsymbol{\Omega})$ provides continuous space-angle information that can be used with $F(\boldsymbol{x},E)$ to reconstruct the angular flux. This allows one to define

$$\Sigma_{\mathrm{t},g}(\boldsymbol{x}) \equiv \frac{\int_{E_g}^{E_{g-1}} \mathrm{d}E\, \Sigma_{\mathrm{t}}(\boldsymbol{x},E)\, F(\boldsymbol{x},E)}{\int_{E_g}^{E_{g-1}} \mathrm{d}E\, F(\boldsymbol{x},E)}, \tag{2.34a}$$

$$\Sigma_{\mathrm{s},g'\to g}(\boldsymbol{x},\boldsymbol{\Omega}'\cdot\boldsymbol{\Omega}) \equiv \frac{\int_{E_g}^{E_{g-1}} \mathrm{d}E \int_{E_{g'}}^{E_{g'-1}} \mathrm{d}E' \Sigma_{\mathrm{s}}(\boldsymbol{x},\boldsymbol{\Omega}'\cdot\boldsymbol{\Omega},E\to E')\, F(\boldsymbol{x},E')}{\int_{E_{g'}}^{E_{g'-1}} \mathrm{d}E'\, F(\boldsymbol{x},E')} \tag{2.34b}$$

to yield the $G$ multi-group radiation transport equations

$$\boldsymbol{\Omega}\cdot\boldsymbol{\nabla} f_g(\boldsymbol{x},\boldsymbol{\Omega}) + \Sigma_{\mathrm{t},g}(\boldsymbol{x})\, f_g(\boldsymbol{x},\boldsymbol{\Omega}) = \int_{4\pi} \mathrm{d}\Omega' \sum_{g'=1}^{G} \Sigma_{\mathrm{s},g'\to g}(\boldsymbol{x},\boldsymbol{\Omega}'\cdot\boldsymbol{\Omega})\, f_{g'}(\boldsymbol{x},\boldsymbol{\Omega}') + Q_g(\boldsymbol{x},\boldsymbol{\Omega})\,,$$
$$\boldsymbol{x}\in V\ ,\ \boldsymbol{\Omega}\in 4\pi\ ,\ g=1,2,\ldots,G, \tag{2.35a}$$

$$f_g(\boldsymbol{x},\boldsymbol{\Omega}) = f_{\mathrm{b},g}(\boldsymbol{x},\boldsymbol{\Omega})\ ,\ \boldsymbol{x}\in\partial V\ ,\ \boldsymbol{\Omega}\cdot\boldsymbol{n}<0\ ,\ g=1,2,\ldots,G. \tag{2.35b}$$

The $G$ equations are coupled by the scattering term. Particles that undergo scattering collisions often emerge with a lower energy than they entered the collision with. These types of collisions are called down-scattering and dominate most radiation shielding situations. Less commonly in radiation shielding situations, a particle can emerge from a collision with an energy greater than it entered the collision with. This is called up-scattering, but because it is usually insignificant it is often neglected.

For a single energy group (a mono-energetic problem), one can drop the $g$ subscript and Eq. (2.35) can be written as

$$\boldsymbol{\Omega}\cdot\boldsymbol{\nabla} f(\boldsymbol{x},\boldsymbol{\Omega}) + \Sigma_{\mathrm{t}}(\boldsymbol{x})\, f(\boldsymbol{x},\boldsymbol{\Omega}) = \int_{4\pi} \mathrm{d}\Omega' \Sigma_{\mathrm{s}}(\boldsymbol{x},\boldsymbol{\Omega}'\cdot\boldsymbol{\Omega})\, f(\boldsymbol{x},\boldsymbol{\Omega}') + Q(\boldsymbol{x},\boldsymbol{\Omega})\,, \boldsymbol{x}\in V\ ,\ \boldsymbol{\Omega}\in 4\pi, \tag{2.36a}$$

$$f(\boldsymbol{x},\boldsymbol{\Omega}) = f_{\mathrm{b}}(\boldsymbol{x},\boldsymbol{\Omega})\ ,\ \boldsymbol{x}\in\partial V\ ,\ \boldsymbol{\Omega}\cdot\boldsymbol{n}<0. \tag{2.36b}$$

Because of the approximation introduced to separate $f(\boldsymbol{R})$, a weighting spectrum $F(\boldsymbol{x},E)$ is required when creating multi-group cross sections from evaluated nuclear data. This means that a solution, or something sufficiently close, to the problem at hand is needed (which is problematic because $F(\boldsymbol{x},E)$ is part of the quantity sought, $f(\boldsymbol{R})$). In practice, representative weighting spectra are defined for the type of calculation that the multi-group cross-section library will be used for. Multi-group cross-section sets are then generated

for isotopes and reactions of interest using those spectra. As long as subsequent calculations using those multi-group cross sections abide by the assumptions of the weighting spectra, the multi-group cross sections are valid and will not introduce insurmountable error into the calculation. This is the reason that multi-group cross-section sets are usually identified by intended application (i.e., the HELIOS *reactor physics* library and the BUGLE-B7 *reactor pressure vessel fluence* library).

### 2.3.2 Discretization of the Spatial Domain

The finite volume-based spatial discretization procedure is outlined next. For this work, all deterministic transport calculations are performed in a Cartesian coordinate system. Information on the discretization of other coordinate systems is available in references such as [85, 88] but are not pursued in this work.

To begin, one can write the position vector $\boldsymbol{x}$ in its Cartesian components as $\boldsymbol{x} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$ with $0 \leq x \leq X$, $0 \leq y \leq Y$, and $0 \leq z \leq Z$. Correspondingly, the direction vector is written as

$$\begin{aligned} \boldsymbol{\Omega} &= \Omega_{\mathrm{x}}\boldsymbol{i} + \Omega_{\mathrm{y}}\boldsymbol{j} + \Omega_{\mathrm{z}}\boldsymbol{k} && \text{(2.37a)} \\ &= \mu\boldsymbol{i} + \sqrt{1-\mu^2}\left(\cos\gamma\boldsymbol{j} + \sin\gamma\boldsymbol{k}\right) && \text{(2.37b)} \\ &= \mu\boldsymbol{i} + \eta\boldsymbol{j} + \xi\boldsymbol{k}, && \text{(2.37c)} \end{aligned}$$

where $\mu$, $\gamma$, $\eta$, and $\xi$ are defined in Section 2.2.1. Therefore,

$$f\left(\boldsymbol{x}, \boldsymbol{\Omega}\right) = f\left(x, y, z, \mu, \gamma\right) = f\left(x, y, z, \mu, \eta, \xi\right), \tag{2.38}$$

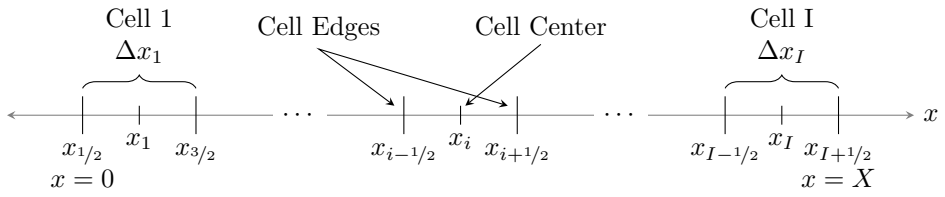
so the streaming term of Eq. (2.36) becomes

$$\begin{aligned} \boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f\left(\boldsymbol{x}, \boldsymbol{\Omega}\right) &= \left(\Omega_{\mathrm{x}}\frac{\partial}{\partial x} + \Omega_{\mathrm{y}}\frac{\partial}{\partial y} + \Omega_{\mathrm{z}}\frac{\partial}{\partial z}\right) f\left(x, y, z, \boldsymbol{\Omega}\right) \\ &= \left(\mu\frac{\partial}{\partial x} + \sqrt{1-\mu^2}\cos\gamma\frac{\partial}{\partial y} + \sqrt{1-\mu^2}\sin\gamma\frac{\partial}{\partial z}\right) f\left(x, y, z, \mu, \gamma\right) \\ &= \left(\mu\frac{\partial}{\partial x} + \eta\frac{\partial}{\partial y} + \xi\frac{\partial}{\partial z}\right) f\left(x, y, z, \mu, \eta, \xi\right) \\ &= \mu\frac{\partial f}{\partial x} + \eta\frac{\partial f}{\partial y} + \xi\frac{\partial f}{\partial z}. \end{aligned} \tag{2.39}$$

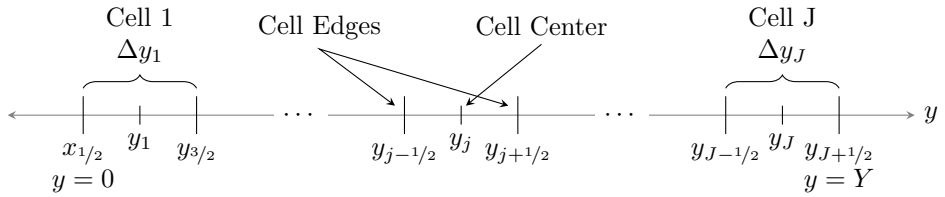
If the right-rectangular domain of the problem is subdivided into a Cartesian mesh with $I$, $J$, and $K$ spatial cells along the $\boldsymbol{i}$-, $\boldsymbol{j}$-, and $\boldsymbol{k}$-coordinate axes, respectively, then one can operate by

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \left(\cdot\right)$$

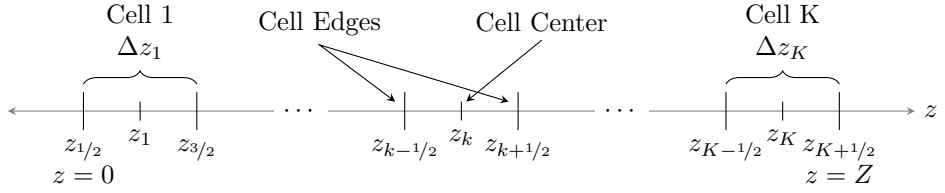
on the particle transport equation to yield a series of conservation equations giving the inflow, outflow, and balance of particles in each spatial mesh cell. The relationship between the continuous spatial domain and the spatial mesh is shown in Fig. 2.6. Performing this operation to Eq. (2.36) gives




**(a)** $x$-mesh Along $\boldsymbol{i}$-coordinate Axis




**(b)** $y$-mesh Along $\boldsymbol{j}$-coordinate Axis




**(c)** $z$-mesh Along $\boldsymbol{k}$-coordinate Axis

**Figure 2.6:** Cartesian Coordinate Axes Subdivided into $I, J, K$ Cells Containing Cell-edge and Cell-center Nodes.

$$\mu \int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \, \frac{\partial f}{\partial x} + \eta \int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \, \frac{\partial f}{\partial y} + \xi \int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \, \frac{\partial f}{\partial z}$$
$$+ \int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \, \Sigma_{\mathrm{t}}(\boldsymbol{x}) f(\boldsymbol{x}, \boldsymbol{\Omega})$$
$$= \int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \int_{4\pi} \mathrm{d}\Omega' \Sigma_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}) f(\boldsymbol{x}, \boldsymbol{\Omega}') + \int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \, Q(\boldsymbol{x}, \boldsymbol{\Omega}),$$
$$\boldsymbol{x} \in V, \, \boldsymbol{\Omega} \in 4\pi, \quad (2.40a)$$

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \, f(\boldsymbol{x}, \boldsymbol{\Omega}) = \int_{x_{i-1/2}}^{x_{i+1/2}} \mathrm{d}x \int_{y_{j-1/2}}^{y_{j+1/2}} \mathrm{d}y \int_{z_{k-1/2}}^{z_{k+1/2}} \mathrm{d}z \, f_{\mathrm{b}}(\boldsymbol{x}, \boldsymbol{\Omega}), \, \boldsymbol{x} \in \partial V, \, \boldsymbol{\Omega} \cdot \boldsymbol{n} < 0. \quad (2.40b)$$

Next, assume that $\Sigma_{\mathrm{t}}$, $\Sigma_{\mathrm{s}}$, and $Q$ is constant within each $i$,$j$,$k$-indexed spatial cell such that

$$\Sigma_{\mathrm{t}}(\boldsymbol{x}) = \Sigma_{\mathrm{t},i,j,k}, \tag{2.41a}$$

$$\Sigma_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}) = \Sigma_{\mathrm{s},i,j,k}(\boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}), \tag{2.41b}$$

$$Q(\boldsymbol{x}, \boldsymbol{\Omega}) = Q_{i,j,k}(\boldsymbol{\Omega}) \tag{2.41c}$$

and apply a spatial central difference to rewrite Eq. (2.40) as

$$\begin{aligned}\mu\left(\frac{f_{i+1/2,j,k}(\boldsymbol{\Omega}) - f_{i-1/2,j,k}(\boldsymbol{\Omega})}{\Delta x_i}\right) + \eta\left(\frac{f_{i,j+1/2,k}(\boldsymbol{\Omega}) - f_{i,j-1/2,k}(\boldsymbol{\Omega})}{\Delta y_j}\right) + \xi\left(\frac{f_{i,j,k+1/2}(\boldsymbol{\Omega}) - f_{i,j,k-1/2}(\boldsymbol{\Omega})}{\Delta z_k}\right) \\ + \Sigma_{\mathrm{t},i,j,k} f_{i,j,k}(\boldsymbol{\Omega}) \\ = \int_{4\pi} \mathrm{d}\Omega' \Sigma_{\mathrm{s},i,j,k}(\boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}) f_{i,j,k}(\boldsymbol{\Omega}') + Q_{i,j,k}(\boldsymbol{\Omega}), \\ i = 1, 2, \ldots, I,\ j = 1, 2, \ldots, J,\ k = 1, 2, \ldots, K,\ \boldsymbol{\Omega} \in 4\pi,\end{aligned} \tag{2.42a}$$

$$f_{1/2,j,k}(\boldsymbol{\Omega}) = f_{\mathrm{b}}(x = 0, y, z, \boldsymbol{\Omega}),\ \mu > 0,\ \boldsymbol{\Omega} \cdot \boldsymbol{n} < 0, \tag{2.42b}$$

$$f_{I+1/2,j,k}(\boldsymbol{\Omega}) = f_{\mathrm{b}}(x = X, y, z, \boldsymbol{\Omega}),\ \mu < 0,\ \boldsymbol{\Omega} \cdot \boldsymbol{n} < 0, \tag{2.42c}$$

$$f_{i,1/2,k}(\boldsymbol{\Omega}) = f_{\mathrm{b}}(x, y = 0, z, \boldsymbol{\Omega}),\ \eta > 0,\ \boldsymbol{\Omega} \cdot \boldsymbol{n} < 0, \tag{2.42d}$$

$$f_{i,J+1/2,k}(\boldsymbol{\Omega}) = f_{\mathrm{b}}(x, y = Y, z, \boldsymbol{\Omega}),\ \eta < 0,\ \boldsymbol{\Omega} \cdot \boldsymbol{n} < 0, \tag{2.42e}$$

$$f_{i,j,1/2}(\boldsymbol{\Omega}) = f_{\mathrm{b}}(x, y, z = 0, \boldsymbol{\Omega}),\ \xi > 0,\ \boldsymbol{\Omega} \cdot \boldsymbol{n} < 0, \tag{2.42f}$$

$$f_{i,j,K+1/2}(\boldsymbol{\Omega}) = f_{\mathrm{b}}(x, y, z = Z, \boldsymbol{\Omega}),\ \xi < 0,\ \boldsymbol{\Omega} \cdot \boldsymbol{n} < 0, \tag{2.42g}$$

where $f_{i\pm1/2,j,k}$, $f_{i,j\pm1/2,k}$, and $f_{i,j,k\pm1/2}$ are the cell-edge angular flux values, $\Delta x_i = x_{i+1/2} - x_{i-1/2}$, $\Delta y_j = y_{j+1/2} - y_{j-1/2}$, $\Delta z_k = z_{k+1/2} - z_{k-1/2}$, $f_{i,j,k}$ are the cell-center angular flux values, and Eqs. (2.42b)–(2.42g) are the six boundary condition equations for the cell-edge specified angular boundary flux values for all incoming directions for each of the six faces of the system boundary.

Thus, for the 3-D Cartesian system with interior volume $V = X \times Y \times Z$ one has $I \times J \times K$ cells that need to be solved [Eq.(2.42a)]. There are six boundary planes on $\partial V$ with a specified boundary source when $\boldsymbol{\Omega} \cdot \boldsymbol{n} < 0$ [Eqs. (2.42b)–(2.42g)]. For each cell edge on each boundary where $\boldsymbol{\Omega} \cdot \boldsymbol{n} < 0$, one may use the specified angular boundary source for the incoming cell-edge value of $f$ and compute the cell-center value of $f$ (accounting for any scattering and inhomogeneous sources). Then, one can compute the outgoing cell-edge value of $f$ and advance to the next cell in the direction of $\boldsymbol{\Omega}$. This can be done for all $\boldsymbol{\Omega}$ (where a finite number of directions are used as described in Section 2.3.3). This will sweep through all spatial cells and all directions to develop the global angular flux solution. How to compute the cell-center and outgoing cell-edge values is described next.

To solve the system of equations in Eq. (2.42), an auxiliary (i.e., closure) equation is necessary to relate the cell-edge and cell-center values of $f$ and close the system of equations. The most common is the linear-continuous "diamond-difference" approximation where $f_{i,j,k}$ is placed at the cell center and is related to each

set of the cell-edge values along the $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ directions as

$$f_{i,j,k}\left(\boldsymbol{\Omega}\right) = \frac{f_{i+1/2,j,k}\left(\boldsymbol{\Omega}\right) - f_{i-1/2,j,k}\left(\boldsymbol{\Omega}\right)}{2}, \tag{2.43a}$$

$$f_{i,j,k}\left(\boldsymbol{\Omega}\right) = \frac{f_{i,j+1/2,k}\left(\boldsymbol{\Omega}\right) - f_{i,j-1/2,k}\left(\boldsymbol{\Omega}\right)}{2}, \tag{2.43b}$$

$$f_{i,j,k}\left(\boldsymbol{\Omega}\right) = \frac{f_{i,j,k+1/2}\left(\boldsymbol{\Omega}\right) - f_{i,j,k-1/2}\left(\boldsymbol{\Omega}\right)}{2}. \tag{2.43c}$$

Other auxiliary equations relating the cell-edge and cell-center values (e.g., using the step and step-characteristic approaches) are possible but are not explored in this work.

By substituting Eq. (2.43) into Eq. (2.42) and rewriting the source term as

$$q_{i,j,k}\left(\boldsymbol{\Omega}\right) = \int\limits_{4\pi} \mathrm{d}\Omega' \Sigma_{\mathrm{s},i,j,k}\left(\boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}\right) f_{i,j,k}\left(\boldsymbol{\Omega}'\right) + Q_{i,j,k}\left(\boldsymbol{\Omega}\right), \tag{2.44}$$

one can solve for the cell-center flux as

$$f_{i,j,k}\left(\boldsymbol{\Omega}\right) = \begin{cases} \frac{\frac{2|\mu|}{\Delta x_i} f_{i-1/2,j,k}(\boldsymbol{\Omega}) + \frac{2|\eta|}{\Delta y_j} f_{i,j-1/2,k}(\boldsymbol{\Omega}) + \frac{2|\xi|}{\Delta z_k} f_{i,j-1/2,k}(\boldsymbol{\Omega}) + q_{i,j,k}(\boldsymbol{\Omega})}{\frac{2|\mu|}{\Delta x_i} + \frac{2|\eta|}{\Delta y_j} + \frac{2|\xi|}{\Delta z_k} + \Sigma_{\mathrm{t},i,j,k}} & \mu > 0,\ \eta > 0,\ \xi > 0 \\ \frac{\frac{2|\mu|}{\Delta x_i} f_{i+1/2,j,k}(\boldsymbol{\Omega}) + \frac{2|\eta|}{\Delta y_j} f_{i,j-1/2,k}(\boldsymbol{\Omega}) + \frac{2|\xi|}{\Delta z_k} f_{i,j-1/2,k}(\boldsymbol{\Omega}) + q_{i,j,k}(\boldsymbol{\Omega})}{\frac{2|\mu|}{\Delta x_i} + \frac{2|\eta|}{\Delta y_j} + \frac{2|\xi|}{\Delta z_k} + \Sigma_{\mathrm{t},i,j,k}} & \mu < 0,\ \eta > 0,\ \xi > 0 \\ \frac{\frac{2|\mu|}{\Delta x_i} f_{i-1/2,j,k}(\boldsymbol{\Omega}) + \frac{2|\eta|}{\Delta y_j} f_{i,j+1/2,k}(\boldsymbol{\Omega}) + \frac{2|\xi|}{\Delta z_k} f_{i,j-1/2,k}(\boldsymbol{\Omega}) + q_{i,j,k}(\boldsymbol{\Omega})}{\frac{2|\mu|}{\Delta x_i} + \frac{2|\eta|}{\Delta y_j} + \frac{2|\xi|}{\Delta z_k} + \Sigma_{\mathrm{t},i,j,k}} & \mu > 0,\ \eta < 0,\ \xi > 0 \\ \frac{\frac{2|\mu|}{\Delta x_i} f_{i+1/2,j,k}(\boldsymbol{\Omega}) + \frac{2|\eta|}{\Delta y_j} f_{i,j+1/2,k}(\boldsymbol{\Omega}) + \frac{2|\xi|}{\Delta z_k} f_{i,j-1/2,k}(\boldsymbol{\Omega}) + q_{i,j,k}(\boldsymbol{\Omega})}{\frac{2|\mu|}{\Delta x_i} + \frac{2|\eta|}{\Delta y_j} + \frac{2|\xi|}{\Delta z_k} + \Sigma_{\mathrm{t},i,j,k}} & \mu < 0,\ \eta < 0,\ \xi > 0 \\ \frac{\frac{2|\mu|}{\Delta x_i} f_{i-1/2,j,k}(\boldsymbol{\Omega}) + \frac{2|\eta|}{\Delta y_j} f_{i,j-1/2,k}(\boldsymbol{\Omega}) + \frac{2|\xi|}{\Delta z_k} f_{i,j+1/2,k}(\boldsymbol{\Omega}) + q_{i,j,k}(\boldsymbol{\Omega})}{\frac{2|\mu|}{\Delta x_i} + \frac{2|\eta|}{\Delta y_j} + \frac{2|\xi|}{\Delta z_k} + \Sigma_{\mathrm{t},i,j,k}} & \mu > 0,\ \eta > 0,\ \xi < 0 \\ \frac{\frac{2|\mu|}{\Delta x_i} f_{i+1/2,j,k}(\boldsymbol{\Omega}) + \frac{2|\eta|}{\Delta y_j} f_{i,j-1/2,k}(\boldsymbol{\Omega}) + \frac{2|\xi|}{\Delta z_k} f_{i,j+1/2,k}(\boldsymbol{\Omega}) + q_{i,j,k}(\boldsymbol{\Omega})}{\frac{2|\mu|}{\Delta x_i} + \frac{2|\eta|}{\Delta y_j} + \frac{2|\xi|}{\Delta z_k} + \Sigma_{\mathrm{t},i,j,k}} & \mu < 0,\ \eta > 0,\ \xi < 0 \\ \frac{\frac{2|\mu|}{\Delta x_i} f_{i-1/2,j,k}(\boldsymbol{\Omega}) + \frac{2|\eta|}{\Delta y_j} f_{i,j+1/2,k}(\boldsymbol{\Omega}) + \frac{2|\xi|}{\Delta z_k} f_{i,j+1/2,k}(\boldsymbol{\Omega}) + q_{i,j,k}(\boldsymbol{\Omega})}{\frac{2|\mu|}{\Delta x_i} + \frac{2|\eta|}{\Delta y_j} + \frac{2|\xi|}{\Delta z_k} + \Sigma_{\mathrm{t},i,j,k}} & \mu > 0,\ \eta < 0,\ \xi < 0 \\ \frac{\frac{2|\mu|}{\Delta x_i} f_{i+1/2,j,k}(\boldsymbol{\Omega}) + \frac{2|\eta|}{\Delta y_j} f_{i,j+1/2,k}(\boldsymbol{\Omega}) + \frac{2|\xi|}{\Delta z_k} f_{i,j+1/2,k}(\boldsymbol{\Omega}) + q_{i,j,k}(\boldsymbol{\Omega})}{\frac{2|\mu|}{\Delta x_i} + \frac{2|\eta|}{\Delta y_j} + \frac{2|\xi|}{\Delta z_k} + \Sigma_{\mathrm{t},i,j,k}} & \mu < 0,\ \eta < 0,\ \xi < 0 \end{cases}, \tag{2.45}$$

so the outgoing cell-edge values are

$$f_{i\pm 1/2,j,k}\left(\boldsymbol{\Omega}\right) = 2 f_{i,j,k}\left(\boldsymbol{\Omega}\right) - f_{i\mp 1/2,j,k}\left(\boldsymbol{\Omega}\right), \tag{2.46a}$$

$$f_{i,j\pm 1/2,k}\left(\boldsymbol{\Omega}\right) = 2 f_{i,j,k}\left(\boldsymbol{\Omega}\right) - f_{i,j\mp 1/2,k}\left(\boldsymbol{\Omega}\right), \tag{2.46b}$$

$$f_{i,j,k\pm 1/2}\left(\boldsymbol{\Omega}\right) = 2 f_{i,j,k}\left(\boldsymbol{\Omega}\right) - f_{i,j,k\mp 1/2}\left(\boldsymbol{\Omega}\right). \tag{2.46c}$$

Because of errors resulting from the diamond difference approximation, it is possible for the outgoing cell-edge values of $f_{i\pm 1/2,j,k}\left(\boldsymbol{\Omega}\right)$, $f_{i,j\pm 1/2,k}\left(\boldsymbol{\Omega}\right)$, and/or $f_{i,j,k\pm 1/2}\left(\boldsymbol{\Omega}\right)$ to be negative, which is non-physical. If negativity occurs, there are several ways to address it; however, the approach taken in this work is to set the outgoing

cell-edge value to zero and to re-average the cell-average flux value as

$$f_{i,j,k}(\mathbf{\Omega}) = \frac{f_{i+1/2,j,k}(\mathbf{\Omega}) + f_{i-1/2,j,k}(\mathbf{\Omega}) + f_{i,j+1/2,k}(\mathbf{\Omega}) + f_{i,j-1/2,k}(\mathbf{\Omega}) + f_{i,j,k+1/2}(\mathbf{\Omega}) f_{i,j,k-1/2}(\mathbf{\Omega})}{6}. \quad (2.47)$$

Equations (2.42)–(2.46) can be reduced from 3-D to 1-D by operating on the equations with $\int_0^{2\pi} \mathrm{d}\gamma\,(\cdot)$ and assuming a homogeneous material distributions and sources in the $y$-$z$ plane (equivalent to an infinite extent in the $y$ and $z$ directions). As a result, one obtains the 1-D forms of Eqs. (2.42)–(2.46):

$$\mu\left(\frac{f_{i+1/2}(\mu) - f_{i-1/2}(\mu)}{\Delta x_i}\right) + \Sigma_{\mathrm{t},i} f_i(\mu) = \int_{-1}^{1} \mathrm{d}\mu' \Sigma_{\mathrm{s},i}(\mu,\mu') f_i(\mu') + Q_i(\mu),$$
$$i = 1,2,\ldots,I,\ -1 \le \mu \le 1, \quad (2.48a)$$

$$f_{1/2}(\mu) = f_{\mathrm{b}}(x = 0,\mu)\ ,\ \mu > 0, \quad (2.48b)$$

$$f_{I+1/2}(\mu) = f_{\mathrm{b}}(x = X,\mu)\ ,\ \mu < 0, \quad (2.48c)$$

$$f_i(\mu) = \frac{f_{i+1/2}(\mu) - f_{i-1/2}(\mu)}{2}, \quad (2.49a)$$

$$q_i(\mu) = \int_{4\pi} \mathrm{d}\mu' \Sigma_{\mathrm{s},i}(\mu) f_i(\mu') + Q_i(\mu), \quad (2.50)$$

$$f_i(\mu) = \begin{cases} \frac{\frac{2|\mu|}{\Delta x_i} f_{i-1/2}(\mu) + q_i(\mu)}{\frac{2|\mu|}{\Delta x_i} + \Sigma_{\mathrm{t},i}} & \mu > 0 \\ \frac{\frac{2|\mu|}{\Delta x_i} f_{i+1/2}(\mu) + q_i(\mu)}{\frac{2|\mu|}{\Delta x_i} + \Sigma_{\mathrm{t},i}} & \mu < 0 \end{cases}, \quad (2.51)$$

$$f_{i\pm1/2}(\mu) = 2 f_i(\mu) - f_{i\mp1/2}(\mu). \quad (2.52)$$

The numerical solution method described in this section requires sweeping through the mesh along $\mathbf{\Omega}$ and progressively solving for cell-center and outgoing cell-edge values. The same is true for the 1-D equations, where the mesh is swept through along all directions of $\mu$ starting from the boundary at $x = 0$ for $\mu > 0$ and starting at $x = X$ for $\mu < 0$. As such, the next section will discuss how to discretize $\mathbf{\Omega}$.

### 2.3.3 Discretization of the Angular Domain

The numerical solution algorithm described in the previous section requires sweeping through the mesh along $\mathbf{\Omega}$ to solve for cell-center and outgoing cell-edge values. In addition, integration over $\mathbf{\Omega}$ is required to compute the scattering source integral. Because of these requirements the discrete ordinates ($S_N$) method [95] is used

in this work to discretize the unit sphere into $N$ directions indicated by their individual direction cosines $\mu_n$, $\eta_n$, and $\xi_n$. Each direction has an associated quadrature weight $\omega_n$ used when integrating a function over the unit sphere (such as when computing the scattering source integral). As such, a particle may only travel along a given discrete direction

$$\begin{aligned}\boldsymbol{\Omega}_n &= \Omega_{n,\mathrm{x}}\boldsymbol{i} + \Omega_{n,\mathrm{y}}\boldsymbol{j} + \Omega_{n,\mathrm{z}}\boldsymbol{k} && \text{(2.53a)}\\ &= \mu_n\boldsymbol{i} + \left(\sqrt{1-\mu_n^2}\cos\gamma_n\right)\boldsymbol{j} + \left(\sqrt{1-\mu_n^2}\sin\gamma_n\right)\boldsymbol{k} && \text{(2.53b)}\\ &= \mu_n\boldsymbol{i} + \eta_n\boldsymbol{j} + \xi_n\boldsymbol{k}. && \text{(2.53c)}\end{aligned}$$

The combination of direction cosines and associated weights is referred to as a quadrature set. See Fig. 2.7 for an example of octant-symmetric (called level-symmetric) quadrature sets of varying levels of refinement in an octant.

The development of quadrature sets is ongoing and a quadrature set designer may use different metrics to evaluate a quadrature set's suitability for his or her purposes. Common criteria include the conditions that $\sum_n \omega_n$ be a convenient quantity (often 1, 2, or $4\pi$ to be consistent with the other terms in the radiation transport equation) and that the quadrature set be able to exactly compute a specified number of spherical harmonics. However, these requirements can be relaxed in favor of others such as grouping directions around a particular direction of interest (a biased quadrature set) or having an ordinate oriented in a specific direction.

The two quadrature sets used in this work for 1-D and 2-D calculations are the Gauss-Legendre and Triangular Gauss-Chebyshev quadrature sets, respectively. More information on these quadrature sets is available in Appendix C. Other quadrature sets explored in this work, but ultimately not used, are Jarrell's linear-discontinuous finite element (LDFE) quadrature set [96, 97] and Abu-Shamay's quadruple-range (QR) quadrature set [98, 99]. The 1-D Gauss-Legendre $S_{32}$ quadrature points used in this work are illustrated in Fig. 2.8. The $S_{32}$ and $S_{64}$ Triangular Gauss-Chebyshev quadrature sets used in this work are illustrated in Fig. 2.9 for a single octant. Note that for the 2-D $x$-$y$ calculations in this work, the polar direction is taken as $\xi$ rather than $\mu$ so

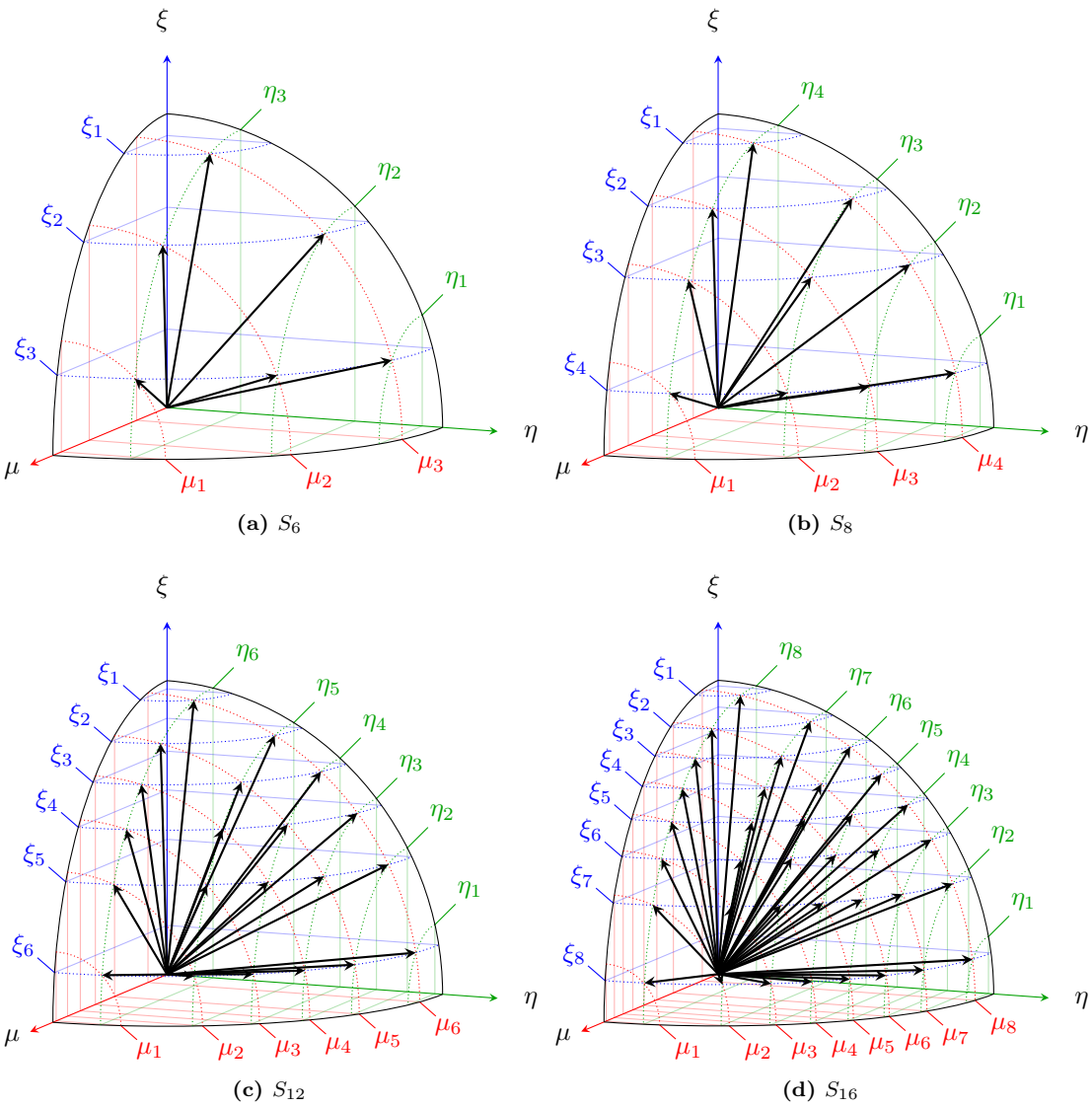
$$\begin{aligned}\boldsymbol{\Omega}_n &= \Omega_{n,\mathrm{x}}\boldsymbol{i} + \Omega_{n,\mathrm{y}}\boldsymbol{j} + \Omega_{n,\mathrm{z}}\boldsymbol{k} && \text{(2.54a)}\\ &= \left(\sqrt{1-\xi_n^2}\cos\gamma_n\right)\boldsymbol{i} + \left(\sqrt{1-\xi_n^2}\sin\gamma_n\right)\boldsymbol{j} + \xi_n\boldsymbol{k} && \text{(2.54b)}\\ &= \mu_n\boldsymbol{i} + \eta_n\boldsymbol{j} + \xi_n\boldsymbol{k}. && \text{(2.54c)}\end{aligned}$$

Using a particular quadrature set, an integral over angular function $f_{i,j,k}(\boldsymbol{\Omega})$ can be computed as

$$\int_{4\pi} \mathrm{d}\Omega f_{i,j,k}(\boldsymbol{\Omega}) = \lim_{N\to\infty}\sum_{n=1}^{N}\omega_n f_{i,j,k}(\boldsymbol{\Omega}_n) = \lim_{N\to\infty}\sum_{n=1}^{N}\omega_n f_{i,j,k,n} \approx \sum_{n=1}^{N}\omega_n f_{i,j,k,n}, \qquad (2.55)$$

so by truncating $N$ an angular integral is approximated by a simple summation.

To see how quadrature sets are used to integrate the scattering source, one can consider the 3-D multi-group




**(a)** $S_6$ **(b)** $S_8$

**(c)** $S_{12}$ **(d)** $S_{16}$

**Figure 2.7:** Variously Refined Level-Symmetric Quadrature Sets Shown in One Octant

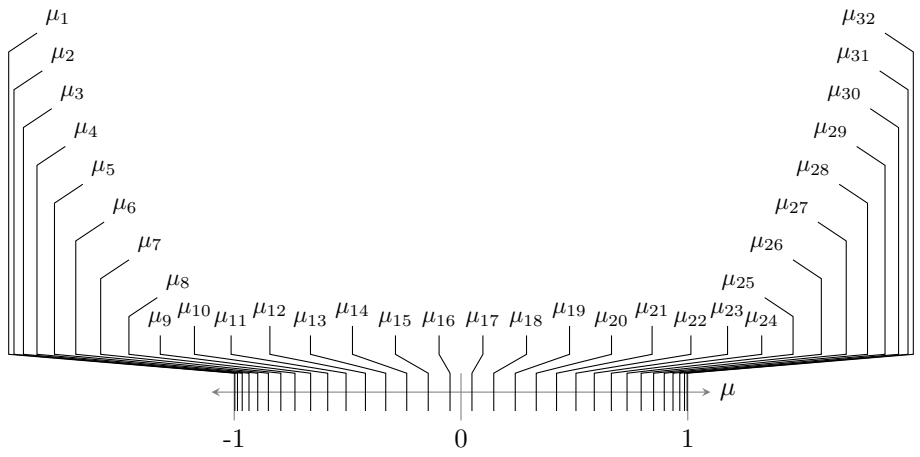
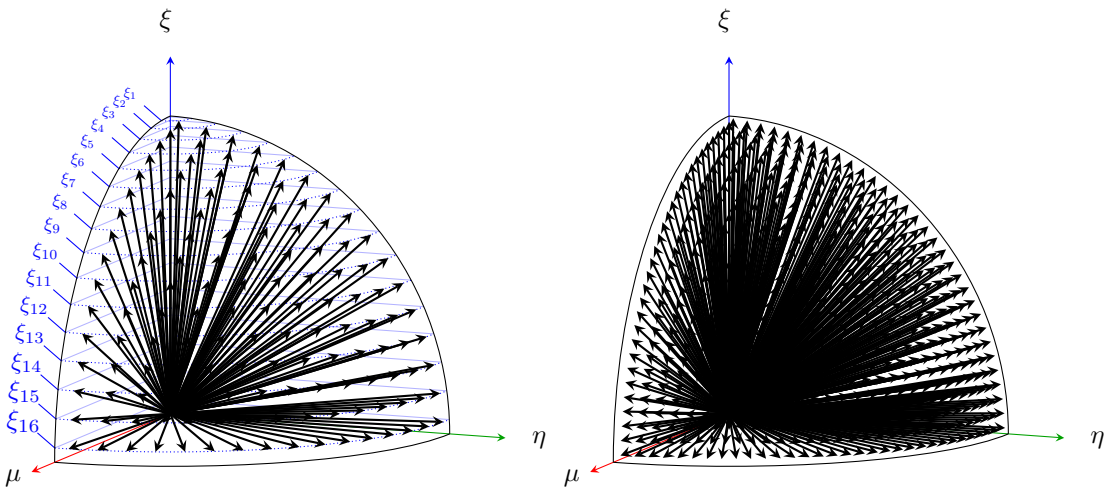



**Figure 2.8:** $S_{32}$ Gauss-Legendre Quadrature Points




**(a)** $S_{32}$ **(b)** $S_{64}$

**Figure 2.9:** Triangular Gauss-Chebyshev Quadrature Sets Shown in One Octant

case discretized onto a Cartesian mesh where the scattering source is [Eq. (2.35)]

$$Q_{\mathrm{s},i,j,k,g}\left(\mathbf{\Omega}\right)=\sum_{g'=1}^{G}\int_{4\pi}\mathrm{d}\Omega'\Sigma_{\mathrm{s},i,j,k,g'\rightarrow g}\left(\mathbf{\Omega}'\cdot\mathbf{\Omega}\right)f_{i,j,k,g'}\left(\mathbf{\Omega}'\right), \tag{2.56}$$

which indicates that the scattering source at in cell indexed $i,j,k$, direction $\mathbf{\Omega}$, for group $g$ consists of all particles that collided in cell indexed $i,j,k$ (from all incoming directions $\mathbf{\Omega}'$ and energy groups $g'$) and emerged from the collision in direction $\mathrm{d}\Omega$ about $\mathbf{\Omega}$ and transitioned to (or stayed in) group $g$. The scattering cross section can then be expanded in Legendre polynomials as

$$\Sigma_{\mathrm{s},i,j,k,g'\rightarrow g}\left(\mu_0\right)=\lim_{L\rightarrow\infty}\sum_{l=0}^{L}\frac{2l+1}{4\pi}\Sigma_{\mathrm{s},i,j,k,g'\rightarrow g,l}P_l\left(\mu_0\right) \tag{2.57}$$

where $\mathbf{\Omega}'\cdot\mathbf{\Omega}=\cos\theta=\mu_0$ as shown in Fig. 2.3 and

$$\Sigma_{\mathrm{s},i,j,k,g'\rightarrow g,l}=2\pi\int_{-1}^{1}\mathrm{d}\mu_0\Sigma_{\mathrm{s},i,j,k,g'\rightarrow g}\left(\mu_0\right)P_l\left(\mu_0\right) \tag{2.58}$$

and $P_l\left(\mu_0\right)$ is the $l^{\mathrm{th}}$ Legendre polynomial. For this work emergence from scattering is treated isotropically for simplicity (i.e., $L=0$). Next, expand $f_{i,j,k,g'}$ in terms of spherical harmonic functions (related to Legendre polynomials through the Addition Theorem) to obtain

$$f_{i,j,k,g'}\left(\mu',\gamma'\right)=\lim_{L\rightarrow\infty}\sum_{l=0}^{L}\sum_{r=-l}^{l}f_{i,j,k,g',l,r}Y_{l,r}\left(\mu',\gamma'\right), \tag{2.59}$$

$$f_{i,j,k,g',l,r}\left(\boldsymbol{x}\right)=\int_{2\pi}\mathrm{d}\gamma'\int_{-1}^{1}\mathrm{d}\mu'f_{i,j,k,g'}\left(\mu',\gamma'\right)Y_{l,r}^{*}\left(\mu',\gamma'\right), \tag{2.60}$$

where $Y_{l,r}$ and $Y_{l,r}^{*}$ are the real and complex conjugate spherical harmonics functions, respectively. Following prior work [87, Section II.D.1] by rewriting Eq. (2.56) using real spherical harmonics throughout, the scattering source becomes

$$Q_{\mathrm{s},i,j,k,g}\left(\mathbf{\Omega}\right)=\sum_{g'=1}^{G}\sum_{l=0}^{L}\Sigma_{\mathrm{s},i,j,k,g'\rightarrow g,l}\sum_{r=-l}^{r}Y_{l,r}\left(\mu,\gamma\right)f_{i,j,k,g',l,r}. \tag{2.61}$$

Because it is more computationally convenient to work with real spherical harmonic functions as opposed to complex-valued ones, one can introduce the simply indexed form for the real spherical harmonic function $R_r\left(\mu,\gamma\right)$ (and re-using the $r$ subscript) from [87, Appendix A]. Thus, one can express the scattering source as

$$Q_{\mathrm{s},i,j,k,g}\left(\mathbf{\Omega}\right)=\sum_{g'=1}^{G}\sum_{r=1}^{N_{r,\mathrm{s}}}\left(2l_r+1\right)\Sigma_{\mathrm{s},i,j,k,g'\rightarrow g,l_r}R_r\left(\mu,\gamma\right)f_{i,j,k,g',r} \tag{2.62}$$

where $N_{r,\mathrm{s}}$ is the maximum number of moments used in the scattering expansion and

$$f_{i,j,k,g',r}=\int_{2\pi}\mathrm{d}\gamma'\int_{-1}^{1}\mathrm{d}\mu'f_{i,j,k}\left(\mu',\gamma'\right)R_r\left(\mu',\gamma'\right) \tag{2.63}$$

using the same $r$ subscript as $R_r(\mu,\gamma)$. Furthermore,

$$l_r = \begin{cases} r-1 & \text{1-D Cartesian} \\ \left\lceil \frac{\sqrt{9-4(2-2r)}-3}{2} \right\rceil & \text{2-D Cartesian} \\ \lceil \sqrt{r} \rceil - 1 & \text{3-D Cartesian} \end{cases}, \tag{2.64}$$

where $\lceil \cdot \rceil$ indicates the integer ceiling function. Some example values of $l_r$ and $R_r(\mu,\gamma)$ for $L=3$ for 1-, 2-, and 3-D Cartesian geometry are shown in Table 2.1.

Thus, the scattering source in Eq. (2.62) with the $S_N$ approximation applied becomes

$$Q_{\mathrm{s},i,j,k,g}(\mathbf{\Omega}_n) = Q_{\mathrm{s},i,j,k,n,g} = \sum_{g'=1}^{G} \sum_{r=1}^{N_{r,\mathrm{s}}} (2l_r+1)\, \Sigma_{\mathrm{s},i,j,k,g'\to g,l_r} R_r(\mu_n,\gamma_n)\, f_{i,j,k,g',r} \tag{2.65}$$

and

$$\Sigma_{\mathrm{s},i,j,k,g'\to g,l} = \sum_{n=1}^{N} \omega_n P_l(\mu_n)\, \Sigma_{\mathrm{s},i,j,k,g'\to g}(\mu_n), \tag{2.66}$$

$$f_{i,j,k,g',r} = \sum_{n=1}^{N} \omega_n f_{i,j,k}(\mu_n,\gamma_n)\, Y^*_{l,r}(\mu_n,\gamma_n) \tag{2.67a}$$

$$= \sum_{n=1}^{N} \omega_n f_{i,j,k}(\mu_n,\gamma_n)\, R_r(\mu_n,\gamma_n). \tag{2.67b}$$

As with the scattering source, the inhomogeneous source (if any) can be expanded using spherical harmonic functions as

$$Q_{i,j,k,g}(\mathbf{\Omega}) = \sum_{r=1}^{N_{r,\mathrm{Q}}} (2l_r+1)\, R_r(\mu,\gamma)\, Q_{i,j,k,g,r}, \tag{2.68a}$$

$$Q_{i,j,k,g,r} = \int_{2\pi} \mathrm{d}\gamma' \int_{-1}^{1} \mathrm{d}\mu' Q_{i,j,k,g}(\mu',\gamma')\, Y^*_{l,r}(\mu',\gamma'), \tag{2.68b}$$

where $N_{r,\mathrm{Q}}$ is the maximum number of moments used in the inhomogeneous source expansion (which may be different than $N_{r,\mathrm{s}}$). Correspondingly, the $S_N$ representation is

$$Q_{i,j,k,n,g} = \sum_{r=1}^{N_{r,\mathrm{Q}}} (2l_r+1)\, R_r(\mu_n,\gamma_n)\, Q_{i,j,k,g,r}, \tag{2.69a}$$

$$Q_{i,j,k,g,r} = \sum_{n=1}^{N} \omega_n Q_{i,j,k,n,g} R_r(\mu_n,\gamma_n). \tag{2.69b}$$

**Table 2.1:** Values of $l_r$ and $R_r(\mu,\gamma)$ for 1-, 2-, and 3-D Cartesian Geometries for Legendre Expansion Order $L=3$

| $r$ | $l_r$ | | | $R_r(\mu,\gamma)$ | | |
|---|---|---|---|---|---|---|
| | 1-D | 2-D | 3-D | 1-D | 2-D | 3-D |
| 1 | 0 | 0 | 0 | $P_0(\mu)$ | $P_0(\mu)$ | $P_0(\mu)$ |
| 2 | 1 | 1 | 1 | $P_1(\mu)$ | $P_1(\mu)$ | $P_1(\mu)$ |
| 3 | 2 | 1 | 1 | $P_2(\mu)$ | $P_1^1(\mu)\cos\gamma$ | $P_1^1(\mu)\cos\gamma$ |
| 4 | 3 | 2 | 1 | $P_3(\mu)$ | $P_2^2(\mu)$ | $P_1^1(\mu)\sin\gamma$ |
| 5 | — | 2 | 2 | — | $\frac{\sqrt{3}}{3}P_2^1(\mu)\cos\gamma$ | $P_2(\mu)$ |
| 6 | — | 2 | 2 | — | $\frac{\sqrt{3}}{6}P_2^2(\mu)\cos 2\gamma$ | $\frac{\sqrt{3}}{3}P_2^1(\mu)\cos\gamma$ |
| 7 | — | 3 | 2 | — | $P_3(\mu)$ | $\frac{\sqrt{3}}{3}P_2^1(\mu)\sin\gamma$ |
| 8 | — | 3 | 2 | — | $\frac{\sqrt{6}}{6}P_3^1(\mu)\cos\gamma$ | $\frac{\sqrt{3}}{6}P_2^2(\mu)\cos 2\gamma$ |
| 9 | — | 3 | 2 | — | $\frac{\sqrt{15}}{30}P_3^2(\mu)\cos 2\gamma$ | $\frac{\sqrt{3}}{6}P_2^2(\mu)\sin 2\gamma$ |
| 10 | — | 3 | 3 | — | $\frac{\sqrt{10}}{60}P_3^3(\mu)\cos 3\gamma$ | $P_3(\mu)$ |
| 11 | — | — | 3 | — | — | $\frac{\sqrt{6}}{6}P_3^1(\mu)\cos\gamma$ |
| 12 | — | — | 3 | — | — | $\frac{\sqrt{6}}{6}P_3^1(\mu)\sin\gamma$ |
| 13 | — | — | 3 | — | — | $\frac{\sqrt{15}}{30}P_3^2(\mu)\cos 2\gamma$ |
| 14 | — | — | 3 | — | — | $\frac{\sqrt{15}}{30}P_3^2(\mu)\sin 2\gamma$ |
| 15 | — | — | 3 | — | — | $\frac{\sqrt{10}}{60}P_3^3(\mu)\cos 3\gamma$ |
| 16 | — | — | 3 | — | — | $\frac{\sqrt{10}}{60}P_3^3(\mu)\sin 3\gamma$ |

Note: $P_l$ are Legendre polynomials; $P_l^{l'}$ are associated Legendre polynomials.

See [87] for more information.

### 2.3.4 Overview of Radiation Transport Equation Solution Algorithm

Algorithms for solving the radiation transport equation for this work are described in Chapter 5. However, an overview of the general method is given here. Following from the three techniques used to discretize the energy, spatial, and angular phase spaces, the multi-group 3-D Cartesian $S_N$ radiation transport equation is

$$\mu_n \left[ \frac{f_{i+1/2,j,k,n,g} - f_{i-1/2,j,k,n,g}}{\Delta x_i} \right] + \eta_n \left[ \frac{f_{i,j+1/2,k,n,g} - f_{i,j-1/2,k,n,g}}{\Delta y_j} \right] + \xi_n \left[ \frac{f_{i,j,k+1/2,n,g} - f_{i,j,k-1/2,n,g}}{\Delta z_k} \right] + \Sigma_{\mathrm{t},i,j,k,g} f_{i,j,k,n,g}$$
$$= \sum_{g'=1}^{G} \sum_{r=1}^{N_{r,\mathrm{s}}} (2l_r + 1) \Sigma_{\mathrm{s},i,j,k,g' \to g,l_r} R_r(\mu_n, \gamma_n) f_{i,j,k,g',r} + \sum_{r=1}^{N_{r,\mathrm{Q}}} (2l_r + 1) R_r(\mu_n, \gamma_n) Q_{i,j,k,g,r}, \quad (2.70a)$$

$$\Sigma_{\mathrm{s},i,j,k,g' \to g,l} = \sum_{n=1}^{N} \omega_n P_l(\mu_n) \Sigma_{\mathrm{s},i,j,k,n,g' \to g}, \quad (2.70b)$$

$$f_{i,j,k,g',r} = \sum_{n=1}^{N} \omega_n f_{i,j,k,n,g'} R_r(\mu_n, \gamma_n), \quad (2.70c)$$

$$Q_{i,j,k,g,r} = \sum_{n=1}^{N} \omega_n Q_{i,j,k,n,g} R_r(\mu_n, \gamma_n), \quad (2.70d)$$

with specified boundary sources on the edge of the system where $i = 1/2, I+1/2$; $j = 1/2, J+1/2$; $k = 1/2, K+1/2$ and $\mathbf{\Omega}_n$ is directed into the system [Eqs. (2.42b)–(2.42g)]. To solve Eq. (2.70), the "scattering source iteration" algorithm is used where in the zeroth iteration, $u = 0$, an initial guess is made for $f_{i,j,k,n,g,u}$ (e.g., $f_{i,j,k,n,g,u=0} = 0$) to compute the scattering source, $Q_{\mathrm{s},i,j,k,n,g,u=0}$, and the steps outlined in Algorithm 2.1 are followed.

Following Algorithm 2.1, the (scattering) source iteration computes the uncollided particle profile when $u = 1$, then the first-collided particle profile when $u = 2$, and so on. In this way, the scattering source is accumulated collision-by-collision until its effect on the particle profile is less than that of the convergence criterion. Conversely, one can interrogate the particle profile by iteration to understand how the profile changes according to the number of collisions. This scattering-source iteration process is equivalent to performing a Neumann expansion on the scattering operator in the integral formulation of the radiation transport equation described in Section 2.2.4.

The convergence criterion used to terminate Algorithm 2.1 varies from code to code. For this work, the calculation is converged when the relative change in the response of interest (e.g., solution to the HSMEs and FTE) is less than some user-specified threshold (e.g., $10^{-3}$).

## 2.4 Monte Carlo Radiation Transport

This section describes how the Monte Carlo method is used to simulate the transport of neutral particles. It begins with a discussion of how the overall Monte Carlo process is used to perform the integrations described in Section 2.2.4. It continues with a description of how non-analog Monte Carlo is performed using variance

**Algorithm 2.1:** Scattering Source Iterations without Upscatter

```
converged ← False; u ← 0; f_{i,j,k,n,g,u} ← 0; Q_{s,i,j,k,n,g,u} ← 0
while not converged do
    u ← u + 1
    forall g do
        Q_{s,i,j,k,n,g,u} ← Eq. (2.65);                          // Update scattering source
        forall n do
            forall i, j, k do
                if Ω_n · n < 0 and on problem boundary then
                    Set incoming cell-edge value to boundary source
                else
                    Set incoming cell-edge value to previous cell's outgoing cell-edge value
                q_{i,j,k,n,g,u} ← Eq. (2.44);                    // Total cell source
                f_{i,j,k,n,g,u} ← Eq. (2.45);                    // Cell-center flux
                f_{i,j,k,g,r} ← Eq. (2.70c);                     // Flux-moment
                Calculate outgoing cell-edge value with Eq. (2.46)
                Advance to next cell along Ω_n
    if convergence criterion < ε then
        converged ← True
```

reduction by introducing the idea of statistical weight associated with a Monte Carlo particle, which is used to preserve fairness. It continues with a description of how quantities of interest are characterized through tallies and the statistical properties of tallies. The section concludes with a description of the two variance-reduction techniques used in this work. First, importance splitting and rouletting is described to introduce the reader to a common variance-reduction technique. Then, DXTRAN is described along with the simplifications made to perform this work.

### 2.4.1 Monte Carlo Transport of Neutral Particles

Unlike deterministic techniques, which begin with the particle transport equation and perform some form of discretization, Monte Carlo radiation transport calculations attempt to simulate average physical system behavior caused by particle interaction using statistical sampling. Effectively, Monte Carlo calculations can be seen as performing the integrations described by the integral form of the radiation transport equation in Section 2.2.4. Because a physical particle typically interacts with no memory of its earlier state(s), a particle can be considered memoryless. Physical particles thus satisfy the Markov property where complete knowledge of the current phase-space state is sufficient information to completely describe the PDFs for the next phase-space state. Therefore, physical particle random walks (and the corresponding Monte Carlo simulation random walks) are Markov processes.

This work focuses on history-based Monte Carlo, where many computational particles are executed independently and their behavior with respect to a quantity of interest (flux, reaction rate, escape probability, etc.) is accumulated and then averaged at the conclusion of the calculation. A computational particle is initialized according to the parameters of the forward source term (position, direction, and energy as $\boldsymbol{x}$, $\boldsymbol{\Omega}$, and $E$, respectively) . If the source parameters are defined according to PDFs, the PDFs are sampled to obtain the particle's initial state. Random sampling is then performed to determine the next state of the

particle following source emission.

The computational particle is then tracked through the geometry by determining the randomly sampled distance to collision or the distance to the closest geometry surface (whichever is closer). If the particle collides, the type of collision is sampled, the particle emerges accordingly, and continues until its next event unless the collision was an absorption. If a surface is crossed, surface-crossing mechanics are performed to prepare the particle for its next flight in a new cell unless the surface is a system boundary or the particle escapes the system. If the particle is absorbed or escapes, it is "killed" and no longer tracked (it can also be killed as a result of a variance-reduction process). Otherwise, the new state becomes the current state and sampling takes place again.

As a particle evolves it has no memory of what has occurred in the past, but to reach the current state the particle implicitly carries information about prior states. As such, the computational particle is often referred to as a history because it is a set of random trajectories describing evolution, from creation by the source through removal by some means, of a particle and any subsequent particles produced, e.g., through variance reduction methods. There is an important distinction here: a history has an associated tally score (for each tally) that is contributed to as the history's source particle *and any progeny* evolve.

To enable the aforementioned random sampling to occur, a pseudorandom number sequence is required. A pseudorandom number sequence is predictable, reproducible, but contains a sequence of values that appear distributed randomly and without correlation. This permits Monte Carlo calculations to be repeated but still captures the stochastic nature of the calculation without bias or correlation. For this work, as is typical, the numbers in a pseudorandom number sequence are uniformly distributed between zero and one, the unit interval $U(0,1)$. The pseudorandom number sequence for this work is produced by the MCNP linear congruential generator [100, 101]. Each random number used in the upcoming discussion is represented as $\zeta \sim U(0,1)$.

### 2.4.2 Tallying Quantities of Interest

As noted previously, a history is a single statistical sample that may contribute to an event of interest at some point in its evolution by way of a tally (where the tally is configured to record the event of interest such as whether a particle leaks out of the system, contributes path length in a volume, etc.). Thus, a history contributes a score to the tally when it or any progeny have satisfied the condition of the tally. Ultimately, the (unknown) score PDF for the tally is what the particles are sampling to yield the quantity of interest (the mean of the score PDF). As the histories score, the scores can be collected to construct an empirical score PDF that, in the limit of infinite histories, converges to the actual score PDF.

Several types of tallies are described next. For each tally, the conditions to contribute to the tally and the subsequent increase in history score for the tally is given. The collection of history scores are then processed as described in Section 2.4.3. Generally, when contributions to tallies are described they are based on the computational particle weight, which is described in more detail in Section 2.4.4.

#### 2.4.2.1 Current Tally

The total current across a surface of interest $A$ defined with normal direction $\boldsymbol{n}$ can be calculated as

$$J = \int \mathrm{d}A \int \mathrm{d}\Omega \int \mathrm{d}E \left|\boldsymbol{\Omega} \cdot \boldsymbol{n}\right| f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right), \tag{2.71}$$

where the integrals over direction and energy are over all directions and energies or just the direction and energy ranges of interest. If the surface $A$ is on the boundary of the system, the current tally can be used to compute the leakage probability. Regardless, the total current (for all directions and energies) corresponds to the fraction of source particles crossing $A$ (possibly greater than one) so the condition to contribute to a current tally on $A$ is satisfied when a computational particle crosses $A$. Each time a particle crosses, it contributes its weight, $w$, to the history score, $s_h$.

#### 2.4.2.2 Collision Tallies

Collision tallies can be used in several ways. First, the total reaction rate in a volume can be used to characterize the frequency of certain types of reactions of interest (e.g., absorption or scattering). Second, the total collision rate can be used to compute the total scalar flux in a volume.

There are two methods available to calculate the total reaction rate for reactions of interest in a volume. The first approach is to tally the reaction of interest directly where the condition to contribute to the tally is satisfied each time the particular collision of interest occurs. Each time this occurs, the particle contributes its weight, $w$, to the history score, $s_h$. For low-probability collisions, this can require many histories to be followed. Instead of satisfying the condition of the tally each time the particular collision of interest occurs, the tally can instead be satisfied each time any type of collision occurs where the history's score $s_h$ is increased by

$$\frac{\Sigma_R}{\Sigma_\mathrm{t}} w$$

where $\Sigma_R$ is the macroscopic cross section for the reaction type of interest at the collision point (e.g., $\Sigma_\mathrm{a}$ or $\Sigma_\mathrm{s}$) and $\Sigma_\mathrm{t}$ is the macroscopic total cross section at the collision point.

If instead the total scalar flux in the volume is desired, the total collision rate is calculated. The total reaction rate, $R$, is related to the scalar flux, $\phi$, by $R = \Sigma_\mathrm{t}\phi$ so the total scalar flux can be calculated as $\phi = R/\Sigma_\mathrm{t}$. Thus, every time a collision occurs the history's score $s_h$ is increased by $w\Sigma_\mathrm{t}^{-1}$.

All quantities described here are total for the volume, $\widehat{V}$, for the specific region of interest that the collision tallies are defined within. If the volume-average reaction rate or scalar flux are desired, then either each contribution is divided by $\widehat{V}$ or the tally is normalized by $\widehat{V}$ at the conclusion of the calculation.

#### 2.4.2.3 Expected Track-length Tally

A shortcoming of using the collision tally to compute the scalar flux in the region is that in an optically thin medium there are few collisions so many histories are needed to precisely predict the scalar flux. Another method to calculate the total scalar flux in a volume is by calculating the total history track length traveled in the volume. The condition to contribute to such a tally is just that the particle must pass through the

region of interest. Thus, each time a particle travels through a region it increases the history's score by the weight-adjusted track length generated, $w\ell$. As with collision tallies, this calculates the total scalar flux in a region so if the volume-average scalar flux is desired, then each contribution is divided by $\widehat{V}$ or the tally is normalized by $\widehat{V}$ at the conclusion of the calculation.

For this work, rather than a track-length tally an *expected* track-length tally is used as described by Solomon [58, Section 3.6]. This is a deterministic tally that provides a direct comparison to the deterministic results provided by COVRT. Track length is generated in two ways. First, when a particle enters a geometric cell it deposits track length based on the expected distance to collision within the cell until it would exit the cell (e.g., distance $d_c$ away). Second, following a collision in a cell track length is deposited based on the additional expected distance to collision within the cell until it exits the cell (distance $d_c$ away). As such, expected track-length is deposited at the boundary of the track-length tally region as a function of particles entering the region and throughout it as a function of collisions within the region.

The condition to contribute to the tally is satisfied either when entering the region of interest or having a collision within the region of interest. When the condition to contribute is satisfied, the history's score $s_h$ is increased by

$$\frac{1}{\Sigma_{\mathrm{t},c}}\left[1-\exp\left(-d_c\cdot\Sigma_{\mathrm{t},c}\right)\right],$$

where $\Sigma_{\mathrm{t},c}$ is the macroscopic total cross section of the cell.

### 2.4.3 Statistical Properties of Tallies

Assuming that $N_h$ histories are followed and each history contributes total score $s_h$ to the tally, the first moment of the empirical score PDF, which is also the sample mean, is computed via Monte Carlo as

$$\mu_{\mathrm{MC}} = M_{1,\mathrm{MC}} = \frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}} s_h. \tag{2.72}$$

Each history might register a total score made up of multiple contributions to the tally (e.g., by crossing a surface multiple times, colliding multiple times, or traversing a region multiple times thus contributing several track lengths) such that

$$s_h = \sum_{i=1}^{N_{\mathrm{c}}} c_i, \tag{2.73}$$

where $c_i$ are the $N_{\mathrm{c}}$ separate, individual, contributions from history $h$ to the tally. The sample estimate of the second moment of the score PDF is computed as

$$M_{2,\mathrm{MC}} = \frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}} s_h^2. \tag{2.74}$$

Finally, the sample variance, used to measure the spread in the scores, can be calculated as

$$\begin{aligned}
\sigma^2_{\mathrm{MC}} &= \frac{1}{N_{\mathrm{h}}-1}\sum_{h=1}^{N_{\mathrm{h}}}\left(s_h-\mu_{\mathrm{MC}}\right)^2 \\
&\approx \frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}}\left(s_h-\mu_{\mathrm{MC}}\right)^2 \\
&= \frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}}\left(s_h^2-2\mu_{\mathrm{MC}}s_h+\mu^2_{\mathrm{MC}}\right)^2 \\
&= \frac{1}{N_{\mathrm{h}}}\left[\sum_{h=1}^{N_{\mathrm{h}}}s_h^2-2\mu_{\mathrm{MC}}\left(\sum_{h=1}^{N_{\mathrm{h}}}s_h\right)+\mu^2_{\mathrm{MC}}\left(\sum_{h=1}^{N_{\mathrm{h}}}1\right)\right] \\
&= \frac{N_{\mathrm{h}}}{N_{\mathrm{h}}}\left[\frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}}s_h^2-2\mu_{\mathrm{MC}}\left(\frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}}s_h\right)+\mu^2_{\mathrm{MC}}\left(\frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}}1\right)\right] \\
&= \frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}}s_h^2-2\mu_{\mathrm{MC}}\mu_{\mathrm{MC}}+\mu^2_{\mathrm{MC}} \\
&= \frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}}s_h^2-2\mu^2_{\mathrm{MC}}+\mu^2_{\mathrm{MC}} \\
&= \frac{1}{N_{\mathrm{h}}}\sum_{h=1}^{N_{\mathrm{h}}}s_h^2-\mu^2_{\mathrm{MC}} \\
&= M_{2,\mathrm{MC}}-M^2_{1,\mathrm{MC}}. \qquad (2.75)
\end{aligned}$$

The details on why the sample variance includes a factor of $(N_{\mathrm{h}}-1)^{-1}$ are available in [102, Chapter 7], but for the purposes here $N_{\mathrm{h}}$ is usually large so $N_{\mathrm{h}}-1\approx N_{\mathrm{h}}$. The sample standard deviation can be calculated as

$$\sigma_{\mathrm{MC}}=\sqrt{\sigma^2_{\mathrm{MC}}}=\sqrt{M_{2,\mathrm{MC}}-M^2_{1,\mathrm{MC}}}. \qquad (2.76)$$

The Weak Law of Large Numbers states that as the number of statistical trials (i.e., histories) is increased, then the sample mean, sample variance, and sample standard deviation will converge to the population mean, population variance, and population standard deviation, respectively. Furthermore, the Central Limit Theorem states that as independent and identically distributed random trials are performed the properly normalized sum tends toward a normal distribution. Thus, the sample variance of the sample mean, used to measure the spread in the scores about the mean when the number of histories is sufficiently large, is calculated as

$$\sigma^2_{\mu,\mathrm{MC}}=\frac{\sigma^2_{\mathrm{MC}}}{N_{\mathrm{h}}} \qquad (2.77)$$

so the standard deviation of the sample mean is

$$\sigma_{\mu,\mathrm{MC}}=\frac{\sigma_{\mathrm{MC}}}{\sqrt{N_{\mathrm{h}}}}. \qquad (2.78)$$

Accordingly, the relative standard fractional uncertainty in the sample mean is

$$R_{\mathrm{MC}}=\frac{\sigma_{\mu,\mathrm{MC}}}{\mu_{\mathrm{MC}}}. \qquad (2.79)$$

Because the standard deviation of the sample mean (and correspondingly, the relative standard fractional uncertainty in the sample mean) diminishes as $N_{\mathrm{h}}^{-1/2}$, this means that many histories are required to substantially reduce the statistical uncertainty in the calculated result. For example, to achieve a factor of 10 reduction in uncertainty one must use 100 times as many histories as the calculation that gave the original result.

Often, a Monte Carlo calculation provides the sample mean and relative standard fractional uncertainty in the sample mean. The sample variance, to approximate the population variance, can be calculated using the number of histories as

$$\begin{aligned} N_{\mathrm{h}} \left(R_{\mathrm{MC}}\mu_{\mathrm{MC}}\right)^2 &= N_{\mathrm{h}} \left(\frac{\sigma_{\mu,\mathrm{MC}}}{\mu_{\mathrm{MC}}}\mu_{\mathrm{MC}}\right)^2 \\ &= N_{\mathrm{h}} \left(\sigma_{\mu,\mathrm{MC}}\right)^2 \\ &= N_{\mathrm{h}} \left(\frac{\sigma_{\mathrm{MC}}}{\sqrt{N_{\mathrm{h}}}}\right)^2 \\ &= N_{\mathrm{h}} \frac{\sigma_{\mathrm{MC}}^2}{N_{\mathrm{h}}} \\ &= \sigma_{\mathrm{MC}}^2. \end{aligned} \tag{2.80}$$

### 2.4.4 Variance Reduction & Computational Particle Statistical Weight

If all random sampling is performed according to physical constants, then the calculation is usually referred to as an "analog calculation." However, this approach can often require an intractably large number of histories to be required to obtain an adequately low uncertainty [Eq. (2.78)]. For example, if an analog calculation is performed for a radiation shielding analysis, and the shield is effective, few histories will penetrate the shield to score in a tally beyond it. Alternatively, the physical processes may be modified in such a way that biases the random walk in such a way that preserves the overall mean behavior and increases the efficiency of estimating the mean. The techniques for modifying the random walk in this way are referred to as variance reduction.

This modification attempts to either (1) reduce the uncertainty (or variance, hence the name) in the sample mean or (2) reduce the computer time taken to calculate the sample mean with a particular level of uncertainty (or correspondingly, permit following a greater number of histories in a fixed amount of time). Generally, a variance-reduction technique that reduces the uncertainty increases the computation's time and vice versa. Because of the need to balance time and uncertainty, Monte Carlo calculations often have a figure of merit (FOM) calculated to assess the relative efficiency of the calculation versus other, similar, calculations on the same computer that use different variance-reduction parameters. A common FOM is

$$FOM_{\mathrm{MC}} = \frac{1}{R_{\mathrm{MC}}^2 T_{\mathrm{MC}}}, \tag{2.81}$$

where $T_{\mathrm{MC}}$ is the total time necessary to perform the calculation. Because $R_{\mathrm{MC}}^2$ converges as $N_{\mathrm{h}}^{-1}$ [Eq. (2.78) and (2.79)] and $T_{\mathrm{MC}}$ scales on average as $N_{\mathrm{h}}$, the FOM converges to a constant value. Based on Eq. (2.81), lower variance and lower time increase the FOM, so seeking parameters that maximize the FOM is often the goal in Monte Carlo radiation shielding calculations. Conversely, one can define a Monte Carlo calculation's

computational cost by inverting the FOM to obtain

$$\$_{\mathrm{MC}} = R_{\mathrm{MC}}^2 T_{\mathrm{MC}} = \frac{\left(M_{2,\mathrm{MC}} - M_{1,\mathrm{MC}}^2\right) T_{\mathrm{MC}}}{M_{1,\mathrm{MC}}^2} \tag{2.82}$$

where the objective is then to minimize the computational cost by balancing the minimization of both variance and time.

To apply variance reduction in a way that does not does not modify the sample mean, the concept of computational particle statistical weight is used. The statistical weight of a computational particle, $w$, is another phase-space parameter of the particle like its position, direction, and energy. Typically, a particle is created with a statistical weight of one.

For a strictly analog calculation, the weight will never change. However, for non-analog calculations, the statistical weight is adjusted when the particle is affected by variance reduction. When the particle contributes to a tally, its contribution is scaled by $w$. For example with a leakage tally, rather than contributing one upon leakage, the particle will contribute $w$. As another example for track-length tallies, rather than contributing $\ell$ the particle will contribute $w\ell$.

There are numerous variance-reduction techniques available in Monte Carlo radiation transport codes. Such techniques are particularly beneficial in shielding analyses because (1) the region of interest is often removed from the radiation source and (2) in many shielding scenarios the shield is designed specifically to prevent radiation from reaching a region of interest.

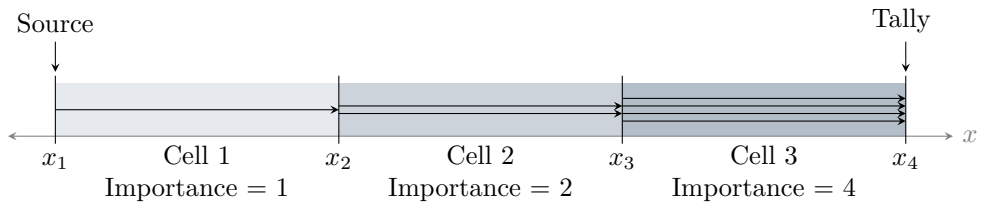
A common variance-reduction technique based on importance splitting and rouletting is described next to demonstrate how statistical weights are modified by variance reduction. Then, a detailed description of the DXTRAN variance-reduction technique follows.

### 2.4.5 Importance Splitting & Rouletting

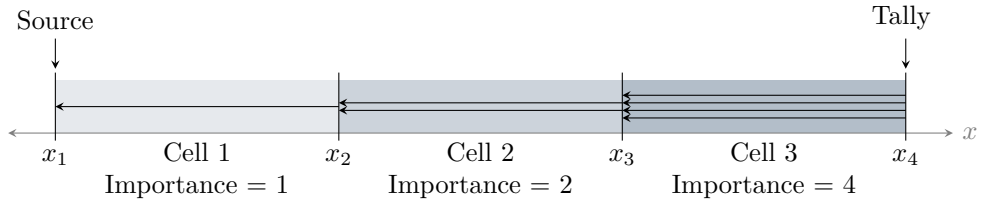
A common variance-reduction technique is geometry-based importance splitting and rouletting. It is primarily used to induce additional sampling of phase-space considered "important" by the analyst. Thus, importance splitting attempts to reduce variance by increasing sampling at the cost of greater computational time. Conversely, rouletting attempts to reduce time by killing particles that appear to be traveling in relatively unimportant directions at the cost of increased variance as a result of greater weight fluctuation.

A common way to implement geometry splitting and rouletting is by assigning varying levels of "importance" to different pieces of phase space within the Monte Carlo analysis. Generally, a user will assign an importance of one at/around the particle source position and gradually increase the phase-space importances as the particle moves toward the region of interest. In this way, the geometry-based importances will often resemble the adjoint flux distribution calculated by a deterministic method.

In Fig. 2.10a, as a particle travels from left to right, from a region of lower importance to higher importance, it is split into $k_{\mathrm{IS}}$ particles according to the ratio of the importances of the geometry cells separated by the




**(a)** Geometry Splitting




**(b)** Rouletting

**Figure 2.10:** Geometry-based Importance Splitting and Rouletting Effect on Particle Population

Monte Carlo surfaces at $x = x_2$ or $x = x_3$, respectively, as

$$k_{\text{IS}} = \frac{I_2}{I_1} = \frac{2}{1} = 2 \text{ or } k_{\text{IS}} = \frac{I_3}{I_2} = \frac{4}{2} = 2.$$

When split, the weight of each particle is multiplied by a factor of

$$k_{\text{IS}}^{-1} = \frac{I_1}{I_2} = \frac{1}{2} \text{ or } k_{\text{IS}}^{-1} = \frac{I_2}{I_3} = \frac{2}{4} = \frac{1}{2},$$

according to the inverse ratio of importance of the geometry cells separated by the Monte Carlo surface.

Thus far only integer-based importance splitting has been described. There are a number of approaches to performing non-integer splitting (e.g., if adjacent cells have importances 2 and 3). One such approach is to perform a random sampling if $k_{\text{IS}}$ is not an integer where the particle will be split into $\lfloor k_{\text{IS}} \rfloor + 1$ particles if $0 \leq \zeta < k_{\text{IS}} - \lfloor k_{\text{IS}} \rfloor$ and it will be split into $\lfloor k_{\text{IS}} \rfloor$ particles otherwise. The weight of each split particle is multiplied by $\lfloor k_{\text{IS}} \rfloor$ in either case to minimize the dispersion in weights.

Conversely, in Fig. 2.10b, as a particle travels from right to left, from a region of higher importance to lower importance, it is rouletted according to the ratio of the importances of the geometry cells separated by the Monte Carlo surfaces at $x = x_3$ or $x = x_2$, respectively, as

$$\alpha_{\text{IR}} = \frac{I_1}{I_2} = \frac{1}{2} \text{ or } \alpha_{\text{IR}} = \frac{I_2}{I_3} = \frac{2}{4} = \frac{1}{2}.$$

If a particle survives rouletting its weight is promoted by the inverse ratio of the importances,

$$\alpha_{\text{IR}}^{-1} = \frac{I_2}{I_1} = \frac{2}{1} = 2 \text{ or } \alpha_{\text{IR}}^{-1} = \frac{I_3}{I_2} = \frac{4}{2} = 2.$$

Thus, importance rouletting attempts to reduce the time spent following particles as they move into relatively less important regions of phase space at the cost of increased weight dispersion, and thus larger variance, by

virtue of promoting the weight for any particles that survive rouletting.

Note that by splitting a computational particle into multiple whole computational particles with reduced weight using the aforementioned integer splitting procedure, the number of contributions to a tally can be increased. However, the total score contributed to the tally will not exceed the total score contributed by an analog history because weight is preserved during integer splitting. This will be contrasted with DXTRAN, described next. Note that this is not true for non-integer splitting and rouletting, where total weight is only preserved on average.

### 2.4.6 DXTRAN

A brief overview of a simplified DXTRAN process in MCNP6 is given in Section 1.5.1. This section gives substantially more detail on the MCNP DXTRAN process. It begins with a detailed description of the spherical MCNP DXTRAN process and the associated weight corrections. As the best available reference, much of this information is taken from the MCNP5 theory manual [1]. This section then describes the simplifying assumptions made for this work. It continues with a description of 1-D DXTRAN processing and a demonstration of how histories affected by DXTRAN can contribute multiple times to a tally.

#### 2.4.6.1 Overview of the DXTRAN Process

DXTRAN can be applied in either of two complementary ways by (a) drawing weight toward a region of interest and then permitting continued sampling (i.e., as a magnet), or (b) preventing high-weight particles from directly streaming into a region of interest (i.e., as a shield). The former approach is more familiar to most users because a DXTRAN region is often thought to encompass a spatial region of interest that is not being adequately sampled. The DXTRAN region is defined about the region of interest to create and direct the created (i.e., DXTRAN) particles toward and into the region to increase the sampling within. However, it is possible to use a DXTRAN region to prevent particles from directly entering the region of interest. This behavior can be beneficial when high-weight particles attempt to enter the region and would otherwise induce large variance fluctuation to tallies within.

The DXTRAN particle is treated as a normal computational particle once it is created. However, if the DXTRAN particle exits the DXTRAN region, it will initiate the DXTRAN process and create daughter DXTRAN particles. In this way, DXTRAN acts as a splitting process where the DXTRAN particle is split and teleported (with weight attenuation) to the surface of the DXTRAN region but then behaves normally afterward. The DXTRAN particle represents the component of the initiating particle that reaches the DXTRAN region on its next free flight.

A subtlety with DXTRAN is how the weight is balanced in such a way to preserve the mean physical behavior. If the particle that initiated DXTRAN attempts to enter the DXTRAN region on its next free-flight following initiation of the DXTRAN process, it is terminated upon reaching the boundary of the DXTRAN region. This preserves mean physical behavior in that the amount of weight created on the DXTRAN surface by the DXTRAN particle is, on average, balanced by the destruction of weight of their initiating particles attempting to enter the DXTRAN region on their next free flight (and subsequently killed).

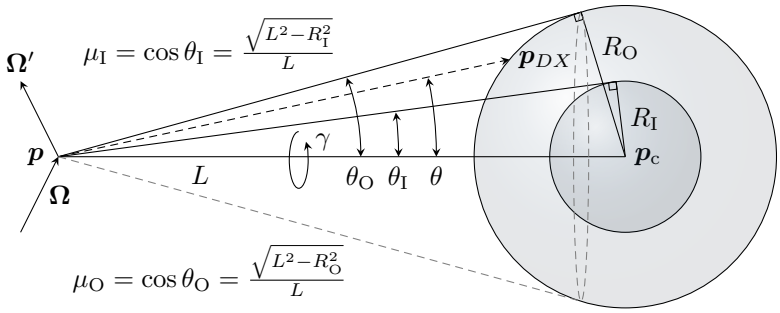



**Figure 2.11:** MCNP Spherical DXTRAN Process Schematic

A schematic of a spherical MCNP DXTRAN region relative to a DXTRAN initiation event at spatial point $\boldsymbol{p}$ is given in Fig. 2.11 for reference in the upcoming discussion.

Additionally, it is possible to have multiple interacting DXTRAN spheres and for DXTRAN spheres to be nested [82], but this work does not consider those cases (despite currently being supported since in the MCNP code since version 6.1-beta2 [103]).

#### 2.4.6.2 Detailed MCNP6 Spherical DXTRAN Process

A single MCNP DXTRAN "sphere" can be defined by concentric inner and outer spheres about the center point of the spheres at $\boldsymbol{p}_{\mathrm{c}} = (x_{\mathrm{c}}, y_{\mathrm{c}}, z_{\mathrm{c}})$. The inner sphere has radius $R_{\mathrm{I}}$ and the outer sphere has radius $R_{\mathrm{O}}$. The inner sphere is used *only* to provide supplemental angle biasing, so when a DXTRAN sphere is being referred to without clarification, it refers to the outer sphere (and often implies that $R_{\mathrm{I}} = R_{\mathrm{O}}$ in practice).

The DXTRAN process in MCNP6 for a particular DXTRAN sphere initiates upon valid source emission or non-absorptive collision at spatial point $\boldsymbol{p}$. A source emission or collision event is valid if the event occurs outside the DXTRAN sphere and has a non-zero analog probability of having the particle at $\boldsymbol{p}$ emerge in the direction of the DXTRAN sphere. For nested spheres, DXTRAN processing occurs for each sphere that $\boldsymbol{p}$ is external to. An example of an invalid event is mono-directional source emission where the orientation and direction of the source guarantees that source particles cannot reach the DXTRAN sphere upon their first free flight.

If DXTRAN is valid the DXTRAN process is started. The first step is to determine whether the DXTRAN process will continue based on the user-specified rouletting parameter $\beta_c$ (`DXC` in MCNP6). The user can assign $0 < \beta_c \leq 1$ on a cell-wise basis with $c = 1, 2, 3, \ldots, N_{\mathrm{cells}}$. The default value for $\beta_c$ is one such that DXTRAN is always played. However, a user might choose to play DXTRAN less often if a particular cell is expected to have a minimal contribution to the DXTRAN sphere. Computational time can be saved by rouletting DXTRAN particles from that cell. Because particles attempting to enter the DXTRAN sphere during free flight are assumed to have initiated the DXTRAN process by the MCNP code, the user must not set $\beta_c = 0$ because the calculation will become biased (no DXTRAN particles can be created but particles colliding in cell $c$ that attempt to enter the DXTRAN sphere in their next free flight will still be killed). With probability $\beta_c$ DXTRAN processing will continue and with probability $(1 - \beta_c)$ the DXTRAN particle will be killed outright and the process will not continue. However, because DXTRAN processing is initiated the

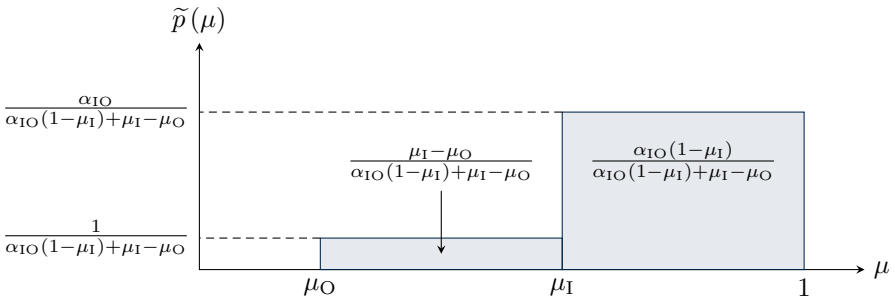



**Figure 2.12:** Biased Inner/outer DXTRAN Sphere PDF.

initiating particle will be killed if it attempts to enter the DXTRAN sphere on its next free flight. For nested spheres, the initiating particle will be killed if it attempts to enter any DXTRAN sphere because the process was initiated for all spheres.

Assuming the DXTRAN particle survived rouletting according to $\beta_c$, a polar axis is defined between $\boldsymbol{p}$ and $\boldsymbol{p}_c$ that has length $L = \|\boldsymbol{p}_c - \boldsymbol{p}\|$. Throughout this work $\|\cdot\|$ represents the Euclidian norm. Two polar cosines can now be determined:

$$\mu_I = \cos(\theta_I) = \frac{\sqrt{L^2 - R_I^2}}{L}, \tag{2.83a}$$

$$\mu_O = \cos(\theta_O) = \frac{\sqrt{L^2 - R_O^2}}{L}. \tag{2.83b}$$

The polar cosines in Eq. (2.83) define a non-analog polar cosine sampling PDF,

$$\widetilde{p}(\mu_{DX}) = \begin{cases} \frac{1}{\alpha_{IO}(1-\mu_I)+\mu_I-\mu_O}, & \mu_O \le \mu_{DX} \le \mu_I \\ \frac{\alpha_{IO}}{\alpha_{IO}(1-\mu_I)+\mu_I-\mu_O}, & \mu_I < \mu_{DX} \le 1 \\ 0 & \text{otherwise} \end{cases}, \tag{2.84}$$

as shown in Fig. 2.12. In Eq. (2.84) $\alpha_{IO}$ is an importance factor given to the inner sphere relative to the outer sphere (for cross-reference, this quantity is represented by $Q$ in the MCNP manual [1]).

As such, the polar direction of the DXTRAN particle is sampled from $\widetilde{p}(\mu)$ such that $\mu_{DX}$ is sampled uniformly in $(\mu_I, 1]$ using $\mu_{DX} = \mu_I + \zeta(1 - \mu_I)$ with probability

$$\alpha_{IO}(1 - \mu_I) / [\alpha_{IO}(1 - \mu_I) + \mu_I - \mu_O]$$

or else $\mu_{DX}$ is sampled uniformly in $[\mu_O, \mu_I]$ using $\mu_{DX} = \mu_O + \zeta(\mu_I - \mu_O)$ with probability

$$(\mu_I - \mu_O) / [\alpha_{IO}(1 - \mu_I) + \mu_I - \mu_O].$$

With the polar cosine selected, the azimuthal angle is sampled uniformly (because of rotational invariance about the polar axis) as $\gamma_{DX} = 2\pi\zeta$ with probability

$$p(\gamma_{DX}) = \frac{1}{2\pi}. \tag{2.85}$$

With $\mu_{\mathrm{DX}}$ and $\gamma_{\mathrm{DX}}$, the direction vector of the DXTRAN particle, $\boldsymbol{\Omega}_{\mathrm{DX}}$, can be calculated[1] as

$$\boldsymbol{\Omega}_{\mathrm{DX}} = \begin{pmatrix} \boldsymbol{\Omega}_{\mathrm{DX,x}} \\ \boldsymbol{\Omega}_{\mathrm{DX,y}} \\ \boldsymbol{\Omega}_{\mathrm{DX,z}} \end{pmatrix} = \begin{pmatrix} \mu_{\mathrm{DX}}\boldsymbol{a}_{\mathrm{x}} + \frac{\sqrt{1-\mu_{\mathrm{DX}}^2}[\boldsymbol{a}_{\mathrm{x}}\boldsymbol{a}_{\mathrm{z}}\cos(\gamma_{\mathrm{DX}})-\boldsymbol{a}_{\mathrm{y}}\sin(\gamma_{\mathrm{DX}})]}{\sqrt{1-\boldsymbol{a}_{\mathrm{z}}^2}} \\ \mu_{\mathrm{DX}}\boldsymbol{a}_{\mathrm{y}} + \frac{\sqrt{1-\mu_{\mathrm{DX}}^2}[\boldsymbol{a}_{\mathrm{y}}\boldsymbol{a}_{\mathrm{z}}\cos(\gamma_{\mathrm{DX}})+\boldsymbol{a}_{\mathrm{x}}\sin(\gamma_{\mathrm{DX}})]}{\sqrt{1-\boldsymbol{a}_{\mathrm{z}}^2}} \\ \mu_{\mathrm{DX}}\boldsymbol{a}_{\mathrm{z}} + \sqrt{1-\mu_{\mathrm{DX}}^2}\sqrt{1-\boldsymbol{a}_{\mathrm{z}}^2}\cos(\gamma_{\mathrm{DX}}) \end{pmatrix}, \tag{2.86}$$

where

$$\boldsymbol{a} = \begin{pmatrix} \boldsymbol{a}_{\mathrm{x}} \\ \boldsymbol{a}_{\mathrm{y}} \\ \boldsymbol{a}_{\mathrm{z}} \end{pmatrix} = \frac{\boldsymbol{p}_{\mathrm{c}} - \boldsymbol{p}}{L}. \tag{2.87}$$

A projection is then made along $\boldsymbol{\Omega}_{\mathrm{DX}}$ from $\boldsymbol{p}$ and the first intersection point encountered on the DXTRAN outer sphere ($\partial\Gamma_{\mathrm{DX}}$) is used as the DXTRAN particle's creation point, $\boldsymbol{p}_{\mathrm{DX}}$.

As the projection between $\boldsymbol{p}$ and $\boldsymbol{p}_{\mathrm{DX}}$ is made, the optical distance between the two points is calculated as

$$\lambda(\boldsymbol{p}, \boldsymbol{p}_{\mathrm{DX}}, E) = \int_0^{\|\boldsymbol{p}_{\mathrm{DX}}-\boldsymbol{p}\|} \Sigma_{\mathrm{t}}(\boldsymbol{p} + \ell\boldsymbol{\Omega}_{\mathrm{DX}}, E)\,\mathrm{d}\ell. \tag{2.88}$$

This calculation is made by ray tracing from $\boldsymbol{p}$ along $\boldsymbol{\Omega}_{\mathrm{DX}}$ for a total distance $\|\boldsymbol{p}_{\mathrm{DX}} - \boldsymbol{p}\|$. As the ray trace proceeds, the Monte Carlo surface crossings on the particle's trajectory are used to determine the total track length traveled through for each Monte Carlo cell's material. Because of this, the ray trace time requirement scales according to the number of Monte Carlo surfaces encountered. This behavior is important later in Chapter 4 when computing the time required to perform DXTRAN processing.

When the optical distance is calculated according to Eq. (2.88) the probability of free flight between $\boldsymbol{p}$ and $\boldsymbol{p}_{\mathrm{DX}}$ is calculated as

$$p_{\mathrm{ff}}(\boldsymbol{p}, \boldsymbol{p}_{\mathrm{DX}}) = \exp\left(-\lambda(\boldsymbol{p}, \boldsymbol{p}_{\mathrm{DX}})\right). \tag{2.89}$$

A temporary statistical weight of the DXTRAN particle is calculated as

$$w_{\mathrm{DX}} = w \underbrace{\frac{1}{\beta_{\mathrm{c}}}}_{(1)} \underbrace{\frac{p(\mu)}{\widetilde{p}(\mu)}}_{(2)} \underbrace{\frac{p_{\mathrm{ff}}(\boldsymbol{p}, \boldsymbol{p}_{\mathrm{DX}})}{1}}_{(3)}, \tag{2.90}$$

where the weight correction terms are included to account for:

(1) surviving the user-assigned rouletting,

(2) emerging in the direction of the DXTRAN sphere, perhaps being sampled non-uniformly with preference toward the inner sphere, and

(3) accounting for the particle streaming to the DXTRAN surface without collision.

[1] Reproduced from the OpenMC [104] Theory and Methodology online documentation (revision `455efffd`) located at `https://openmc.readthedocs.io/en/stable/methods/index.html`.

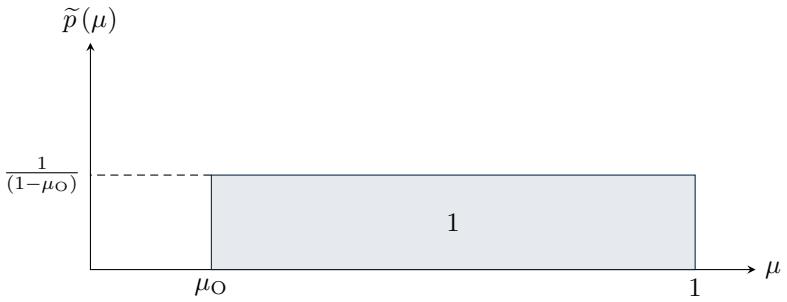



**Figure 2.13:** Only Outer DXTRAN Sphere PDF.

After $w_{DX}$ is calculated, it is compared with a DXTRAN weight cutoff limit $w_{DXcut}$. If $w_{DX} > w_{DXcut}$, the particle is created at $\boldsymbol{p}_{DX}$ in direction $\boldsymbol{\Omega}$ with final weight given by Eq. (2.90). If $w_{DX} < w_{DXcut}$, the DXTRAN particle is rouletted. The DXTRAN particle is killed with probability $1 - w_{DX}/w_{DXcut}$ and it survives with probability $w_{DX}/w_{DXcut}$ to be created at $\boldsymbol{p}_{DX}$ in direction $\boldsymbol{\Omega}$ with final weight

$$w_{DX} = w \frac{1}{\beta_c} \frac{p(\mu)}{\widetilde{p}(\mu)} \frac{p_{ff}(\boldsymbol{p}, \boldsymbol{p}_{DX})}{1} \frac{1}{\alpha_{\mathcal{T}}}, \tag{2.91}$$

where the final factor of $\alpha_{\mathcal{T}}^{-1}$ accounts for surviving rouletting during transmission.

Some simplifications are made to the aforementioned DXTRAN process to enable the work herein. These changes are described next.

#### 2.4.6.3 Simplifying Assumptions for this Work

Several simplifications are made to the previously described MCNP6 DXTRAN process for this work. These are:

1. The inner sphere has the same radius as the outer sphere. This eliminates complexity by making the non-analog emergence PDF

$$\widetilde{p}(\mu_{DX}) = \begin{cases} \frac{1}{1-\mu_O}, & \mu_O < \mu_{DX} \leq 1 \\ 0 & \text{otherwise} \end{cases}, \tag{2.92}$$

   as shown in Fig. 2.13. This is reasonable because, in the author's experience, analysts seldom define inner spheres with $R_I < R_O$. Extending the work herein to study the effect of using $R_I < R_O$ and determining how best to set $R_I$ to maximize Monte Carlo FOM represents an area for future work.

2. There is no weight-based rouletting during transmission. This makes DXTRAN a weight-independent game. In turn, this dramatically simplifies the deterministic solution of the history-score moment equations in Chapter 3 and eliminates the weight correction of $1/\alpha_{\mathcal{T}}$.

3. DXTRAN processing is only performed for non-absorptive collisions. That is, DXTRAN is disabled entirely for source emission. This simplification is made to make computing the inner product of the

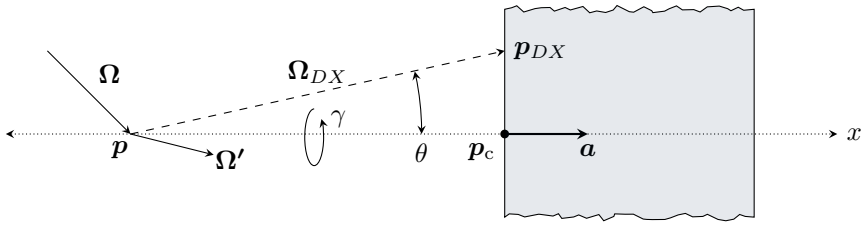



**Figure 2.14:** 1-D Cartesian DXTRAN Region

HSMEs with the forward source easier as described at the end of Chapter 3. This is a reasonable simplification for several reasons. In cases with large spatial separation between the source and DXTRAN region there will be minimal dispersion in DXTRAN particle weight. Similarly, for scattering-dominated cases the weight dispersion for the DXTRAN particles is dominated by collisions close to the DXTRAN region versus further away. Note that this means that particles emitted from the source (i.e., uncollided particles) *can* enter the DXTRAN region.

4. This work primarily focuses on the DXTRAN variance-reduction technique. However, several test cases incorporate importance splitting and rouletting. In all cases, implicit capture [1, page 2-153] and weight cutoff (weight rouletting) [1, page 2-146] are explicitly disabled and no other variance-reduction techniques are considered.

5. As previously mentioned, only a single DXTRAN region is considered [Section 2.4.6.1].

As a result of these simplifications, the DXTRAN process is performed on a weight-independent basis and the DXTRAN particle weight correction becomes

$$w_{\mathrm{DX}} = \frac{w}{\beta_c} \frac{p\left(\mu\right)}{\widetilde{p}\left(\mu\right)} p_{\mathrm{ff}}\left(\boldsymbol{p}, \boldsymbol{p}_{\mathrm{DX}}\right). \tag{2.93}$$

This process description is consistent with Fig. 1.1 in Chapter 1. These simplifications still leave sufficient freedom for an analyst to choose the DXTRAN region size and position and the cell-wise rouletting parameters. As such, these DXTRAN parameters are what this work seeks to analyze and optimize.

#### 2.4.6.4 3-D Cartesian DXTRAN

As noted in Chapter 1, this work builds upon the COVRT software, which uses the Cartesian spatial discretization scheme described in Section 2.3.2. However, MCNP regions could only be described using spheres at the outset of this work, as described previously. As such, one element of the novel work herein is the ability to define DXTRAN regions using arbitrary convex polyhedra (see Appendix B). This permits a direct comparison between MCNP results and COVRT results. For this work, only right-rectangular regions are used to define DXTRAN regions.

#### 2.4.6.5 1-D Cartesian DXTRAN

An example using a 1-D DXTRAN region is shown in Fig. 2.14, where the DXTRAN region is effectively infinite in the $y$-$z$ plane. The DXTRAN process using a 1-D Cartesian region is similar to the 3-D spherical regions. A polar axis is defined along $\boldsymbol{a}$ and $\mu_{\mathrm{DX}}$ and $\gamma_{DX}$ are sampled as in Eqs. (2.92) and (2.85). From

there, $\mathbf{\Omega}_{\mathrm{DX}}$ is calculated with Eq. (2.86), the projection to the DXTRAN surface is made to find $\boldsymbol{p}_{DX}$, the free-flight probability is calculated with Eq. (2.89), and the DXTRAN particle is created with weight according to Eq. (2.93). For the 2-D calculations herein, the DXTRAN region is effectively infinite in the $z$ direction.

#### 2.4.6.6 Joint Scoring with DXTRAN

One side effect of the DXTRAN process is that a history has an unbounded ability to score if allowed to collide two or more times, which can be compared with importance splitting as described in Section 2.4.5. This can lead to large dispersion in scores as a result of DXTRAN. This also creates subtlety that must be accurately characterized by the second-moment HSME described in Chapter 3, where multiple tally contributions from a history, and the resulting effect on tally second moment, are accounted for.

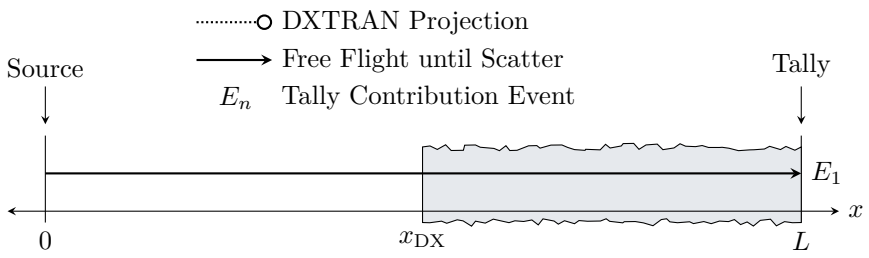
The ability for a particle to score multiple times is demonstrated in Fig. 2.15, which shows the tally contribution pathways for a particle that undergoes collision-only DXTRAN processing in a 1-D calculation. The DXTRAN region is the shaded region in the right-half of the figure; the unshaded region to the left represents space outside the DXTRAN region.

Figure 2.15a shows an uncollided particle's scoring pathway. The only way an uncollided particle can contribute to the tally is to undergo free flight from the forward source to the tally region. To reiterate: the source particle is able to penetrate into the DXTRAN region and contribute to the tally because DXTRAN is played only at collisions.

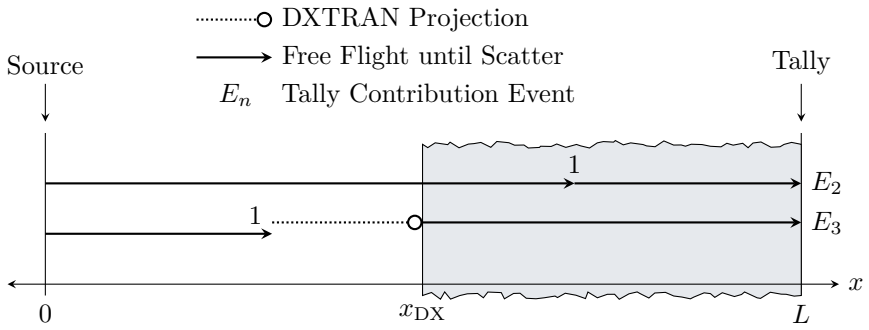
Figure 2.15b shows how particles that undergo exactly one collision can contribute to the tally. The particle can collide either inside or outside the DXTRAN region. If the collision occurs inside the DXTRAN region, no DXTRAN particle is generated, but if the initiating particle emerges in the direction of the tally and undergoes free flight until the tally, it will contribute to the tally. Conversely, if the initiating particle collides outside the DXTRAN region, it will produce a DXTRAN particle on the DXTRAN surface, directed into the DXTRAN region, that must then undergo free-flight until scoring. Thus, a history that undergoes only one collision can contribute to the tally either via event $E_2 \oplus E_3$ (i.e., $E_2$ exclusive-or event $E_3$)—it cannot contribute to the tally by both random walks.

However, Figure 2.15c shows how particles that undergo exactly two collisions can contribute to the tally. First, free flight into the DXTRAN region followed by two collisions in the DXTRAN region followed by scoring are illustrated by events $E_4$ and $E_5$. Event $E_6$ has the particle entering the DXTRAN system, colliding and emerging such that it exits the DXTRAN region on its next free flight, and colliding again. In this case, a DXTRAN particle is generated on the DXTRAN surface that must then undergo free flight to contribute to the tally.

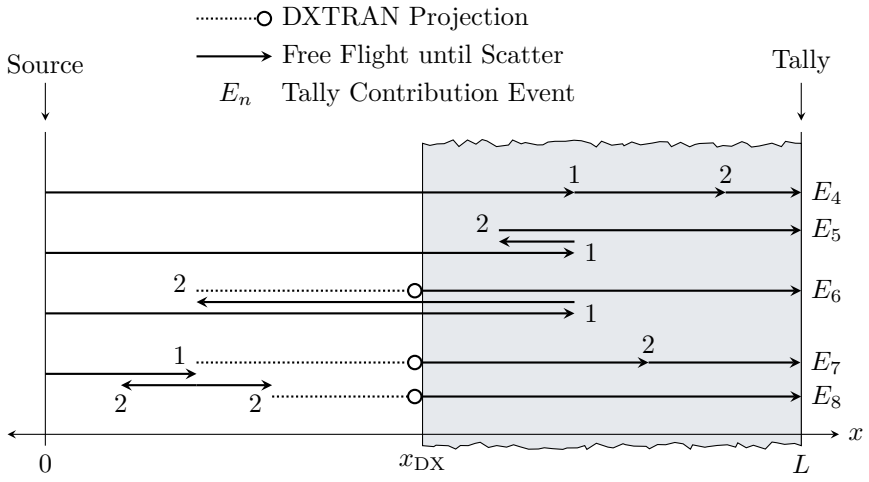
The last set of contributing events in Fig. 2.15c demonstrates how DXTRAN can contribute to the tally multiple times. For example, a particle can undergo free flight from the source, collide outside the DXTRAN region, generate a DXTRAN particle on the DXTRAN surface that can then undergo free flight until it has a collision in the DXTRAN region, undergo another free flight and then contribute to the tally (event $E_7$). In addition, the initiating particle can go on to have a second collision outside the DXTRAN region, producing




**(a)** 0 Collisions




**(b)** 1 Collision




**(c)** 2 Collisions

**Figure 2.15:** Example DXTRAN Tally Contribution Events for 0, 1, and 2 Collisions

a DXTRAN particle on the DXTRAN surface that then goes on to free fly until contributing to the tally (event $E_8$). With this last evolution, one can observe contributions to the tally caused by $E_7$, $E_8$, or $E_7 \wedge E_8$ (i.e., $E_7$ and $E_8$). In addition, one can observe scores from tally events $E_3 \wedge E_8$. With these joint events ($E_7 \wedge E_8$ and $E_3 \wedge E_8$), a history has the ability to score more than its initial weight. The effect on multiple collisions and joint scores is shown explicitly in Fig. D.3. This effect will become important in the calculation of $Q_2(\boldsymbol{P}_0)$ in Chapter 3.

## 2.5 Summary

This chapter provides a heuristic derivation of the steady-state fixed-source particle transport equation without fission. This equation is given in its integro-differential forward and adjoint forms. It is also given in a strictly integral formulation. This derivation and the various forms of the equation are provided to establish nomenclature and to familiarize the reader with how the equations are used in the remainder of this work. In particular, Chapters 3 and 4 use path integral formulations of statistical moment equations, which are similar in form to the strictly integral formulation of the particle transport equation. Then, the statistical moment equations are re-cast into an integro-differential form resembling the adjoint particle transport equation and solved using the procedure described in Chapter 5.

To enable the numeric solution methods described in Chapter 5, this chapter also provides the discretization procedures necessary (in space and angle) to solve the particle transport equation deterministically. It also provides the energy discretization process for completeness, though multi-group cross sections are specified directly for the test cases in this work. Further, this chapter provides an overview of the Monte Carlo statistical sampling, transport, and tallying processes. Finally, this chapter gives a review of the MCNP DXTRAN variance-reduction technique as well as the simplifications made to it for this work.

# Chapter 3

# History-score Moment Equations

The history-score probability density function (HSPDF) analytically describes the random walks that a Monte Carlo particle can follow and the associated score that it may contribute to a tally. As such, the HSPDF gives the probability of a history and any progeny contributing differential score $\mathrm{d}s$ about $s$ to a tally from a given position of phase space. Accordingly, the scoring moments of the HSPDF, the history-score moment equations (HSMEs), can be calculated to determine the associated statistical moments for the tally as a function of phase-space position. The first moment is the tally mean [Eq. (2.72)], which is the expected tally score from a given position in phase space. The first two moments can be used to calculate the tally population variance [Eq. (2.75)].

Similar to the HSPDF, the future-time probability density function (FTPDF) analytically describes the time necessary to following each component of the Monte Carlo random walk, so it describes the probability of a history and any progeny contributing time $\mathrm{d}\tau$ about $\tau$ to the calculation from a given position of phase space. The first moment of the FTPDF is the future-time equation (FTE), which gives the expected time for a history as a function of phase-space position.

The foundation of the HSMEs and the FTE are the transport kernels contained within them. As such, this chapter begins with a review of the transport kernels used in this work. These kernels are then used to construct the HSPDFs that describe particle random walks. The first and second statistical moments of the HSPDFs are then calculated to obtain the HSMEs. The analog HSMEs are examined first to familiarize the reader with the overall approach to calculating the HSMEs and as a basis for comparison once variance reduction is applied. The HSMEs incorporating DXTRAN transport are then derived. Discussion of the FTPDF and FTE is reserved for Chapter 4.

## 3.1 Summary of Transport Kernels

Transport kernels are PDFs used to describe the probability of a particle at some point in weight-augmented phase space element $\boldsymbol{P} = (\boldsymbol{x}, \boldsymbol{\Omega}, E, t, w)$ undergoing transition to some subsequent point in phase space $\mathrm{d}\boldsymbol{P}' = \mathrm{d}V'\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}t'\mathrm{d}w'$ about $\boldsymbol{P}' = (\boldsymbol{x}', \boldsymbol{\Omega}', E', t', w')$. Because transport kernels are PDFs, transport kernels must obey the rules of PDFs: they cannot be negative and when integrated over the total resulting phase space (i.e., $\int \mathrm{d}\boldsymbol{P}' = \int \mathrm{d}\boldsymbol{x}' \int \mathrm{d}\Omega' \int \mathrm{d}E' \int \mathrm{d}t' \int \mathrm{d}w'$) the result is one.

A series of transport kernels are used to describe the transport progression for a particle from one region of

phase space until termination (e.g., absorption or leakage). This series of transport progressions represents the random walk that a particle can undergo. Thus, there are a variety of transport progressions but when all possible transport progression are considered, the HSPDF giving the probability of contributing score d$s$ about $s$ is the result.

Each transport kernel used in this work is defined and discussed in the upcoming subsections. Note that, because this work is focused on steady-state calculations, the time variable has been eliminated so $\boldsymbol{P} = (\boldsymbol{x}, \boldsymbol{\Omega}, E, w)$.

### 3.1.1 Free-flight Transmission Kernel

The analog particle free-flight transmission kernel used to specify the probability of a particle moving in free flight from $\boldsymbol{x}$ to $\boldsymbol{x}'$ (with the same direction, energy, and weight) is

$$\mathcal{T}\left(\boldsymbol{P}, \boldsymbol{P}'\right) \mathrm{d}\Omega' \mathrm{d}E' \mathrm{d}w' = \exp\left( - \int_0^{\|\boldsymbol{x}'-\boldsymbol{x}\|} \Sigma_{\mathrm{t}}\left(\boldsymbol{x} + \ell\boldsymbol{\Omega}, E\right) \mathrm{d}\ell \right) \delta\left(\Omega' - \Omega\right) \delta\left(E' - E\right) \delta\left(w' - w\right) \mathrm{d}\Omega' \mathrm{d}E' \mathrm{d}w'. \tag{3.1}$$

This kernel is characterized by exponential decay in probability of traveling, unchanging, with direction $\boldsymbol{\Omega}$, energy $E$, and weight $w$ along a characteristic ray from $\boldsymbol{x}$ to $\boldsymbol{x}'$ according to the optical thickness between $\boldsymbol{x}$ and $\boldsymbol{x}'$.

This kernel is also characterized by an optical reciprocity relationship where the probability of free flight from $\boldsymbol{x}$ to $\boldsymbol{x}'$ along direction $\boldsymbol{\Omega}$ is equal to the probability of free-flight from $\boldsymbol{x}'$ to $\boldsymbol{x}$ along direction $-\boldsymbol{\Omega}$. This relationship is important to the relationship between forward and adjoint transport. A particle has the same probability of transmitting from $\boldsymbol{x}$ to $\boldsymbol{x}'$ along direction $\boldsymbol{\Omega}$ in a forward sense as it does from transmitting from $\boldsymbol{x}'$ to $\boldsymbol{x}$ along direction $-\boldsymbol{\Omega}$ in an adjoint sense.

### 3.1.2 Collision Kernel

The collision kernel gives the probability that a particle that has undergone collision in incremental distance d$\ell'$ along a characteristic ray between $\boldsymbol{x}$ and $\boldsymbol{x}'$ along $\boldsymbol{\Omega}$ in phase-space $\boldsymbol{P}$ and result in the particle in phase-space d$\boldsymbol{P}'$ about $\boldsymbol{P}'$. In an analog calculation there is no difference between $\boldsymbol{P}$ and $\boldsymbol{P}'$, and (usually) in non-analog games the only change that may occur is in the weight domain. The kernel is given by

$$\mathcal{K}\left(\boldsymbol{P}, \boldsymbol{P}'\right) \mathrm{d}\boldsymbol{P}' = \Sigma_{\mathrm{t}}\left(\boldsymbol{x}, E\right) \mathrm{d}\ell' \delta\left(\boldsymbol{x}' - \boldsymbol{x}\right) \delta\left(\boldsymbol{\Omega}' - \boldsymbol{\Omega}\right) \delta\left(E' - E\right) \delta\left(w' - w\right) \mathrm{d}V' \mathrm{d}\Omega' \mathrm{d}E' \mathrm{d}w'. \tag{3.2}$$

The factor of d$\ell'$ preserves unit balance and is not shown from this point forward consistent with prior work and to minimize additional notation.

### 3.1.3 Absorption Kernel

The absorption kernel gives the probability of absorption in d$\boldsymbol{P}'$ about $\boldsymbol{P}'$ (with no ability to re-emerge because absorption is considered terminal for this work). As such, the absorption kernel is given by

$$\mathcal{A}\left(\boldsymbol{P}, \boldsymbol{P}'\right) \mathrm{d}\boldsymbol{P}' = \frac{\Sigma_{\mathrm{a}}\left(\boldsymbol{x}, E\right)}{\Sigma_{\mathrm{t}}\left(\boldsymbol{x}, E\right)} \delta\left(\boldsymbol{x}' - \boldsymbol{x}\right) \delta\left(\boldsymbol{\Omega}' - \boldsymbol{\Omega}\right) \delta\left(E' - E\right) \delta\left(w'\right) \mathrm{d}V' \mathrm{d}\Omega' \mathrm{d}E' \mathrm{d}w' \tag{3.3}$$

where $\Sigma_{\mathrm{a}}(\boldsymbol{x}, E)$ is the macroscopic absorption cross section. Because there is no ability to re-emerge, the resulting weight is set to zero using the Dirac delta function in the emerging weight domain. When integrating over the resulting space, the result is expressed as

$$\mathcal{A}(\boldsymbol{P}) = \frac{\Sigma_{\mathrm{a}}(\boldsymbol{x}, E)}{\Sigma_{\mathrm{t}}(\boldsymbol{x}, E)}, \tag{3.4}$$

which is the absorption probability at position $\boldsymbol{x}$ for energy $E$.

### 3.1.4 Emergence Kernel

The emergence kernel gives the probability of surviving and emerging from a collision into a new phase space with only a change in direction and energy (for an analog case). For this work, the emergence is only associated with scattering, but this kernel could also define events such as fission emergence. The kernel is denoted

$$\mathcal{E}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}' = \frac{\Sigma_{\mathrm{s},0}(\boldsymbol{x}, E)}{\Sigma_{\mathrm{t}}(\boldsymbol{x}, E)}\delta(\boldsymbol{x}' - \boldsymbol{x})\,p(\boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E')\,\delta(w' - w)\,\mathrm{d}V'\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}w' \tag{3.5}$$

where ${}^{\Sigma_{\mathrm{s},0}(\boldsymbol{x},E)}/_{\Sigma_{\mathrm{t}}(\boldsymbol{x},E)}$ is the scattering ratio (i.e., the collision survival probability) and noting that $p(\boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E')$ is the same analog angle-energy emergence probability density function used in the DXTRAN kernel in Section 3.7.1.

### 3.1.5 Surface-crossing Kernel

The surface-crossing kernel gives the probability of crossing a Monte Carlo surface at $\boldsymbol{P}$ and emerging in $\mathrm{d}\boldsymbol{P}'$ about $\boldsymbol{P}'$. In analog games, there is no difference between $\boldsymbol{P}$ and $\boldsymbol{P}'$; however, there are significant differences for non-analog games such as surface-based importance splitting/rouletting. For this work, the analog surface crossing kernel is expressed simply as

$$\mathcal{S}(\boldsymbol{P}, \boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}' = \begin{cases} \delta(\boldsymbol{x}' - \boldsymbol{x})\,\delta(\boldsymbol{\Omega}' - \boldsymbol{\Omega})\,\delta(E' - E)\,\delta(w' - w)\,\mathrm{d}V'\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}w', & \boldsymbol{x} \in \{\boldsymbol{x}_{\text{MC Surfaces}}\} \\ 0, & \text{otherwise} \end{cases}, \tag{3.6}$$

which is used where $\{\boldsymbol{x}_{\text{MC Surfaces}}\}$ is the set of all points on all Monte Carlo geometry surfaces. The analog surface crossing kernel acts as an indicator for when surface crossing occurs preceding contribution to a surface-crossing tally and permits easier bookkeeping when the HSPDFs are described later in Section 3.3.

In addition, this kernel helps provide bookkeeping for when variance reduction is played at surface crossings. For example, in the case of importance splitting and rouletting applied at geometry surfaces (see Section 2.4.5), this kernel precedes the associated biasing kernels. These biasing kernels are described later in Section 3.5.

## 3.2 Summary of Scoring Functions

In addition to the transport kernels described previously, scoring kernels are necessary to define where and how a history may contribute score to a tally. These scoring kernels act as source terms to the HSPDF. For example, one can define $p_{\mathcal{A}}(\boldsymbol{P}, s_{\mathcal{A}})\,\mathrm{d}s_{\mathcal{A}}$ as the probability of contributing score $\mathrm{d}s_{\mathcal{A}}$ about $s_{\mathcal{A}}$ for an absorption at phase-space position $\boldsymbol{P}$ (i.e., where a score is made when an absorption occurs for an absorptive collision tally as described in Section 2.4.2.2). The physical quantity being calculated is the absorption rate.

This scoring function is defined as

$$p_{\mathcal{A}}\left(\boldsymbol{P}, s_{\mathcal{A}}\right) = \delta\left(s_{\mathcal{A}} - w\right) \tag{3.7}$$

given that an absorptive collision occurred so

$$\int_{-\infty}^{\infty} \mathrm{d}s_{\mathcal{A}} p_{\mathcal{A}}\left(\boldsymbol{P}, s_{\mathcal{A}}\right) = 1. \tag{3.8}$$

Similarly for a collision tally, one can define $p_{\mathcal{K}}\left(\boldsymbol{P}, s_{\mathcal{K}}\right) \mathrm{d}s_{\mathcal{K}}$ as the probability of contributing score $\mathrm{d}s_{\mathcal{K}}$ about $s_{\mathcal{K}}$ for a non-absorptive collision at phase-space position $\boldsymbol{P}$ (i.e., where a score is made to a non-absorptive collision tally whenever a collision occurs as described in Section 2.4.2.2). The physical quantity being calculated is the non-absorptive collision rate. This scoring function is defined as

$$p_{\mathcal{K}}\left(\boldsymbol{P}, s_{\mathcal{K}}\right) = \delta\left(s_{\mathcal{K}} - w\right) \tag{3.9}$$

given that a non-absorptive collision occurred so

$$\int_{-\infty}^{\infty} \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}, s_{\mathcal{K}}\right) = 1. \tag{3.10}$$

For a surface-crossing tally, one can define $p_{\mathcal{S}}\left(\boldsymbol{P}, s_{\mathcal{S}}\right) \mathrm{d}s_{\mathcal{S}}$ as the probability of contributing score $\mathrm{d}s_{\mathcal{S}}$ about $s_{\mathcal{S}}$ for a surface crossing at phase-space position $\boldsymbol{P}$ (i.e., where a score is made to a surface-crossing tally whenever the surface is crossed as described in Section 2.4.2.2). The physical quantity being calculated is the surface current. This scoring function is defined as

$$p_{\mathcal{S}}\left(\boldsymbol{P}, s_{\mathcal{S}}\right) = \delta\left(s_{\mathcal{S}} - w\right) \tag{3.11}$$

given that a surface-crossing occurred so

$$\int_{-\infty}^{\infty} \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}\left(\boldsymbol{P}, s_{\mathcal{S}}\right) = 1. \tag{3.12}$$

One should recognize that for the HSMEs the history score is being accounted for and not the behavior of particles, so there is no factor of $|\boldsymbol{\Omega} \cdot \boldsymbol{n}|$ included in these analytic equations. That is, when a Monte Carlo particle crosses a surface upon which there is a current tally specified, the associated history's score increases by the weight of the particle crossing without respect to the particle's orientation relative to the surface [Section 2.4.2.1].

Finally, for an expected track-length tally, one can define $p_{\mathcal{T}}\left(\boldsymbol{P}, s_{\mathcal{T}}\right) \mathrm{d}s_{\mathcal{T}}$ as the probability of contributing score $\mathrm{d}s_{\mathcal{T}}$ about $s_{\mathcal{T}}$ for the expected track length generated at phase-space position $\boldsymbol{P}$ (i.e., where a score is made to an expected track-length tally whenever track length is deterministically generated as described in Section 2.4.2.3). The physical quantity being calculated is the expected track length generated in the cell.

This scoring function is defined as

$$p_{\mathcal{T}}\left(\boldsymbol{P}, s_{\mathcal{T}}\right)=\delta\left(s_{\mathcal{S}}-\frac{w}{\Sigma_{\mathrm{t}}\left(\boldsymbol{x}\right)}\left[1-\exp\left(-d\left(\boldsymbol{x}, \boldsymbol{\Omega}\right) \cdot\left(\boldsymbol{x}\right)\right)\right]\right), \tag{3.13}$$

where $\Sigma_{\mathrm{t}}\left(\boldsymbol{x}\right)$ is the macroscopic total cross section at position $\boldsymbol{x}$ and $d\left(\boldsymbol{x}\right)$ is the distance to the cell boundary from $\boldsymbol{x}$ along $\boldsymbol{\Omega}$. As with the other scoring functions,

$$\int_{-\infty}^{\infty} \mathrm{d}s_{\mathcal{T}} p_{\mathcal{T}}\left(\boldsymbol{P}, s_{\mathcal{T}}\right)=1. \tag{3.14}$$

Note: all of these scoring functions assume that the total collision rate and track lengths are being tallied (consistent with Section 2.4.2).

## 3.3 Analog History-score Probability Density Functions

By combining the previously defined transport and scoring kernels, partial HSPDFs can be defined that give the probability of a history and any progeny contributing score $\mathrm{d}s$ about $s$ from a region of phase space $\boldsymbol{P}_0$ for various Monte Carlo random walks. Examples of doing this are given next for random walks terminating with absorption and random walks resulting in surface crossing and non-absorptive collision that then continue. These three individual HSPDFs are then combined into the total HSPDF.

### 3.3.1 Analog Absorption History-score Probability Density Function

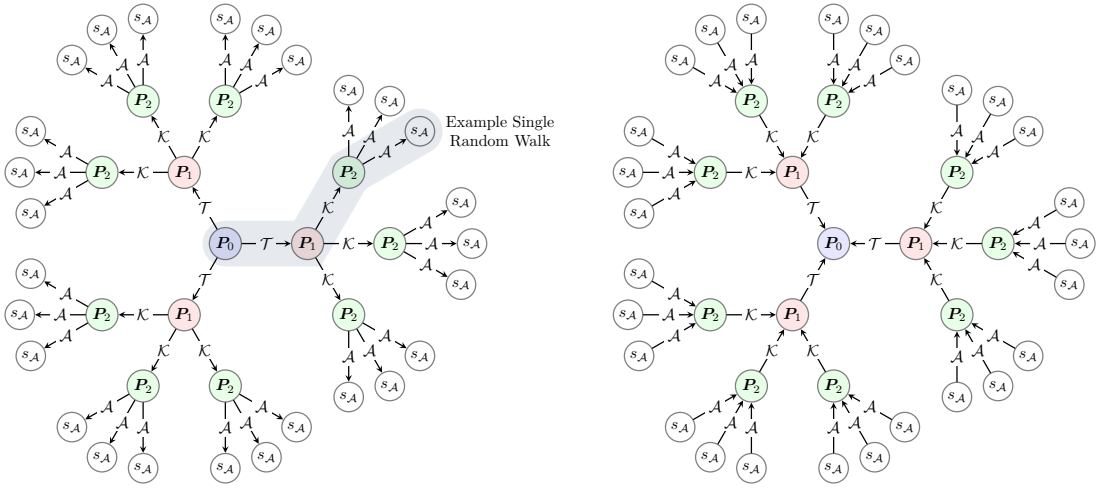
To analytically represent the analog random walk transport progression from phase-space $\boldsymbol{P}_0$, which includes free-flight to an absorptive collision followed by scoring and termination, one can write

$$\begin{aligned}&\psi_{\mathcal{A}}\left(\boldsymbol{P}_0, s\right) \mathrm{d}s\\ &\quad=\int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \mathcal{A}\left(\boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{A}} p_{\mathcal{A}}\left(\boldsymbol{P}_2, s_{\mathcal{A}}\right) \delta\left(s-s_{\mathcal{A}}\right) \mathrm{d}s.\end{aligned} \tag{3.15}$$

Equation (3.15) is the HSPDF (i.e., the random walk, comparable to a path integral through phase space) for absorption.

One way to interpret Eq. (3.15) is in the forward sense: by reading from left to right. That is, a particle can undergo free flight from $\boldsymbol{P}_0$ to $\boldsymbol{P}_1$, undergo collision at $\boldsymbol{P}_1$ resulting in $\boldsymbol{P}_2$, and being absorbed at $\boldsymbol{P}_2$ while also contributing a score $\mathrm{d}s$ about $s$. Conversely, one can read Eq. (3.15) in an adjoint sense: from right to left. That is, one can integrate over all absorptive scores and $\boldsymbol{P}_2$ to collect all scores caused by absorptive collision and then integrate over $\boldsymbol{P}_1$ to move the particles "backwards" in flight to $\boldsymbol{P}_0$ to determine their probability of contributing those scores.

Figure 3.1 provides an illustration of these two interpretations. Figure 3.1a shows a hypothetical random walk that terminates with absorptive scoring among the infinite random walks for a history emitted from a forward source at phase-space position $\boldsymbol{P}_0$. Making multiple samples of this phase space by following multiple histories attempts to sample the phase space to determine the absorptive collision rate. Conversely,




**(a)** Forward **(b)** Adjoint

**Figure 3.1:** Forward and Adjoint Interpretation of the Monte Carlo Random Walk Terminating with Absorption

Fig. 3.1b shows how the probability of all absorptive scores throughout all phase-space position $\boldsymbol{P}_2$ can be integrated back through the collision and free-flight transmission kernels to accumulate the probability of absorptive scoring from phase-space position $\boldsymbol{P}_0$. Recognize that the direction of the arrows has been reversed to represent integration from absorptive scores to $\boldsymbol{P}_0$.

### 3.3.2 Analog Surface-crossing and Non-absorptive Collision History-score Probability Density Function

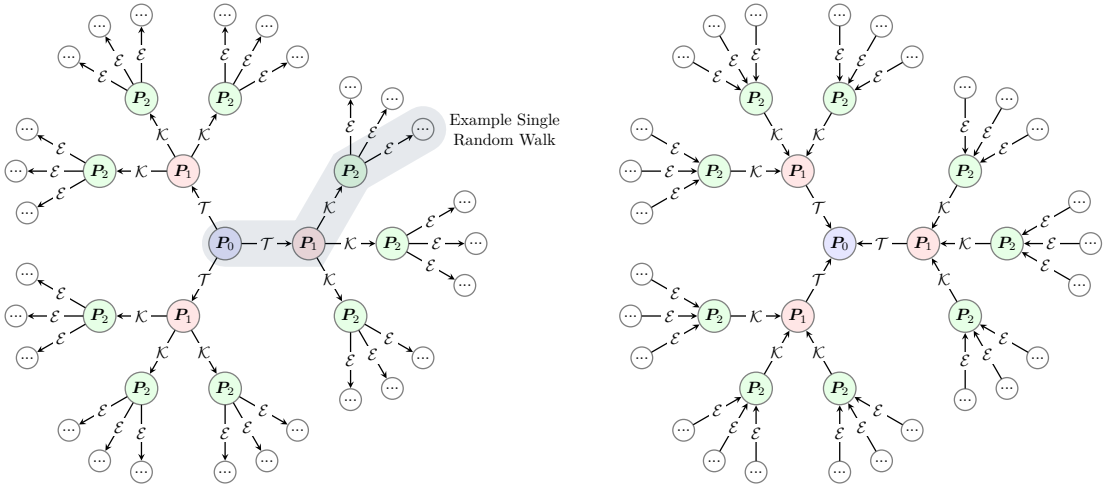
For an analog calculation, the two other transport progressions that a particle can undergo as considered herein are flight to and through Monte Carlo geometry surfaces and/or flight to a non-absorptive collision followed by emergence (scattering). In both of these cases, the particle can contribute (e.g., to a surface tally or to a collision tally) but can continue on to contribute to the same or another tally before terminating. Thus, one must make the distinction between a contribution (registering with a tally, which a history can do more than once) and a history's score (the total of all tally contributions for a history). The HSPDFs associated with surface crossing and analog non-absorptive collision are

$$\psi_{\mathcal{S}}\left(\boldsymbol{P}_0, s\right) \mathrm{d}s = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{S}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}\left(\boldsymbol{P}_3, s_{\mathcal{S}}\right) \psi\left(\boldsymbol{P}_3, s - s_{\mathcal{S}}\right) \mathrm{d}s, \quad (3.16)$$

and

$$\psi_{\mathcal{E}}\left(\boldsymbol{P}_0, s\right) \mathrm{d}s = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right)$$
$$\times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \psi\left(\boldsymbol{P}_3, s - s_{\mathcal{K}}\right) \mathrm{d}s, \quad (3.17)$$

respectively. The terms $s - s_{\mathcal{S}}$ and $s - s_{\mathcal{K}}$ account for the earlier contributions by the history to the tally (to a surface-crossing and collision estimator, respectively) such that the history can only go on from $\boldsymbol{P}_3$ to contribute a score that makes up the difference between the history's total score $s$ and that which was




**(a)** Forward **(b)** Adjoint

**Figure 3.2:** Forward and Adjoint Interpretation of the Monte Carlo Emergence Random Walk

already contributed (i.e., $s_{\mathcal{S}}$ or $s_{\mathcal{K}}$).

Note that Eqs. (3.16) and (3.17) are recursive in nature (like the integral transport equations described in Section 2.2.4). As such, either of these two transport progressions can be followed by any of the three transport progressions described in their next random walk: absorption, another surface crossing, (exclusive) or another emergence, and so on until termination (leakage or absorption).

Figure 3.2a gives a forward interpretation (without scoring) of Eq. (3.17) read left to right where a particle starts at phase-space position $\boldsymbol{P}_0$ and undergoes free flight followed by collision at phase-space position $\boldsymbol{P}_1$ according to the kernel $\mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1)$. The particle's collision is then processed according to the collision kernel, $\mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2)$. Finally, the particle's non-absorptive emergence from the collision will be processed from $\boldsymbol{P}_2$ to $\boldsymbol{P}_3$ according to $\mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3)$. This random walk has sampled one of the possible ways that the particle can evolve through phase space and is shown, for example, as the highlighted path in Figure 3.2a that then continues on to another random walk (indicated by "...") until it eventually terminates. Following many histories will sample the other available random walks.

However, Fig. 3.2b demonstrates the adjoint interpretation of Eq. (3.17) read right to left. Recognize that the direction of the arrows has been reversed. In the adjoint sense, all possible random walks are integrated back through their emergence from phase-space $\boldsymbol{P}_2$ into phase-space $\boldsymbol{P}_2$. They are all then integrated back through their collision processing into phase-space $\boldsymbol{P}_1$, and then through their collision and free flight into phase-space $\boldsymbol{P}_0$. In this way, the effect of contributing score to a tally at any future point in phase-space reached from passing through $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, and so on has been integrated back into $\boldsymbol{P}_0$. It is through these integrations that the probability of scoring d$s$ about $s$ from phase-space $\boldsymbol{P}_0$ is calculated.

### 3.3.3 Combined History-score Probability Density Function

By recognizing that the HSPDFs in Eqs. (3.15)–(3.17) are independent (because the individual random walks are Markov processes), one can construct the complete HSPDF as

$$\psi\left(\boldsymbol{P}_0, s\right) \mathrm{d}s = \psi_{\mathcal{A}}\left(\boldsymbol{P}_0, s\right) \mathrm{d}s + \psi_{\mathcal{S}}\left(\boldsymbol{P}_0, s\right) \mathrm{d}s + \psi_{\mathcal{E}}\left(\boldsymbol{P}_0, s\right) \mathrm{d}s. \tag{3.18}$$

This complete HSPDF indicates that the probability of a history and any progeny contributing score $\mathrm{d}s$ about $s$ from phase-space position $\boldsymbol{P}_0$ is a result of scoring as a result of being absorbed and then terminating, *or* scoring as a result of crossing a surface tally and then continuing the random walk to contribute other scores until terminating, *or* scoring as a result of non-absorptive collision tally and then continuing the random walk to contribute other scores until terminating.

The $m^{\text{th}}$ statistical moment of the HSPDF is calculated as

$$\Psi_m\left(\boldsymbol{P}_0\right) = \int \mathrm{d}s\, s^m \psi\left(\boldsymbol{P}_0, s\right). \tag{3.19}$$

This quantity is the HSME associated with the random walk processes considered and is a phase-space density giving the statistical moments of the tally response. The HSMEs can then be integrated over phase space with a certain forward source $Q\left(\boldsymbol{P}_0\right)$ to determine the integral tally statistical response as

$$M_m = \int \mathrm{d}\boldsymbol{P}_0 \Psi_m\left(\boldsymbol{P}_0\right) Q\left(\boldsymbol{P}_0\right). \tag{3.20}$$

As such, the HSMEs calculate tally statistical moments via integration that are comparable to the tally statistical moments accumulated by summing tally history scores as described in Section 2.4.3. Thus, the HSMEs provide a mechanism to analytically (for simple cases) or numerically predict Monte Carlo tally statistical moments. Note that Eq. (3.20) for $m = 1$ is equivalent to Eq. (2.25).

The calculation of these moment equations is described next in Section 3.4 for analog HSMEs and in Section 3.7 for HSMEs incorporating the simplified DXTRAN behavior described in Section 2.4.6.3.

## 3.4 Analog History-score Moment Equations

As noted previously, the analog HSMEs are evaluated explicitly for two purposes. First, it provides an illustration of the score-moment integration process described by Eq. (3.19). This provides a mathematical formalism to describe the associated transport processes effect on weight. Second, it provides a basis for comparison once variance-reduction techniques are introduced. In particular, when calculating $\Psi_1\left(\boldsymbol{P}_0\right)$ with variance reduction incorporated, the result should be the same as $\Psi_1\left(\boldsymbol{P}_0\right)$ using a strictly analog approach (as long as the variance-reduction techniques are fair).

Because of the linearity of the terms in the HSPDF arising from the independence of the random walks described by the HSPDF, each moment for each random walk is considered individually in the upcoming sections. This helps segregate how the components of the random walks in the HSPDF form the HSMEs and helps ease notational burden.

As a terminal event, HSMEs of the HSPDF representing the random walk terminating with absorption is considered first. Then surface crossing is considered followed by non-absorptive collision with scattering emergence.

### 3.4.1 First Moment of Absorption HSPDF

To compute the first moment of the absorption portion of the HSPDF, integrate Eq. (3.15) according to Eq. (3.19) as

$$\begin{aligned}
\Psi_{1,\mathcal{A}}\left(\boldsymbol{P}_0\right) &= \int_{-\infty}^{\infty} \mathrm{d}s\, s\, \psi_{\mathcal{A}}\left(\boldsymbol{P}_0, s\right) \\
&= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{A}} p_{\mathcal{A}}\left(\boldsymbol{P}_2, s_{\mathcal{A}}\right) \mathcal{A}\left(\boldsymbol{P}_2\right) \\
&\quad \times \int_{-\infty}^{\infty} \mathrm{d}s\, s\, \delta\left(s - s_{\mathcal{A}}\right) \\
&= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{A}} p_{\mathcal{A}}\left(\boldsymbol{P}_2, s_{\mathcal{A}}\right) \mathcal{A}\left(\boldsymbol{P}_2\right) s_{\mathcal{A}} \\
&= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_A\left(\boldsymbol{P}_2, s\right) \mathcal{A}\left(\boldsymbol{P}_2\right), \qquad (3.21)
\end{aligned}$$

where $s_{\mathcal{A}}$ was replaced with $s$. Note that $\int \mathrm{d}s\, s\, p_{\mathcal{A}}\left(\boldsymbol{P}_2, s\right)$ is the expected (mean) score from an absorption at $\boldsymbol{P}_2$.

Thus, the first statistical moment of the absorption HSPDF is determined by the mean score from absorption (e.g., from an absorptive collision estimator) at $\boldsymbol{P}_2$ integrated over all collisions resulting in absorption in phase space element $\mathrm{d}\boldsymbol{P}_2$ about $\boldsymbol{P}_2$ that entered the absorptive collision from phase space $\boldsymbol{P}_1$. In turn, the random walks that resulted in $\boldsymbol{P}_1$ are integrated over all $\mathrm{d}\boldsymbol{P}_1$ about $\boldsymbol{P}_1$ along all free-flight transmission paths to the originating position $\boldsymbol{P}_0$. Read in a forward sense, a particle starts at $\boldsymbol{P}_0$ and undergoes free-flight until $\boldsymbol{P}_1$ at which point it collides with absorption, scores $\mathrm{d}s$ about $s$, and is terminated.

### 3.4.2 Second Moment of Absorption HSPDF

For the second statistical moment of the absorption portion of the HSPDF, integrate again using Eq. (3.19) with $s^2$ as

$$\begin{aligned}
\Psi_{2,\mathcal{A}}(\boldsymbol{P}_0) &= \int_{-\infty}^{\infty} \mathrm{d}s\, s^2 \psi_{\mathcal{A}}(\boldsymbol{P}_0, s) \\
&= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s_A p_A(\boldsymbol{P}_2, s_A) A(\boldsymbol{P}_2) \\
&\quad \times \int_{-\infty}^{\infty} \mathrm{d}s\, s^2 \delta(s - s_{\mathcal{A}}) \\
&= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s_{\mathcal{A}} p_{\mathcal{A}}(\boldsymbol{P}_2, s_{\mathcal{A}}) \mathcal{A}(\boldsymbol{P}_2) s_{\mathcal{A}}^2 \\
&= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s\, s^2 p_{\mathcal{A}}(\boldsymbol{P}_2, s) \mathcal{A}(\boldsymbol{P}_2)
\end{aligned} \tag{3.22}$$

and note that $\int \mathrm{d}s\, s^2\, p_{\mathcal{A}}(\boldsymbol{P}_2, s)$ is the second moment of the score from an absorption at $\boldsymbol{P}_2$. The adjoint-forward interpretations described in the previous section are the same except now the history scores $s^2$ when it is tallied upon absorption at $\boldsymbol{P}_2$.

### 3.4.3 First Moment of Surface Crossing HSPDF

To compute the first moment of the surface crossing portion of the HSPDF, integrate according to Eq. (3.19) again as:

$$\begin{aligned}
\Psi_{1,\mathcal{S}}(\boldsymbol{P}_0) &= \int_{-\infty}^{\infty} \mathrm{d}s\, s\, \psi_{\mathcal{S}}(\boldsymbol{P}_0, s) \\
&= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) \int_{-\infty}^{\infty} \mathrm{d}s\, s\, \psi(\boldsymbol{P}_2, s - s_{\mathcal{S}}).
\end{aligned} \tag{3.23}$$

To continue, make the substitution $q = s - s_{\mathcal{S}}$ so that $s = q + s_{\mathcal{S}}$ and $\mathrm{d}s = \mathrm{d}q$ to give

$$\begin{aligned}
\Psi_{1,\mathcal{S}}(\boldsymbol{P}_0) &= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) \\
&\quad \times \int_{-\infty}^{\infty} \mathrm{d}q\, (q + s_{\mathcal{S}})\, \psi(\boldsymbol{P}_2, q) \\
&= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\
&\quad \times \left[ \int \mathrm{d}s_{\mathcal{S}}\, s_{\mathcal{S}}\, p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) \int_{-\infty}^{\infty} \mathrm{d}q\, \psi(\boldsymbol{P}_2, q) \right. \\
&\quad \left. + \int \mathrm{d}s_{\mathcal{S}}\, p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) \int_{-\infty}^{\infty} \mathrm{d}q\, q\, \psi(\boldsymbol{P}_2, q) \right],
\end{aligned} \tag{3.24}$$

where

$$\int_{-\infty}^{\infty} \mathrm{d}q\, \psi\left(\boldsymbol{P}, q\right) = 1 \tag{3.25}$$

by virtue of being a probability density in $q$ and

$$\int_{-\infty}^{\infty} \mathrm{d}s\, s\, \psi\left(\boldsymbol{P}, s\right) = \Psi_1\left(\boldsymbol{P}\right), \tag{3.26}$$

by definition. As such, the first moment of the surface crossing HSPDF becomes

$$\begin{aligned}\Psi_{1,\mathcal{S}}\left(\boldsymbol{P}_0\right) = &\int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right) \\ &+ \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \Psi_1\left(\boldsymbol{P}_2\right),\end{aligned} \tag{3.27}$$

where $s_{\mathcal{S}}$, as a placeholder, is replaced by $s$. Thus, the expected score for a history at phase space $\boldsymbol{P}_0$ attributable to surface crossing can arise from a surface-crossing estimator contributing to a tally directly (the top line of Eq. (3.27)) or a history at some point in its random walk contributing to a surface-crossing estimator (the bottom line of the previous equation). As noted previously, this integral equation is recursive in nature where the expected score at $\boldsymbol{P}_0$ attributable to surface crossing depends both on surface-crossing estimator contributions at $\boldsymbol{P}_2$ and future contributions as a result of the history crossing the surface at $\boldsymbol{P}_2$ continuing on as represented by $\Psi_1\left(\boldsymbol{P}_2\right)$ to have any one of the three additional random walks described earlier.

### 3.4.4 Second Moment of Surface Crossing HSPDF

To compute the second moment of the surface crossing portion of the HSPDF, integrate according to Eq. (3.19) with $s^2$ again as:

$$\begin{aligned}\Psi_{2,\mathcal{S}}\left(\boldsymbol{P}_0\right) &= \int_{-\infty}^{\infty} \mathrm{d}s\, s^2 \psi_{\mathcal{S}}\left(\boldsymbol{P}_0, s\right) \\ &= \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}\left(\boldsymbol{P}_2, s_{\mathcal{S}}\right) \\ &\quad \times \int_{-\infty}^{\infty} \mathrm{d}s\, s^2 \psi\left(\boldsymbol{P}_2, s - s_{\mathcal{S}}\right).\end{aligned} \tag{3.28}$$

Again make the substitution $q = s - s_{\mathcal{S}}$ so that $s = q + s_{\mathcal{S}}$ and $\mathrm{d}s = \mathrm{d}q$ to give

$$\begin{aligned}\Psi_{2,\mathcal{S}}\left(\boldsymbol{P}_0\right) = &\int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}\left(\boldsymbol{P}_2, s_{\mathcal{S}}\right) \\ &\times \int_{-\infty}^{\infty} \mathrm{d}q\, \left(q + s_{\mathcal{S}}\right)^2 \psi\left(\boldsymbol{P}_2, q\right)\end{aligned} \tag{3.29}$$

and expand the binomial as $(q+s_{\mathcal{S}})^2 = q^2 + s_{\mathcal{S}}^2 + 2qs_{\mathcal{S}}$ to obtain

$$\begin{aligned}\Psi_{2,\mathcal{S}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\ \times \left[ \int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) \int_{-\infty}^{\infty} \mathrm{d}q\, q^2 \psi(\boldsymbol{P}_2, q) \right. \\ + \int \mathrm{d}s_{\mathcal{S}} s_{\mathcal{S}}^2 p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) \int_{-\infty}^{\infty} \mathrm{d}q \psi(\boldsymbol{P}_2, q) \\ \left. +2 \int \mathrm{d}s_{\mathcal{S}} s_{\mathcal{S}} p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) \int_{-\infty}^{\infty} \mathrm{d}q\, q \psi(\boldsymbol{P}_2, q) \right], \end{aligned} \tag{3.30}$$

where

$$\int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) = 1, \tag{3.31a}$$

$$\int_{-\infty}^{\infty} \mathrm{d}q\, q^2 \psi(\boldsymbol{P}_2, q) = \Psi_2(\boldsymbol{P}_2), \tag{3.31b}$$

$$\int_{-\infty}^{\infty} \mathrm{d}q\, \psi(\boldsymbol{P}_2, q) = 1, \tag{3.31c}$$

$$\int_{-\infty}^{\infty} \mathrm{d}q\, q\, \psi(\boldsymbol{P}_2, q) = \Psi_1(\boldsymbol{P}_2), \tag{3.31d}$$

so

$$\begin{aligned}\Psi_{2,\mathcal{S}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\ \times \left[ \Psi_2(\boldsymbol{P}_2) + \int \mathrm{d}s_{\mathcal{S}} s_{\mathcal{S}}^2 p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) + 2 \int \mathrm{d}s_{\mathcal{S}} s_{\mathcal{S}} p_{\mathcal{S}}(\boldsymbol{P}_2, s_{\mathcal{S}}) \Psi_1(\boldsymbol{P}_2) \right] \end{aligned} \tag{3.32}$$

or

$$\begin{aligned}\Psi_{2,\mathcal{S}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \Psi_2(\boldsymbol{P}_2) \\ + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s\, s^2 p_{\mathcal{S}}(\boldsymbol{P}_2, s) \\ + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s\, s\, p_{\mathcal{S}}(\boldsymbol{P}_2, s) \Psi_1(\boldsymbol{P}_2), \end{aligned} \tag{3.33}$$

where $s_{\mathcal{S}}$ is again replaced by $s$.

This result merits discussion for several reasons. First, as a second-moment order equation, all components should be second-moment order. This is the case because each of three additive terms result in second-order quantities: $\Psi_2(\boldsymbol{P}_2)$, $\int \mathrm{d}s\, s^2 p_{\mathcal{S}}(\boldsymbol{P}_2, s)$, or $\int \mathrm{d}s\, s\, p_{\mathcal{S}}(\boldsymbol{P}_2, s) \Psi_1(\boldsymbol{P}_2)$.

Second, the last term is notable because it is second-moment order by virtue of being the product of two

first-moment ordered quantities: the expected contribution $\mathrm{d}s$ about $s$ to a surface estimator and the expected contribution from random walks that continue on from $\boldsymbol{P}_2$. This is a joint-scoring term that couples the second-moment order HSME to the first-moment order HSME. Indeed, if one continues to calculate higher moments such coupling terms arise to bind higher-moment HSMEs to all lower-moment HSMEs.

As final analytical results are collected, terms independent of the HSME under consideration are segregated into separate source terms. For example, Eq. (3.33) can be rewritten as

$$\Psi_{2,\mathcal{S}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \Psi_2\left(\boldsymbol{P}_2\right) + Q_{2,\mathcal{S}}\left(\boldsymbol{P}_0\right), \tag{3.34}$$

$$\begin{aligned} Q_{2,\mathcal{S}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s^2 p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right) \\ & + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right) \Psi_1\left(\boldsymbol{P}_2\right). \end{aligned} \tag{3.35}$$

### 3.4.5 First Moment of Scattering Emission HSPDF

To compute the first moment of the non-absorptive collision followed by analog scattering emergence portion of the HSPDF, integrate according to Eq. (3.19) as:

$$\begin{aligned} \Psi_{1,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int_{-\infty}^{\infty} \mathrm{d}s\, s\, \psi_{\mathcal{E}}\left(\boldsymbol{P}_0, s\right) \\ = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \\ & \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int_{-\infty}^{\infty} \mathrm{d}s\, s\, \psi\left(\boldsymbol{P}_3, s - s_{\mathcal{K}}\right). \end{aligned} \tag{3.36}$$

Again make the substitution $q = s - s_{\mathcal{K}} \implies s = q + s_{\mathcal{K}}$ and $\mathrm{d}q = \mathrm{d}s$ to obtain

$$\begin{aligned} \Psi_{1,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \\ & \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int_{-\infty}^{\infty} \mathrm{d}q\, \left(q + s_{\mathcal{K}}\right) \psi\left(\boldsymbol{P}_3, q\right). \end{aligned} \tag{3.37}$$

Distribute to obtain

$$\begin{aligned} \Psi_{1,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\ & \times \left[ \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int_{-\infty}^{\infty} \mathrm{d}q\, \psi\left(\boldsymbol{P}_3, q\right) \right. \\ & \left. + \int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int_{-\infty}^{\infty} \mathrm{d}q\, q\, \psi\left(\boldsymbol{P}_3, q\right) \right] \end{aligned} \tag{3.38}$$

where $\int_{-\infty}^{\infty} \mathrm{d}q\, \psi\left(\boldsymbol{P}_3, q\right) = 1$ so in turn $\int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) = 1$ and again $\int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) = 1$ so

$$\begin{aligned}\Psi_{1,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\ & \times \left[\int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) + \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right)\right]\end{aligned} \tag{3.39}$$

or

$$\begin{aligned}\Psi_{1,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right) \\ & + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right)\end{aligned} \tag{3.40}$$

with $s_{\mathcal{K}}$ replaced with $s$. This result has a similar form as for surface crossing previously (except now a non-absorptive collision estimator is considered), so the earlier discussion is not repeated.

### 3.4.6 Second Moment of Scattering Emission HSPDF

To compute the second moment of the non-absorptive collision followed by analog scattering emergence portion of the HSPDF, integrate according to Eq. (3.19) with $s^2$ again as:

$$\begin{aligned}\Psi_{2,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int_{-\infty}^{\infty} s^2 \psi_{\mathcal{E}}\left(\boldsymbol{P}_0, s\right) \mathrm{d}s \\ = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \\ & \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int_{-\infty}^{\infty} \mathrm{d}s\, s^2\, \psi\left(\boldsymbol{P}_3, s - s_{\mathcal{K}}\right)\end{aligned} \tag{3.41}$$

and make the substitution $q = s - s_{\mathcal{K}} \implies s = q + s_{\mathcal{K}}$ with $\mathrm{d}q = \mathrm{d}s$ and distribute the resulting binomial $\left(q + s_{\mathcal{K}}\right)^2 = q^2 + s_{\mathcal{K}}^2 + 2 q s_{\mathcal{K}}$ to obtain

$$\begin{aligned}\Psi_{2,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\ & \times \left[\int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int_{-\infty}^{\infty} \mathrm{d}q\, q^2\, \psi\left(\boldsymbol{P}_3, q\right)\right. \\ & + \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}}^2 p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \\ & \left. + 2 \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int_{-\infty}^{\infty} \mathrm{d}q\, q\, \psi\left(\boldsymbol{P}_3, q\right)\right].\end{aligned} \tag{3.42}$$

Making the simplifications demonstrated previously yields

$$\begin{aligned}\Psi_{2,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\ & \times \Bigg[ \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_2\left(\boldsymbol{P}_3\right) \\ & + \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}}^2 p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \\ & + 2 \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \Bigg] \end{aligned} \tag{3.43}$$

or

$$\begin{aligned}\Psi_{2,\mathcal{E}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_2\left(\boldsymbol{P}_3\right) \\ & + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}}^2 p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \\ & + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \end{aligned} \tag{3.44}$$

### 3.4.7 Summary

When the analog history-score moment equations for all three random walk paths are combined and then in turn separated as described in Section 3.4.4, one obtains:

$$\begin{aligned}\Psi_1\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \Psi_1\left(\boldsymbol{P}_2\right) \\ & + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \\ & + Q_1\left(\boldsymbol{P}_0\right), \end{aligned} \tag{3.45a}$$

$$\begin{aligned}Q_1\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \mathcal{A}\left(\boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{A}}\left(\boldsymbol{P}_2, s\right) \\ & + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right) \\ & + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right), \end{aligned} \tag{3.45b}$$

and

$$\begin{aligned}\Psi_2\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \Psi_2\left(\boldsymbol{P}_2\right) \\ & + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_2\left(\boldsymbol{P}_3\right) \\ & + Q_2\left(\boldsymbol{P}_0\right), \end{aligned} \tag{3.46a}$$

$$
\begin{aligned}
Q_2\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \mathcal{A}\left(\boldsymbol{P}_2\right) \int \mathrm{d}s\, s^2\, p_{\mathcal{A}}\left(\boldsymbol{P}_2, s\right) \\
& + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s^2\, p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \Psi_1\left(\boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right) \\
& + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s^2\, p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right) \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right).
\end{aligned}
\tag{3.46b}
$$

Equation (3.45) is equivalent to the standard adjoint integral transport equation. That is, it gives the expected score of a history at $\boldsymbol{P}_0$ as a result of any absorptive collision, surface crossing, and non-absorptive collision estimators (via $Q_1\left(\boldsymbol{P}_0\right)$). If a particular problem doesn't feature a particular estimator, that component of $Q_1\left(\boldsymbol{P}_0\right)$ is zero. Regardless, $\Psi_1\left(\boldsymbol{P}_0\right)$ is used to compute the first moment, $M_1$, of the integral detector response using Eq. (3.20).

However, Eq. (3.46) represents information not captured by standard integral adjoint transport techniques. It directly computes the second moment contribution to a tally as a function of phase-space position $\boldsymbol{P}_0$. It is this equation that variance reduction modifies and is used to compute the second moment, $M_2$, of the integral detector response, also using Eq. (3.20).

Chapter 4 will discuss the FTE, which captures the effect of variance reduction on computer time and is used to calculate the computer time, $T$, associated with a given integral detector response.

### 3.5 Importance-splitting and Rouletting Transport Kernels and HSPDF

Before discussing DXTRAN variance reduction, an example of integer importance splitting and rouletting variance reduction applied at Monte Carlo geometry surfaces is given. Following from the discussion in Section 2.4.5, the integer importance splitting biasing kernel is defined as

$$
\mathcal{B}_{\mathrm{IS}}\left(\boldsymbol{P}, \boldsymbol{P}'\right) \mathrm{d}\boldsymbol{P}' = \delta\left(\boldsymbol{x}' - \boldsymbol{x}\right) \delta\left(\boldsymbol{\Omega}' - \boldsymbol{\Omega}\right) \delta\left(E' - E\right) \delta\left(w' - \frac{w}{k_{\mathrm{IS}}}\right) \mathrm{d}V' \mathrm{d}\Omega' \mathrm{d}E' \mathrm{d}w' \tag{3.47}
$$

and the associated importance rouletting biasing kernel is defined as

$$
\begin{aligned}
\mathcal{B}_{\mathrm{IR}}\left(\boldsymbol{P}, \boldsymbol{P}'\right) = & \delta\left(\boldsymbol{x}' - \boldsymbol{x}\right) \delta\left(\boldsymbol{\Omega}' - \boldsymbol{\Omega}\right) \delta\left(E' - E\right) \\
& \times \left[\alpha_{\mathrm{IR}} \delta\left(w' - \frac{w}{\alpha_{\mathrm{IR}}}\right) + \left(1 - \alpha_{\mathrm{IR}}\right) \delta\left(w'\right)\right] \mathrm{d}V' \mathrm{d}\Omega' \mathrm{d}E' \mathrm{d}w'.
\end{aligned}
\tag{3.48}
$$

Equation 3.47 indicates that as a result of splitting all $k_{\mathrm{IS}}$ progeny emerge with new statistical weight $w' = w/k_{\mathrm{IS}}$. Equation 3.48 indicates that as a result of rouletting with probability $\alpha_{\mathrm{IR}}$ the particle emerges with weight $w' = w/\alpha_{\mathrm{IR}}$ and with probability $1 - \alpha_{\mathrm{IR}}$ the particle is killed.

Both of these biasing kernels are applied in the surface-crossing HSPDF following the surface crossing kernel based on the direction of surface crossing if the cells separated by the Monte Carlo surface have different importances. If so, the importance-splitting HSPDF is written as

$$\psi_{\mathrm{IS}}\left(\boldsymbol{P}_0, s\right)\mathrm{d}s = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{S}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}\left(\boldsymbol{P}_3, s_{\mathcal{S}}\right) \times \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{IS}}\left(\boldsymbol{P}_3, \boldsymbol{P}_4\right) \prod_{p=1}^{k_{\mathrm{IS}}-1} \left[\int \mathrm{d}s_p \psi\left(\boldsymbol{P}_4, s - s_p\right)\right] \psi\left(\boldsymbol{P}_4, s - \sum_{p=1}^{k_{\mathrm{IS}}-1} s_p - s_{\mathcal{S}}\right) \mathrm{d}s. \quad (3.49)$$

The product in Eq. (3.49) gives the score contributed by random walks followed by progeny $p = 1, 2, \ldots, k_{\mathrm{IS}} - 1$ and the term $\psi\left(\boldsymbol{P}_3, s - \sum_{p=1}^{k_{\mathrm{IS}}-1} s_p\right)$ represents the random walk of the ${k_{\mathrm{IS}}}^{\mathrm{th}}$ split particle, which contributes the balance of the score for the history. For all particles in phase-space $\boldsymbol{P}_4$, $w_4 \neq w_3 = w_2 = w_1 = w_0$.

Conversely, the importance-rouletting kernel is included in the associated importance-rouletting HSPDF as

$$\psi_{\mathrm{IR}}\left(\boldsymbol{P}_0, s\right)\mathrm{d}s = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{S}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int \mathrm{d}s_{\mathcal{S}} p_{\mathcal{S}}\left(\boldsymbol{P}_3, s_{\mathcal{S}}\right) \times \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{IR}}\left(\boldsymbol{P}_3, \boldsymbol{P}_4\right) \psi\left(\boldsymbol{P}_4, s - s_{\mathcal{S}}\right) \mathrm{d}s. \quad (3.50)$$

Equation (3.50) indicates that only one particle will emerge from importance rouletting at phase-space $\boldsymbol{P}_4$ and will go on to contribute score $\mathrm{d}s$ about $s - s_{\mathcal{S}}$ similar to analog surface crossing. However, by virtue of having undergone rouletting the particle at $\boldsymbol{P}_4$ will have $w_4 \neq w_3 = w_2 = w_1 = w_0$.

More information regarding how these kernels are applied, including proofs of fairness, is available in the work by Booth, Juzaitis, and Solomon [42, 43, 58–60].

### 3.6 Weight-independent vs. Weight-dependent Variance Reduction

As described in Section 2.4.5 and Section 3.5, importance splitting and rouletting are weight-independent variance-reduction techniques. That is, the techniques modify particle weight but are applied regardless of the incident weight of the particle.

When attempting to model Monte Carlo random walks with the HSMEs, the difficulty in solving for and implementing the HSMEs is intimately tied to the weight-dependence of the variance-reduction technique being studied. For strictly weight-dependent variance-reduction techniques it has been shown [42] that the weight domain in the weight-augmented phase space can be systematically separated such that the HSMEs can be scaled as

$$\Psi_m\left(\boldsymbol{R}, aw\right) = a^m \Psi_m\left(\boldsymbol{R}, w\right) = a^m w^m \Psi_m\left(\boldsymbol{R}, w = 1\right). \quad (3.51)$$

The ability to separate weight avoids the need to discretize the weight domain into a weight mesh (e.g., as Solomon needed to do in [58, 59]). Obviating the need for a weight mesh simplifies the numerical algorithms and reduces computer memory requirements and solver time. In turn, this makes implementing the HSMEs into pre-existing deterministic solvers more straightforward. It is for these reasons that the simplification is made in Section 2.4.6.3 to remove weight-dependence by eliminating weight-rouletting during transmission.

### 3.7 DXTRAN History-Score Moment Equations

Having examined the analog HSMEs and demonstrated variance-reduction biasing kernels for importance splitting and rouletting, the DXTRAN biasing kernels, HSPDF, and HSMEs are considered next. First, the two additional transport kernels arising from DXTRAN are deduced. Next, the random walk (partial HSPDF) that incorporates DXTRAN is formulated. Finally, the first two statistical moments of the HSPDF (the first- and second-moment HSMEs) are calculated using Eq. (3.19) similar to the analog cases.

#### 3.7.1 DXTRAN Kernel

The DXTRAN transport biasing kernel associated with free-flight from the initiating point to the DXTRAN surface and creation of the DXTRAN particle on the DXTRAN surface is deduced to be

$$\begin{aligned}\mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P},\boldsymbol{P}'\right)\mathrm{d}\boldsymbol{P}' = & \underbrace{\mathbb{1}\left(\boldsymbol{x}\notin\left\{\boldsymbol{x}_{\mathrm{DX}}\right\}\right)}_{①}\underbrace{\delta\left(\boldsymbol{x}'-\boldsymbol{x}_{\mathrm{DX}}\left(\boldsymbol{x},\boldsymbol{\Omega}'\right)\right)}_{②}\\ & \times\underbrace{\tilde{p}\left(\boldsymbol{\Omega},E\rightarrow\boldsymbol{\Omega}',E'\right)}_{③}\\ & \times\Bigg[\underbrace{\beta\left(\boldsymbol{x}\right)}_{④}\underbrace{\delta\left(w'-\frac{w}{\beta\left(\boldsymbol{x}\right)}\frac{p\left(\boldsymbol{\Omega},E\rightarrow\boldsymbol{\Omega}',E'\right)}{\tilde{p}\left(\boldsymbol{\Omega},E\rightarrow\boldsymbol{\Omega}',E'\right)}p_{\mathrm{ff}}\left(\boldsymbol{x},\boldsymbol{x}_{\mathrm{DX}}\right)\right)}_{⑤}\\ & \quad+\underbrace{\left(1-\beta\left(\boldsymbol{x}\right)\right)}_{⑥}\underbrace{\delta\left(w'\right)}_{⑦}\Bigg]\\ & \times\mathrm{d}V'\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}w'.\end{aligned}\tag{3.52}$$

Each of the terms in this biasing kernel are described next.

Term ① is an indicator function that specifies when the kernel applies. That is, the kernel applies when the initiating position $\boldsymbol{x}$ is not within the set of points that contain the boundary or interior of the DXTRAN region, $\{\boldsymbol{x}_{\mathrm{DX}}\}$. If the indicator is satisfied, the DXTRAN biasing kernel is applied, which represents performing DXTRAN processing. If not, the DXTRAN biasing kernel is not applied so DXTRAN variance-reduction processing is not performed.

Term ② gives the position that the DXTRAN particle is created ($\boldsymbol{p}_{\mathrm{DX}}$ in Figs. 1.1, 2.11, and 2.14). The position that the DXTRAN particle is created is a function of the initiating position $\boldsymbol{x}$ and the biased direction of the DXTRAN particle determined by sampling from the biased scattering emergence PDF, $\tilde{p}\left(\boldsymbol{\Omega},E\rightarrow\boldsymbol{\Omega}',E'\right)$. With that, Term ③ is the biased scattering emergence PDF with all outgoing directions $\boldsymbol{\Omega}'$ guaranteed to point toward the DXTRAN region using the process to construct $\tilde{p}\left(\boldsymbol{\Omega},E\rightarrow\boldsymbol{\Omega}',E'\right)$ described in Sections 2.4.6.2 and 2.4.6.3.

Term ④ is the probability of performing DXTRAN variance reduction that is defined by the user on a space-dependent basis. As described in Section 2.4.6.2, with probability $\beta(\boldsymbol{x})$ the DXTRAN particle is created and with probability $1-\beta(\boldsymbol{x})$ (Term ⑥) no DXTRAN particle is created (indicated as a DXTRAN particle created with zero weight in Term ⑦).

Assuming that the DXTRAN particle survives rouletting in Term ④, it is created using Term ⑤ with weight

$$w' = \frac{w}{\beta(\boldsymbol{x})} \frac{p(\boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E')}{\tilde{p}(\boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E')} p_{\text{ff}}(\boldsymbol{x}, \boldsymbol{x}_{\text{DX}}), \tag{3.53}$$

which is equivalent to Eq. (2.93).

Through deducing Eq. (3.52), the simplified DXTRAN particle creation mechanics described in Section 2.4.6.3 have been described analytically.

### 3.7.2 DXTRAN Free-flight Transmission Kernel

In concert with the DXTRAN biasing kernel described previously, the DXTRAN process modifies the analog free-flight transmission kernel because movement into a DXTRAN region (in a forward sense) is disallowed. As noted, the DXTRAN region consists of the set of points $\{\boldsymbol{x}_{\text{DX}}\} = \Gamma_{\text{DX}} + \delta\Gamma_{\text{DX}}$ that specify the interior and boundary points of the DXTRAN region, respectively. As a matter of notational convenience, define $\boldsymbol{x}_{\text{DX}} = \delta\Gamma_{\text{DX}}(\boldsymbol{x}, \boldsymbol{\Omega})$ as the point on the DXTRAN surface closest to $\boldsymbol{x}$ for free flight from $\boldsymbol{x}$ toward $\boldsymbol{x}_{\text{DX}}$ along $\boldsymbol{\Omega}$.

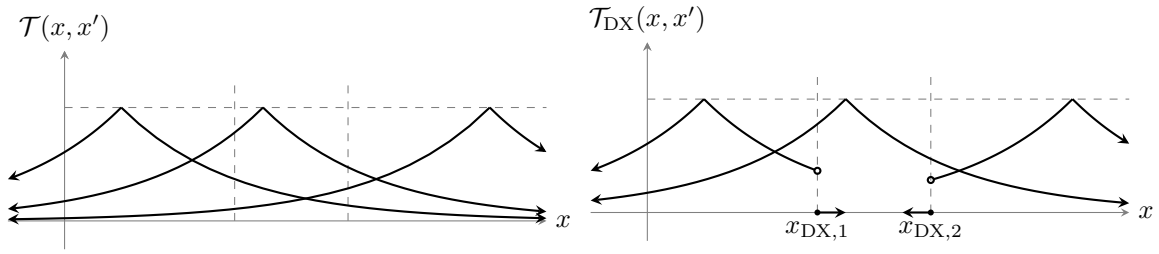
Determining whether movement from $\boldsymbol{x}$ to $\boldsymbol{x}'$ results in particle death on the surface of the DXTRAN region requires meeting two conditions:

1. The initial position, $\boldsymbol{x}$, must be external to the DXTRAN region ($\boldsymbol{x} \neq \boldsymbol{h} \forall \boldsymbol{h} \in \{\boldsymbol{x}_{\text{DX}}\}$), and
2. The final position must not be in, or beyond, the DXTRAN region

$$\left[ \frac{\boldsymbol{x} + \boldsymbol{x}'}{\|\boldsymbol{x} + \boldsymbol{x}'\|} \cdot \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} = 1 \cap \|\boldsymbol{x} + \boldsymbol{x}'\| \geq \|\boldsymbol{g}\| \right] \forall\, \boldsymbol{g} \in \{\boldsymbol{x}_{\text{DX}}\}.$$

If either of these conditions are not met, then the particle transmits normally. Enumerating these scenarios yields:

1. Initial position $\boldsymbol{x}$ outside DXTRAN region, final position $\boldsymbol{x}'$ not in or beyond the DXTRAN region: transmit normally,
2. Initial position $\boldsymbol{x}$ outside DXTRAN region, final position $\boldsymbol{x}'$ in or beyond the DXTRAN region: transmit the particle until the DXTRAN region boundary is encountered and then kill the particle, and
3. Initial position $\boldsymbol{x}$ inside DXTRAN region: transmit normally.




**(a)** Analog Transmission Kernel **(b)** DXTRAN Transmission Kernel

**Figure 3.3:** Analog and DXTRAN Transmission Kernels

As such, redefine the transmission kernel, $\mathcal{T}(\boldsymbol{P},\boldsymbol{P}')$, as a piecewise function to set emerging weight $w'$ to zero if the transmission occurs into (or through) a DXTRAN region:

$$\mathcal{T}_{\mathrm{DX}}(\boldsymbol{P},\boldsymbol{P}')\,\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}w' = \begin{cases} \exp\left(-\lambda(\boldsymbol{x},\boldsymbol{x}')\right)\delta(\boldsymbol{\Omega}'-\boldsymbol{\Omega})\,\delta(E'-E)\,\delta(w')\,\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}w' & \mathbb{1}(\text{truncate}) \\ \mathcal{T}(\boldsymbol{P},\boldsymbol{P}')\,\mathrm{d}\Omega'\mathrm{d}E'\mathrm{d}w', & \text{otherwise} \end{cases}, \tag{3.54a}$$

where

$$\mathbb{1}(\text{truncate}) = \left[\begin{array}{c} \boldsymbol{x}\notin\{\boldsymbol{x}_{\mathrm{DX}}\} \text{ and} \\ \frac{\boldsymbol{x}+\boldsymbol{x}'}{\|\boldsymbol{x}+\boldsymbol{x}'\|}\cdot\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}=1 \text{ and} \\ \|\boldsymbol{x}+\boldsymbol{x}'\|\geq\|\boldsymbol{g}\| \end{array}\right] \forall\,\boldsymbol{g}\in\{\boldsymbol{x}_{\mathrm{DX}}\}, \tag{3.54b}$$

indicating whether truncation is necessary. If $\{\boldsymbol{x}_{\mathrm{DX}}\}=\emptyset$ (i.e., there are no DXTRAN regions), then $\mathcal{T}_{\mathrm{DX}}(\boldsymbol{P},\boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}' = \mathcal{T}(\boldsymbol{P},\boldsymbol{P}')\,\mathrm{d}\boldsymbol{P}'$.

A demonstration of the behavior of $\mathcal{T}(\boldsymbol{P},\boldsymbol{P}')$ and $\mathcal{T}_{\mathrm{DX}}(\boldsymbol{P},\boldsymbol{P}')$ is given in Fig. 3.3 for three different hypothetical initial $\boldsymbol{P}$. In Fig. 3.3a, for free-flight transmission away from some hypothetical point, the probability of having flown to a particular position diminishes exponentially according to the optical distance traveled. However, in Fig. 3.3b the exponential decrease in probability of flight is the same unless the flight attempts to enter the DXTRAN region at which point the flight is terminated and continuing on is impossible unless the flight commences within the DXTRAN region.

### 3.7.3 DXTRAN History-score Probability Density Function

For the DXTRAN history-score-moment equations, the main modification is to the non-absorptive collision followed by scattering emergence HSPDF with the addition of the DXTRAN kernel to yield

$$\begin{aligned}\psi_{\mathrm{DX}}(\boldsymbol{P}_0,s)\,\mathrm{d}s = &\int\mathrm{d}\Omega_1\int\mathrm{d}E_1\int\mathrm{d}w_1\mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0,\boldsymbol{P}_1)\int\mathrm{d}\boldsymbol{P}_2\mathcal{K}(\boldsymbol{P}_1,\boldsymbol{P}_2)\int\mathrm{d}s_{\mathcal{K}}p_{\mathcal{K}}(\boldsymbol{P}_2,s_{\mathcal{K}}) \\ &\times\int\mathrm{d}\boldsymbol{P}_4\mathcal{B}_{\mathrm{DX}}(\boldsymbol{P}_2,\boldsymbol{P}_4)\int\mathrm{d}s_{\mathrm{DX}}\psi(\boldsymbol{P}_4,s_{\mathrm{DX}}) \\ &\times\int\mathrm{d}\boldsymbol{P}_3\mathcal{E}(\boldsymbol{P}_2,\boldsymbol{P}_3)\,\psi(\boldsymbol{P}_3,s-s_{\mathrm{DX}}-s_{\mathcal{K}})\,\mathrm{d}s.\end{aligned} \tag{3.55}$$

In this case there is an additional particle created following collision versus Eq. (3.17). In this formulation, reading from left to right, a particle undergoes free-flight (unless it attempts to enter a DXTRAN region) from phase space $\boldsymbol{P}_0$ until it undergoes a collision at $\boldsymbol{P}_1$. At this point, if a collision estimator is being used

then a contribution of $\mathrm{d}s_{\mathcal{K}}$ about $s_{\mathcal{K}}$ will be made via $p_{\mathcal{K}}$. Following collision and any tally contribution, two particles are tracked. The first is the DXTRAN particle generated via $\mathcal{B}_{\mathrm{DX}}$ in phase space $\boldsymbol{P}_4$ (now located on the DXTRAN region surface at $\boldsymbol{x}_{\mathrm{DX}}$) where tracking continues using the complete HSPDF that will go on to score $\mathrm{d}s_{\mathrm{DX}}$ about $s_{\mathrm{DX}}$. In addition, the initiating particle emerges normally from the collision into phase space $\boldsymbol{P}_3$ where it proceeds using the complete HSPDF and contributes $\mathrm{d}s$ about $s - s_{\mathrm{DX}} - s_{\mathcal{K}}$. Note that $\mathcal{T}$ became $\mathcal{T}_{\mathrm{DX}}$ and is also replaced in all other history-score probability density functions because the particle must be killed when it attempts to enter the DXTRAN region on its next free flight.

### 3.7.4 First Moment of DXTRAN History-score Probability Density Function

The DXTRAN HSMEs $\Psi_{1,\mathrm{DX}}(\boldsymbol{P}_0)$ and $\Psi_{2,\mathrm{DX}}(\boldsymbol{P}_0)$ are now derived. One can calculate the first DXTRAN history-score-moment equation as

$$\Psi_{1,\mathrm{DX}}(\boldsymbol{P}_0) = \int \mathrm{d}s\, s\, \psi_{\mathrm{DX}}(\boldsymbol{P}_0, s)\,, \tag{3.56}$$

which becomes

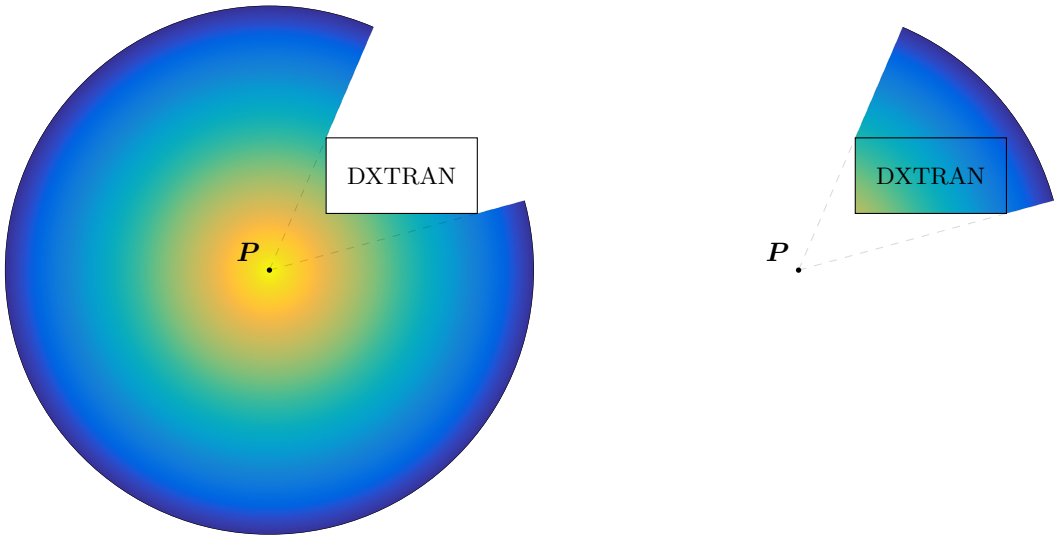
$$\begin{aligned}\Psi_{1,\mathrm{DX}}(\boldsymbol{P}_0) = &\int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}(\boldsymbol{P}_2, s_{\mathcal{K}}) \\ &\times \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4) \int \mathrm{d}s_{\mathrm{DX}} \psi(\boldsymbol{P}_4, s_{\mathrm{DX}}) \\ &\times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \int \mathrm{d}s\, s\, \psi(\boldsymbol{P}_3, s - s_{\mathrm{DX}} - s_{\mathcal{K}})\,.\end{aligned} \tag{3.57}$$

Make the substitution $q = s - s_{\mathrm{DX}} - s_{\mathcal{K}} \implies s = q + s_{\mathrm{DX}} + s_{\mathcal{K}}$ and recognize that $\mathrm{d}s = \mathrm{d}q$ to obtain

$$\begin{aligned}\Psi_{1,\mathrm{DX}}(\boldsymbol{P}_0) = &\int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}(\boldsymbol{P}_2, s_{\mathcal{K}}) \\ &\times \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4) \int \mathrm{d}s_{\mathrm{DX}} \psi(\boldsymbol{P}_4, s_{\mathrm{DX}}) \\ &\times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \int \mathrm{d}q\, (q + s_{\mathrm{DX}} + s_{\mathcal{K}})\, \psi(\boldsymbol{P}_3, q)\,,\end{aligned} \tag{3.58}$$

which can be expanded as

$$\begin{aligned}\Psi_{1,\mathrm{DX}}(\boldsymbol{P}_0) = &\int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\ &\times \Bigg[ \int \mathrm{d}s_{\mathcal{K}}\, s_{\mathcal{K}}\, p_{\mathcal{K}}(\boldsymbol{P}_2, s_{\mathcal{K}}) \\ &\quad + \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4) \int \mathrm{d}s_{\mathrm{DX}}\, s_{\mathrm{DX}}\, \psi(\boldsymbol{P}_4, s_{\mathrm{DX}}) \\ &\quad + \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \int \mathrm{d}q\, q\, \psi(\boldsymbol{P}_3, q) \Bigg]\,,\end{aligned} \tag{3.59}$$




**(a)** $\mathcal{E}(\boldsymbol{P}, \boldsymbol{P}')\,\mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}', \boldsymbol{P}'')$ **(b)** $\mathcal{B}_{DX}(\boldsymbol{P}, \boldsymbol{P}')\,\mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}', \boldsymbol{P}'')$

**Figure 3.4:** Effect of Emergence and DXTRAN Kernels Followed by DXTRAN Transmission Kernel

which becomes

$$\begin{aligned}\Psi_{1,\mathrm{DX}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\ \times \Bigg[ \int \mathrm{d}s\, s\, p_{\mathcal{K}}(\boldsymbol{P}_2, s) \\ + \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4)\, \Psi_1(\boldsymbol{P}_4) \\ + \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3)\, \Psi_1(\boldsymbol{P}_3) \Bigg]. \end{aligned} \tag{3.60}$$

Now the $\Psi_1(\boldsymbol{P}_3)$ and $\Psi_1(\boldsymbol{P}_4)$ HSMEs are operated on by

$$\int \mathrm{d}\Omega_3 \int \mathrm{d}E_3 \int \mathrm{d}w_3 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_3, \boldsymbol{P}_4) \text{ and } \int \mathrm{d}\Omega_4 \int \mathrm{d}E_4 \int \mathrm{d}w_4 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_4, \boldsymbol{P}_5), \tag{3.61}$$

respectively. This truncates the free flight following emergence as the random walk enters the DXTRAN region as shown in Figure 3.4a but allows the DXTRAN kernel to transport normally as shown in Figure 3.4b. The synonymous adjoint interpretation is that particles cannot exit DXTRAN regions. In Figure 3.4, it is assumed that scattering to any new angle is equally likely (leading to the azimuthally uniform radial gradient) and that transmission along the resulting direction is exponentially attenuated by radius (leading to attenuation by radius away from the collision point at $\boldsymbol{P}$). It is evident that taking the sum of these two

behaviors yields the analog behavior of emergence followed by standard transmission. As such, one obtains

$$\int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \int \mathrm{d}\Omega_6 \int \mathrm{d}E_6 \int \mathrm{d}w_6 \mathcal{T}\left(\boldsymbol{P}_3, \boldsymbol{P}_6\right)$$
$$= \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_1\left(\boldsymbol{P}_4\right) \int \mathrm{d}\Omega_6 \int \mathrm{d}E_6 \int \mathrm{d}w_6 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_4, \boldsymbol{P}_5\right)$$
$$+ \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \int \mathrm{d}\Omega_6 \int \mathrm{d}E_6 \int \mathrm{d}w_6 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_3, \boldsymbol{P}_6\right), \quad (3.62)$$

which then provides the analog adjoint transport equation for $\Psi_1\left(\boldsymbol{P}_0\right)$ when combined with $\Psi_{1,\mathcal{A}}\left(\boldsymbol{P}_0\right)$ and $\Psi_{1,\mathcal{S}}\left(\boldsymbol{P}_0\right)$, as expected. Note that in Eq. (3.62)

$$\int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_1\left(\boldsymbol{P}_4\right) \int \mathrm{d}\Omega_6 \int \mathrm{d}E_6 \int \mathrm{d}w_6 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_4, \boldsymbol{P}_5\right)$$
$$= \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_1\left(\boldsymbol{P}_4\right) \int \mathrm{d}\Omega_6 \int \mathrm{d}E_6 \int \mathrm{d}w_6 \mathcal{T}\left(\boldsymbol{P}_4, \boldsymbol{P}_5\right) \quad (3.63)$$

because the DXTRAN kernel transmits onto the edge of the DXTRAN region. Because the edge of the DXTRAN region is included in $\{\boldsymbol{x}_{\mathrm{DX}}\}$ the DXTRAN biasing kernel, $\mathcal{B}_{\mathrm{DX}}$, is not applied because the indicator function is not satisfied and free-flight transmission is treated as analog by $\mathcal{T}_{\mathrm{DX}}$. As such, transmission following creation of the DXTRAN particle is performed in analog until the DXTRAN particle leaves the DXTRAN region and again satisfies the indicator in $\mathcal{B}_{\mathrm{DX}}$.

### 3.7.5 Second Moment of DXTRAN History-score Probability Density Function

The second DXTRAN history-score-moment equation is

$$\Psi_{2,\mathrm{DX}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}s\, s^2\, \psi_{\mathrm{DX}}\left(\boldsymbol{P}_0, s\right), \quad (3.64)$$

which becomes

$$\Psi_{2,\mathrm{DX}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right)$$
$$\times \left[ \int \mathrm{d}s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \right.$$
$$\times \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \int \mathrm{d}s_{\mathrm{DX}} \psi\left(\boldsymbol{P}_4, s_{\mathrm{DX}}\right)$$
$$\left. \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int \mathrm{d}s\, s^2\, \psi\left(\boldsymbol{P}_3, s - s_{\mathrm{DX}} - s_{\mathcal{K}}\right) \right]. \quad (3.65)$$

Again making the substitution $q = s - s_{\mathrm{DX}} - s_{\mathcal{K}} \implies s = q + s_{\mathrm{DX}} + s_{\mathcal{K}}$ and $\mathrm{d}q = \mathrm{d}s$ followed by distributing the terms yields

$$\begin{aligned}
\Psi_{2,\mathrm{DX}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\
& \times \left[\int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_2\left(\boldsymbol{P}_3\right)\right. \\
& + \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_2\left(\boldsymbol{P}_4\right) \\
& + \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}}^2 p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \\
& + 2 \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_1\left(\boldsymbol{P}_4\right) \\
& + 2 \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \\
& \left. + 2 \int \mathrm{d}s_{\mathcal{K}} s_{\mathcal{K}} p_{\mathcal{K}}\left(\boldsymbol{P}_2, s_{\mathcal{K}}\right) \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_1\left(\boldsymbol{P}_4\right)\right] . \qquad (3.66)
\end{aligned}$$

Inside the DXTRAN region, this reduces to

$$\begin{aligned}
\Psi_{2,\mathrm{DX}}\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\
& \times \left[\int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_2\left(\boldsymbol{P}_3\right)\right. \\
& + \int \mathrm{d}s\, s^2 p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right) \\
& \left. + 2 \int \mathrm{d}s\, s\, p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right)\right] \qquad (3.67)
\end{aligned}$$

because of the presence of the indicator function in the DXTRAN kernel, which yields the same as in the case of not having DXTRAN present. External to the DXTRAN region, expand the $\mathcal{B}_{\mathrm{DX}}$-related terms as

$$\begin{aligned}
\int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_2\left(\boldsymbol{P}_4\right) = & \int \mathrm{d}V_4 \int \mathrm{d}\Omega_4 \int \mathrm{d}E_4 \int \mathrm{d}w_4 \Psi_2\left(\boldsymbol{x}_4, \boldsymbol{\Omega}_4, E_4, w_4\right) \\
& \times \delta\left(\boldsymbol{x}_4 - \boldsymbol{x}_{\mathrm{DX}}\right) \widetilde{p}\left(\boldsymbol{\Omega}_2, E_2 \rightarrow \boldsymbol{\Omega}_4, E_4\right) \\
& \times \left[\beta\left(\boldsymbol{x}_2\right) \delta\left(w_4 - \frac{w_2}{\beta\left(\boldsymbol{x}_2\right)} \frac{p\left(\boldsymbol{\Omega}_2, E_2 \rightarrow \boldsymbol{\Omega}_4, E_4\right)}{\tilde{p}\left(\boldsymbol{\Omega}_2, E_2 \rightarrow \boldsymbol{\Omega}_4, E_4\right)} p_{\mathrm{ff}}\left(\boldsymbol{x}_2, \boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4\right)\right) + \left(1 - \beta\left(\boldsymbol{x}\right)\right) \delta\left(w_4\right)\right] \qquad (3.68) \\
= & \int \mathrm{d}\Omega_4 \int \mathrm{d}E_4 \frac{p_{\mathrm{ff}}^2\left(\boldsymbol{x}_2, \boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4\right)}{\beta\left(\boldsymbol{x}_2\right)} \frac{p^2\left(\boldsymbol{\Omega}_2, E_2 \rightarrow \boldsymbol{\Omega}_4, E_4\right)}{\tilde{p}\left(\boldsymbol{\Omega}_2, E_2 \rightarrow \boldsymbol{\Omega}_4, E_4\right)} \Psi_2\left(\boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4, E_4, w_2\right) , \qquad (3.69)
\end{aligned}$$

$$
\begin{aligned}
&2\int \mathrm{d}\boldsymbol{P}_3\mathcal{E}\left(\boldsymbol{P}_2,\boldsymbol{P}_3\right)\Psi_1\left(\boldsymbol{P}_3\right)\int \mathrm{d}\boldsymbol{P}_4\mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2,\boldsymbol{P}_4\right)\Psi_1\left(\boldsymbol{P}_4\right)\\
&\quad=2\int \mathrm{d}V_3\int \mathrm{d}\Omega_3\int \mathrm{d}E_3\int \mathrm{d}w_3\\
&\qquad\times\delta\left(\boldsymbol{x}_3-\boldsymbol{x}_2\right)p\left(\boldsymbol{\Omega}_2,E_2\rightarrow\boldsymbol{\Omega}_3,E_3\right)\delta\left(w_3-w_2\right)\Psi_1\left(\boldsymbol{x}_3,\boldsymbol{\Omega}_3,E_3,w_3\right)\\
&\qquad\times\int \mathrm{d}V_4\int \mathrm{d}\Omega_4\int \mathrm{d}E_4\int \mathrm{d}w_4\Psi_1\left(\boldsymbol{x}_4,\boldsymbol{\Omega}_4,E_4,w_4\right)\delta\left(\boldsymbol{x}_4-\boldsymbol{x}_{\mathrm{DX}}\right)\widetilde{p}\left(\boldsymbol{\Omega}_2,E_2\rightarrow\boldsymbol{\Omega}_4,E_4\right)\\
&\qquad\times\left[\beta\left(\boldsymbol{x}_2\right)\delta\left(w_4-\frac{w_2}{\beta\left(\boldsymbol{x}_2\right)}\frac{p\left(\boldsymbol{\Omega}_2,E_2\rightarrow\boldsymbol{\Omega}_4,E_4\right)}{\tilde{p}\left(\boldsymbol{\Omega}_2,E_2\rightarrow\boldsymbol{\Omega}_4,E_4\right)}p_{\mathrm{ff}}\left(\boldsymbol{x}_2,\boldsymbol{x}_{\mathrm{DX}},\boldsymbol{\Omega}_4\right)\right)+\left(1-\beta\left(\boldsymbol{x}\right)\right)\delta\left(w'\right)\right] \qquad (3.70)\\
&\quad=2\int \mathrm{d}\Omega_3\int \mathrm{d}E_3 p\left(\boldsymbol{\Omega}_2,E_2\rightarrow\boldsymbol{\Omega}_3,E_3\right)\Psi_1\left(\boldsymbol{x}_2,\boldsymbol{\Omega}_3,E_3,w_2\right)\\
&\qquad\times\int \mathrm{d}\Omega_4\int \mathrm{d}E_4 p\left(\boldsymbol{\Omega}_2,E_2\rightarrow\boldsymbol{\Omega}_4,E_4\right)p_{\mathrm{ff}}\left(\boldsymbol{x}_2,\boldsymbol{x}_{\mathrm{DX}},\boldsymbol{\Omega}_4\right)\Psi_1\left(\boldsymbol{x}_{\mathrm{DX}},\boldsymbol{\Omega}_4,E_4,w_2\right), \qquad (3.71)
\end{aligned}
$$

and

$$
\begin{aligned}
&2\int \mathrm{d}s_{\mathcal{K}}s_{\mathcal{K}}p_{\mathcal{K}}\left(\boldsymbol{P}_2,s_{\mathcal{K}}\right)\int \mathrm{d}\boldsymbol{P}_4\mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2,\boldsymbol{P}_4\right)\Psi_1\left(\boldsymbol{P}_4\right)\\
&\quad=2\int \mathrm{d}s_{\mathcal{K}}s_{\mathcal{K}}p_{\mathcal{K}}\left(\boldsymbol{P}_2,s_{\mathcal{K}}\right)\\
&\qquad\times\int \mathrm{d}\Omega_4\int \mathrm{d}E_4 p\left(\boldsymbol{\Omega}_2,E_2\rightarrow\boldsymbol{\Omega}_4,E_4\right)p_{\mathrm{ff}}\left(\boldsymbol{x}_2,\boldsymbol{x}_{\mathrm{DX}},\boldsymbol{\Omega}_4\right)\Psi_1\left(\boldsymbol{x}_{\mathrm{DX}},\boldsymbol{\Omega}_4,E_4,w_2\right). \qquad (3.72)
\end{aligned}
$$

This expansion leads to the moment and associated source equations external the DXTRAN region when combined with $\Psi_{2,\mathcal{S}}\left(\boldsymbol{P}_0\right)$ and $\Psi_{2,\mathcal{A}}\left(\boldsymbol{P}_0\right)$:

$$
\begin{aligned}
\Psi_2\left(\boldsymbol{P}_0\right)=&\int \mathrm{d}\Omega_1\int \mathrm{d}E_1\int \mathrm{d}w_1\mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0,\boldsymbol{P}_1\right)\int \mathrm{d}\boldsymbol{P}_2\mathcal{S}\left(\boldsymbol{P}_1,\boldsymbol{P}_2\right)\Psi_2\left(\boldsymbol{P}_2\right)\\
&+\int \mathrm{d}\Omega_1\int \mathrm{d}E_1\int \mathrm{d}w_1\mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0,\boldsymbol{P}_1\right)\int \mathrm{d}\boldsymbol{P}_2\mathcal{K}\left(\boldsymbol{P}_1,\boldsymbol{P}_2\right)\\
&\quad\times\int \mathrm{d}\boldsymbol{P}_3\mathcal{E}\left(\boldsymbol{P}_2,\boldsymbol{P}_3\right)\Psi_2\left(\boldsymbol{P}_3\right)\\
&+\int \mathrm{d}\Omega_1\int \mathrm{d}E_1\int \mathrm{d}w_1\mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0,\boldsymbol{P}_1\right)\int \mathrm{d}\boldsymbol{P}_2\mathcal{K}\left(\boldsymbol{P}_1,\boldsymbol{P}_2\right)\\
&\quad\times\int \mathrm{d}\boldsymbol{P}_4\mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2,\boldsymbol{P}_4\right)\Psi_2\left(\boldsymbol{P}_4\right)\\
&+Q_2\left(\boldsymbol{P}_0\right), \qquad (3.73a)
\end{aligned}
$$

$$
\begin{aligned}
Q_2\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \mathcal{A}\left(\boldsymbol{P}_2\right) \int \mathrm{d}s\, s^2\, p_{\mathcal{A}}\left(\boldsymbol{P}_2, s\right) \\
& + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s^2\, p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \Psi_1\left(\boldsymbol{P}_2\right) \int \mathrm{d}s\, s\, p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right) \\
& + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s^2 p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\
& \quad \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_1\left(\boldsymbol{P}_4\right) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\
& \quad \times \int \mathrm{d}s\, s\, p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\
& \quad \times \int \mathrm{d}s\, s\, p_{\mathcal{K}}\left(\boldsymbol{P}_2, s\right) \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Psi_1\left(\boldsymbol{P}_4\right), \qquad (3.73\mathrm{b})
\end{aligned}
$$

which can be expanded as

$$
\begin{aligned}
\Psi_2\left(\boldsymbol{P}_0\right) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \Psi_2\left(\boldsymbol{P}_2\right) \\
& + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\
& \quad \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_2\left(\boldsymbol{P}_3\right) \\
& + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\
& \quad \times \int \mathrm{d}\Omega_4 \int \mathrm{d}E_4 \frac{p_{\mathrm{ff}}^2\left(\boldsymbol{x}_2, \boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4\right)}{\beta\left(\boldsymbol{x}_2\right)} \frac{p^2\left(\boldsymbol{\Omega}_2, E_2 \rightarrow \boldsymbol{\Omega}_4, E_4\right)}{\tilde{p}\left(\boldsymbol{\Omega}_2, E_2 \rightarrow \boldsymbol{\Omega}_4, E_4\right)} \Psi_2\left(\boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4, E_4, w_2\right) \\
& + Q_2\left(\boldsymbol{P}_0\right), \qquad (3.74\mathrm{a})
\end{aligned}
$$

$$
\begin{aligned}
Q_2(\boldsymbol{P}_0) = & \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \mathcal{A}(\boldsymbol{P}_2) \int \mathrm{d}s\, s^2\, p_{\mathcal{A}}(\boldsymbol{P}_2, s) \\
& + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s\, s^2\, p_{\mathcal{S}}(\boldsymbol{P}_2, s) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \Psi_1(\boldsymbol{P}_2) \int \mathrm{d}s\, s\, p_{\mathcal{S}}(\boldsymbol{P}_2, s) \\
& + \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s\, s^2 p_{\mathcal{K}}(\boldsymbol{P}_2, s) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\
& \quad \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \Psi_1(\boldsymbol{P}_3) \\
& \quad \times \int \mathrm{d}\Omega_4 \int \mathrm{d}E_4 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_4) p_{\mathrm{ff}}(\boldsymbol{x}_2, \boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4) \Psi_1(\boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4, E_4, w_2) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\
& \quad \times \int \mathrm{d}s\, s\, p_{\mathcal{K}}(\boldsymbol{P}_2, s) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \Psi_1(\boldsymbol{P}_3) \\
& + 2 \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\
& \quad \times \int \mathrm{d}s\, s\, p_{\mathcal{K}}(\boldsymbol{P}_2, s) \\
& \quad \times \int \mathrm{d}\Omega_4 \int \mathrm{d}E_4 p(\boldsymbol{\Omega}_2, E_2 \to \boldsymbol{\Omega}_4, E_4) p_{\mathrm{ff}}(\boldsymbol{x}_2, \boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4) \Psi_1(\boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4, E_4, w_2). \qquad (3.74\mathrm{b})
\end{aligned}
$$

### 3.8 Summary

This chapter has shown how to formulate HSPDFs and then how to calculate the associated HSMEs for analog random walks that terminate in absorption, continue following Monte Carlo surface crossing, and continue following non-absorptive scattering with emergence. Using these HSMEs as a basis for comparison, the DXTRAN biased transport kernel and accompanying free-flight transmission kernel with DXTRAN truncation are deduced. These kernels form the DXTRAN HSPDF, which the first and second HSMEs are calculated for.

As a result of the HSMEs, the Monte Carlo moments can be calculated as

$$
M_m = \int \mathrm{d}\boldsymbol{P}_0 \Psi_m(\boldsymbol{P}_0) Q(\boldsymbol{P}_0). \qquad (3.75)
$$

More specifically, having computed $\Psi_1(\boldsymbol{P}_0)$ the predicted Monte Carlo tally mean is

$$
\mu = M_1 = \int \mathrm{d}\boldsymbol{P}_0 \Psi_1(\boldsymbol{P}_0) Q(\boldsymbol{P}_0), \qquad (3.76)
$$

where $Q(\boldsymbol{P}_0)$ is the forward source. Similarly, using $\Psi_2(\boldsymbol{P}_0)$ the predicted Monte Carlo tally population variance is

$$
\sigma^2 = M_2 - M_1^2 = \int \mathrm{d}\boldsymbol{P}_0 \Psi_2(\boldsymbol{P}_0) Q(\boldsymbol{P}_0) - \widehat{\mu}^2. \qquad (3.77)
$$

These quantities are expected to be the same as a statistically well-converged Monte Carlo-calculated tally sample mean (from Eq. (2.72)) and sample variance (from Eq. (2.80)). When $M_m$, $\mu$, and $\sigma^2$ are calculated analytically as they have been to this point, they are expressed with no additional notation.

When calculated using Monte Carlo, they are expressed as $M_{m,\mathrm{MC}}$, $\mu_{\mathrm{MC}}$, and $\sigma^2_{\mathrm{MC}}$. When calculated using a deterministic method (such as those described in Chapter 5 using the discretization described in Chapter 2), they are expressed as $\widehat{M}_m$, $\widehat{\mu}$, and $\widehat{\sigma}^2$.

# Chapter 4

## Future-time Equation & Computational Cost Optimization

To analytically characterize the computational cost [Eq. (2.82)] of a Monte Carlo calculation, the analytic values for $M_1$, $M_2$, and $T$ are needed. The HSMEs in Chapter 2 provide $M_1$ and $M_2$ [Eq. (3.75)]. This chapter describes the Future-time Equation (FTE), which is used to calculate $T$. The chapter then continues by describing how Monte Carlo timing kernels, used within the FTE, are obtained (with more details in Appendix E) and then checked for appropriateness. Finally, an overview of the optimization algorithms explored as part of this work to determine the algorithm best-suited to compute a minimum computational cost is given as well as notes on some algorithms considered but ultimately not used.

### 4.1 Future Time Equation

A break with historical nomenclature is made when discussing the FTE in this work. Historically, $\tau$ has been used in various, sometimes confounding, contexts such as:

1. to indicate the times associated with certain Monte Carlo calculation operations such as tallying, processing collisions, etc.,
2. as a function of only phase space (i.e., $\tau\left(\boldsymbol{P}_0\right)$) to represent the expected future time (the first-moment FTE), and also sometimes
3. as the future-time phase space variable in the future time probability density function (FTPDF), $\tau\left(\boldsymbol{P}_0, \tau\right) \mathrm{d}\tau$.

Because of the confusion that can arise from these various (re-)uses of $\tau$, the following nomenclature is adopted:

1. $\tau_x$ represents the time associated with a particular Monte Carlo computational event $x$,
2. $\upsilon\left(\boldsymbol{P}_0, \tau\right) \mathrm{d}\tau$ represents the FTPDF giving the probability of contributing future time $\mathrm{d}\tau$ about $\tau$ to the tally from phase-space position $\boldsymbol{P}_0$,
3. $\Upsilon\left(\boldsymbol{P}_0\right)$ represents the expected future time as a function of phase space where

$$\Upsilon\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\tau\, \tau\, \upsilon\left(\boldsymbol{P}_0, \tau\right), \tag{4.1}$$

**Table 4.1:** MCNP6 Calculation Times from Valgrind Profiling

| FTE Term | COVRT Input Variable | Value [min] | Note |
|---|---|---|---|
| $\tau_{\text{xs}}$ | `acetot` | $8.45004\times10^{-8}$ | — |
| $\tau_{\text{bank}}$ | `bankit` | $2.24612\times10^{-7}$ | — |
| $\tau_{\text{col}}$ | `colidn` | $1.52900\times10^{-7}$ | Reused from [58]. |
| $\tau_{\text{DX}}$ | `dxtran` | $5.88000\times10^{-7}$ | — |
| $\tau_{\text{src}}$ | `startp` | $1.34064\times10^{-6}$ | — |
| $\tau_{\text{surf}}$ | `surfac` | $4.39085\times10^{-7}$ | — |
| $\tau_{\text{tally}}$ | `tally` | $4.31506\times10^{-7}$ | Reused from [58]. |
| $\tau_{\text{geom}}$ | `track` | $1.05709\times10^{-7}$ | — |
| $\tau_{\text{ff}}$ | `transm` | $9.55000\times10^{-8}$ | — |
| $\tau_{\text{ww}}$ | `wtwndo` | $6.12454\times10^{-8}$ | Reused from [58]. |

which is consistent with how $\psi(\boldsymbol{P}_0, s)\,ds$ and $\Psi_m(\boldsymbol{P}_0)$ are related. That is, Eq. (4.1) gives the expected time contributed to the overall calculation time from phase-space position $\boldsymbol{P}_0$.

4. Finally, $T$ represents the future time contributed to the overall computation time where

$$T = \int \mathrm{d}\boldsymbol{P}_0 \left[\Upsilon(\boldsymbol{P}_0) + \tau_{\text{src}}\right] Q(\boldsymbol{P}_0) \tag{4.2}$$

and where $\tau_{\text{src}}$ is the time required to process a forward source event.

To predict $T$, one can begin by defining the times for each Monte Carlo calculation event as

$\tau_{\text{tally}}$ the time required to process a tally event,

$\tau_{\text{col}}$ the time required to process a collision event,

$\tau_{\text{xs}}$ the time required to process a cross-section lookup,

$\tau_{\text{geom}}$ the time required to perform transmission along a free-flight trajectory,

$\tau_{\text{ff}}$ the time required to perform raytracing, per Monte Carlo surface, for DXTRAN,

$\tau_{\text{DX}}$ the time required to perform all other DXTRAN operations,

$\tau_{\text{surf}}$ the time required to process surface-crossing,

$\tau_{\text{bank}}$ the time required to process a particle bank event (either providing or retrieving a particle from the bank), and

$\tau_{\text{src}}$ the time required to process a forward source event as noted previously.

These timing kernels are obtained by profiling representative Monte Carlo calculations on a Los Alamos National Laboratory dedicated-use computing node through an allocation on a supercomputer. Each node has two Intel Xeon CPUs (model E5-2695 v4) operating at 2.10 GHz. The specific times obtained and used for this work are described in Table 4.1 and more details on the process to obtain these times are available in Appendix E. These times are taken as constant by type of event. For example, it is assumed that all

tally events take $\tau_{\text{tally}}$ rather than characterizing particular tally types with different times, e.g., surface, track-length, etc. While this level of refinement may be valuable to future work, this work and prior work [58] observed that the FTE is not insurmountably sensitive to this level of granularity versus competing factors (e.g., the effect of computer system load on calculation time).

In addition, one must define time contribution probability density functions for each computational event in the form $p_e(\boldsymbol{P},\tau)\,\mathrm{d}\tau$ where

$$p_e(\boldsymbol{P},\tau) = \delta(\tau - \tau_x). \tag{4.3}$$

This definition indicates that for computational event $e$ at phase-space position $\boldsymbol{P}$, additional computational time $\tau_e$ is contributed to the calculation (where the various $\tau_e$ values are listed in Table 4.1). Thus, each $p_e(\boldsymbol{P},\tau)\,\mathrm{d}\tau$ gives the probability of contributing time $\mathrm{d}\tau$ about $\tau$ from phase-space position $\boldsymbol{P}$ for given computational event $e$ for each of the aforementioned computational events. These time contribution PDFs act as source terms to the FTE by indicating what type of events contribute time, where, and by how much. That is, they are comparable to the scoring density functions used in the HSMEs described in Section 3.2. They are defined in a similar manner and exist within phase space as a function of the associated event. For example, $p_{\text{surf}}(\boldsymbol{P},\tau_{\text{surf}})\,\mathrm{d}\tau_{\text{surf}}$ only exists at Monte Carlo surfaces and $p_{\text{col}}(\boldsymbol{P},\tau_{\text{col}})\,\mathrm{d}\tau_{\text{col}}$ cannot exist in voids.

### 4.2 Analog Future-time Probability Density Functions

Next, one can form the partial FTPDFs comparable to the HSPDFs for analog absorption, surface crossing, and non-absorptive collision as

$$\begin{aligned}\upsilon_{\mathcal{A}}(\boldsymbol{P}_0,\tau)\,\mathrm{d}\tau = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0,\boldsymbol{P}_1) \int \mathrm{d}\tau_{\text{geom}}\, p_{\text{geom}}(\boldsymbol{P}_1,\tau_{\text{geom}}) \\ \times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1,\boldsymbol{P}_2) \int \mathrm{d}\tau_{\text{col}}\, p_{\text{col}}(\boldsymbol{P}_2,\tau_{\text{col}}) \int \mathrm{d}\tau_{\text{tally}}\, p_{\text{tally}}(\boldsymbol{P}_2,\tau_{\text{tally}}) \\ \times \mathcal{A}(\boldsymbol{P}_2)\,\delta(\tau - \tau_{\text{geom}} - \tau_{\text{col}} - \tau_{\text{tally}})\,\mathrm{d}\tau,\end{aligned} \tag{4.4a}$$

$$\begin{aligned}\upsilon_{\mathcal{S}}(\boldsymbol{P}_0,\tau)\,\mathrm{d}\tau = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0,\boldsymbol{P}_1) \int \mathrm{d}\tau_{\text{geom}}\, p_{\text{geom}}(\boldsymbol{P}_1,\tau_{\text{geom}}) \\ \times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1,\boldsymbol{P}_2) \int \mathrm{d}\tau_{\text{surf}}\, p_{\text{surf}}(\boldsymbol{P}_2,\tau_{\text{surf}}) \int \mathrm{d}\tau_{\text{tally}}\, p_{\text{tally}}(\boldsymbol{P}_2,\tau_{\text{tally}}) \int \mathrm{d}\tau_{\text{xs}}\, p_{\text{xs}}(\boldsymbol{P}_2,\tau_{\text{xs}}) \\ \times \upsilon(\boldsymbol{P}_2,\tau - \tau_{\text{geom}} - \tau_{\text{surf}} - \tau_{\text{tally}} - \tau_{\text{xs}})\,\mathrm{d}\tau,\end{aligned} \tag{4.4b}$$

and

$$\begin{aligned}\upsilon_{\mathcal{E}}(\boldsymbol{P}_0,\tau)\,\mathrm{d}\tau = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0,\boldsymbol{P}_1) \int \mathrm{d}\tau_{\text{geom}}\, p_{\text{geom}}(\boldsymbol{P}_1,\tau_{\text{geom}}) \\ \times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1,\boldsymbol{P}_2) \int \mathrm{d}\tau_{\text{col}}\, p_{\text{col}}(\boldsymbol{P}_2,\tau_{\text{col}}) \int \mathrm{d}\tau_{\text{tally}}\, p_{\text{tally}}(\boldsymbol{P}_2,\tau_{\text{tally}}) \\ \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2,\boldsymbol{P}_3) \int \mathrm{d}\tau_{\text{xs}}\, p_{\text{xs}}(\boldsymbol{P}_3,\tau_{\text{xs}}) \\ \times \upsilon(\boldsymbol{P}_3,\tau - \tau_{\text{geom}} - \tau_{\text{surf}} - \tau_{\text{tally}} - \tau_{\text{xs}})\,\mathrm{d}\tau.\end{aligned} \tag{4.4c}$$

Equation (4.4) represents the three analog random walks capable of contributing time to the calculation. At each stage of the random walk, one or more computational events are performed to advance the state of the random walk with associated time taken. For example, in Eq. (4.4a), during each free-flight transmission time $\tau_{\text{geom}}$ is required and thus contributed to the overall calculation time. At the subsequent collision time is taken to process the collision ($\tau_{\text{col}}$) and time is contributed to perform an absorption tally (if applicable). Finally, the history can contribute any additional time calculation time that already has not been accounted for by $\tau_{\text{geom}}$, $\tau_{\text{col}}$, and $\tau_{\text{tally}}$.

Taking the three previous FTPDFs in Eq. (4.4) as the only transport progressions and time contribution paths possible, one can assemble the complete analog FTPDF as

$$\upsilon\left(\boldsymbol{P}_0,\tau\right)\mathrm{d}\tau=\upsilon_{\mathcal{A}}\left(\boldsymbol{P}_0,\tau\right)\mathrm{d}\tau+\upsilon_{\mathcal{S}}\left(\boldsymbol{P}_0,\tau\right)\mathrm{d}\tau+\upsilon_{\mathcal{E}}\left(\boldsymbol{P}_0,\tau\right)\mathrm{d}\tau. \tag{4.5}$$

The first statistical moment of the FTPDF, giving the expected time contributed to the tally from phase space position $\mathrm{d}\boldsymbol{P}_0$ about $\boldsymbol{P}_0$ can be calculated with Eq. (4.1) so the integral detector time can be calculated with Eq. (4.2).

The individual analog FTEs are calculated in the next three subsections. Then, the effect of DXTRAN is introduced to Eq. (4.4c) and the modified FTE recalculated.

### 4.3 Analog Future-time Equation

One may compute the analog absorption FTE using Eq. (4.4a) as

$$\Upsilon_{\mathcal{A}}\left(\boldsymbol{P}_0\right)=\int\mathrm{d}\tau\,\tau\,\upsilon_{\mathcal{A}}\left(\boldsymbol{P}_0,\tau\right) \tag{4.6}$$

to obtain

$$\begin{aligned}\Upsilon_{\mathcal{A}}\left(\boldsymbol{P}_0\right)=\int\mathrm{d}\Omega_1\int\mathrm{d}E_1\int\mathrm{d}w_1\mathcal{T}\left(\boldsymbol{P}_0,\boldsymbol{P}_1\right)\int\mathrm{d}\tau_{\text{geom}}\,p_{\text{geom}}\left(\boldsymbol{P}_1,\tau_{\text{geom}}\right)\\ \times\int\mathrm{d}\boldsymbol{P}_2\mathcal{K}\left(\boldsymbol{P}_1,\boldsymbol{P}_2\right)\int\mathrm{d}\tau_{\text{col}}\,p_{\text{col}}\left(\boldsymbol{P}_2,\tau_{\text{col}}\right)\int\mathrm{d}\tau_{\text{tally}}\,p_{\text{tally}}\left(\boldsymbol{P}_2,\tau_{\text{tally}}\right)\\ \times\mathcal{A}\left(\boldsymbol{P}_2\right)\int\mathrm{d}\tau\,\tau\,\delta\left(\tau-\tau_{\text{geom}}-\tau_{\text{col}}-\tau_{\text{tally}}\right),\end{aligned} \tag{4.7}$$

and by recognizing that

$$\delta\left(\tau-\tau_{\text{geom}}-\tau_{\text{col}}-\tau_{\text{tally}}\right)=\delta\left(\tau-\left(\tau_{\text{geom}}+\tau_{\text{col}}+\tau_{\text{tally}}\right)\right) \tag{4.8}$$

one obtains

$$\begin{aligned}\Upsilon_{\mathcal{A}}\left(\boldsymbol{P}_0\right)=\int\mathrm{d}\Omega_1\int\mathrm{d}E_1\int\mathrm{d}w_1\mathcal{T}\left(\boldsymbol{P}_0,\boldsymbol{P}_1\right)\int\mathrm{d}\tau_{\text{geom}}\,p_{\text{geom}}\left(\boldsymbol{P}_1,\tau_{\text{geom}}\right)\\ \times\int\mathrm{d}\boldsymbol{P}_2\mathcal{K}\left(\boldsymbol{P}_1,\boldsymbol{P}_2\right)\int\mathrm{d}\tau_{\text{col}}\,p_{\text{col}}\left(\boldsymbol{P}_2,\tau_{\text{col}}\right)\int\mathrm{d}\tau_{\text{tally}}\,p_{\text{tally}}\left(\boldsymbol{P}_2,\tau_{\text{tally}}\right)\\ \times\mathcal{A}\left(\boldsymbol{P}_2\right)\left(\tau_{\text{geom}}+\tau_{\text{col}}+\tau_{\text{tally}}\right).\end{aligned} \tag{4.9}$$

Next, one can distribute the times and perform all integrations to obtain

$$\Upsilon_{\mathcal{A}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \left\{ \int \mathrm{d}\tau_{\mathrm{geom}}\, \tau_{\mathrm{geom}}\, p_{\mathrm{geom}}(\boldsymbol{P}_1, \tau_{\mathrm{geom}}) \right.$$
$$\left. + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \left[ \int \mathrm{d}\tau_{\mathrm{col}}\, \tau_{\mathrm{col}}\, p_{\mathrm{col}}(\boldsymbol{P}_2, \tau_{\mathrm{col}}) + \int \mathrm{d}\tau_{\mathrm{tally}}\, \tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}_2, \tau_{\mathrm{tally}}) \right] \mathcal{A}(\boldsymbol{P}_2) \right\}. \quad (4.10)$$

Thus, in a random walk that ends in absorption time will be contributed by tracking through the geometry to the point of absorption, by processing the collision and, if applicable, by tallying and finally terminating with absorption.

Next, compute the analog surface-crossing FTE using Eq. (4.4b) as

$$\Upsilon_{\mathcal{S}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\tau_{\mathrm{geom}}\, p_{\mathrm{geom}}(\boldsymbol{P}_1, \tau_{\mathrm{geom}})$$
$$\times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}\tau_{\mathrm{surf}}\, p_{\mathrm{surf}}(\boldsymbol{P}_2, \tau_{\mathrm{surf}}) \int \mathrm{d}\tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}_2, \tau_{\mathrm{tally}}) \int \mathrm{d}\tau_{\mathrm{xs}}\, p_{\mathrm{xs}}(\boldsymbol{P}_2, \tau_{\mathrm{xs}})$$
$$\times \int \mathrm{d}\tau\, \tau\, \upsilon(\boldsymbol{P}_2, \tau - (\tau_{\mathrm{geom}} + \tau_{\mathrm{surf}} + \tau_{\mathrm{tally}} + \tau_{\mathrm{xs}})), \quad (4.11)$$

where the substitution $q = \tau - (\tau_{\mathrm{geom}} + \tau_{\mathrm{surf}} + \tau_{\mathrm{tally}} + \tau_{\mathrm{xs}})$ and $\mathrm{d}q = \mathrm{d}\tau$ can be made to yield

$$\Upsilon_{\mathcal{S}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\tau_{\mathrm{geom}}\, p_{\mathrm{geom}}(\boldsymbol{P}_1, \tau_{\mathrm{geom}})$$
$$\times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}\tau_{\mathrm{surf}}\, p_{\mathrm{surf}}(\boldsymbol{P}_2, \tau_{\mathrm{surf}}) \int \mathrm{d}\tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}_2, \tau_{\mathrm{tally}}) \int \mathrm{d}\tau_{\mathrm{xs}}\, p_{\mathrm{xs}}(\boldsymbol{P}_2, \tau_{\mathrm{xs}})$$
$$\times \int \mathrm{d}q\, (q + \tau_{\mathrm{geom}} + \tau_{\mathrm{surf}} + \tau_{\mathrm{tally}} + \tau_{\mathrm{xs}})\, \upsilon(\boldsymbol{P}_2, q). \quad (4.12)$$

Because

$$\Upsilon(\boldsymbol{P}) = \int \mathrm{d}q\, q\, \upsilon(\boldsymbol{P}, q) \quad (4.13)$$

by definition, Eq. (4.12) becomes

$$\Upsilon_{\mathcal{S}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1)$$
$$\times \left\{ \int \mathrm{d}\tau_{\mathrm{geom}}\, \tau_{\mathrm{geom}}\, p_{\mathrm{geom}}(\boldsymbol{P}_1, \tau_{\mathrm{geom}}) + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \right.$$
$$\left. \times \left[ \Upsilon(\boldsymbol{P}_2) + \int \mathrm{d}\tau_{\mathrm{surf}}\, \tau_{\mathrm{surf}}\, p_{\mathrm{surf}}(\boldsymbol{P}_2, \tau_{\mathrm{surf}}) + \int \mathrm{d}\tau_{\mathrm{tally}}\, \tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}_2, \tau_{\mathrm{tally}}) + \int \mathrm{d}\tau_{\mathrm{xs}}\, \tau_{\mathrm{xs}}\, p_{\mathrm{xs}}(\boldsymbol{P}_2, \tau_{\mathrm{xs}}) \right] \right\}. \quad (4.14)$$

by distributing the sum and performing all integrations.

Finally, compute the analog non-absorptive collision FTE using Eq. (4.4c) as

$$\begin{aligned}\Upsilon_{\mathcal{E}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\tau_{\mathrm{geom}}\, p_{\mathrm{geom}}\left(\boldsymbol{P}_1, \tau_{\mathrm{geom}}\right) \\ \times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\tau_{\mathrm{col}}\, p_{\mathrm{col}}\left(\boldsymbol{P}_2, \tau_{\mathrm{col}}\right) \int \mathrm{d}\tau_{\mathrm{tally}}\, p_{\mathrm{tally}}\left(\boldsymbol{P}_2, \tau_{\mathrm{tally}}\right) \\ \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int \mathrm{d}\tau_{\mathrm{xs}}\, p_{\mathrm{xs}}\left(\boldsymbol{P}_3, \tau_{\mathrm{xs}}\right) \\ \times \int \mathrm{d}\tau\, \tau\, \upsilon\left(\boldsymbol{P}_3, \tau - \left(\tau_{\mathrm{geom}} + \tau_{\mathrm{col}} + \tau_{\mathrm{tally}} + \tau_{\mathrm{xs}}\right)\right). \end{aligned} \tag{4.15}$$

Again make the substitution $q = \tau - \left(\tau_{\mathrm{geom}} + \tau_{\mathrm{col}} + \tau_{\mathrm{tally}} + \tau_{\mathrm{xs}}\right)$ and $\mathrm{d}q = \mathrm{d}\tau$ to yield

$$\begin{aligned}\Upsilon_{\mathcal{E}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\tau_{\mathrm{geom}}\, p_{\mathrm{geom}}\left(\boldsymbol{P}_1, \tau_{\mathrm{geom}}\right) \\ \times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\tau_{\mathrm{col}}\, p_{\mathrm{col}}\left(\boldsymbol{P}_2, \tau_{\mathrm{col}}\right) \int \mathrm{d}\tau_{\mathrm{tally}}\, p_{\mathrm{tally}}\left(\boldsymbol{P}_2, \tau_{\mathrm{tally}}\right) \\ \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int \mathrm{d}\tau_{\mathrm{xs}}\, p_{\mathrm{xs}}\left(\boldsymbol{P}_3, \tau_{\mathrm{xs}}\right) \\ \times \int \mathrm{d}q\, \left(q + \tau_{\mathrm{geom}} + \tau_{\mathrm{col}} + \tau_{\mathrm{tally}} + \tau_{\mathrm{xs}}\right)\, \upsilon\left(\boldsymbol{P}_3, q\right). \end{aligned} \tag{4.16}$$

As before, expand the sum and perform all integrations to obtain

$$\begin{aligned}\Upsilon_{\mathcal{E}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \Bigg\{ \int \mathrm{d}\tau_{\mathrm{geom}}\, \tau_{\mathrm{geom}}\, p_{\mathrm{geom}}\left(\boldsymbol{P}_1, \tau_{\mathrm{geom}}\right) \\ + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \left[ \int \mathrm{d}\tau_{\mathrm{col}}\, \tau_{\mathrm{col}}\, p_{\mathrm{col}}\left(\boldsymbol{P}_2, \tau_{\mathrm{col}}\right) + \int \mathrm{d}\tau_{\mathrm{tally}}\, \tau_{\mathrm{tally}}\, p_{\mathrm{tally}}\left(\boldsymbol{P}_2, \tau_{\mathrm{tally}}\right) \right] \\ \times \left[ \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \left[ \Upsilon\left(\boldsymbol{P}_3\right) + \int \mathrm{d}\tau_{\mathrm{xs}}\, \tau_{\mathrm{xs}}\, p_{\mathrm{xs}}\left(\boldsymbol{P}_3, \tau_{\mathrm{xs}}\right) \right] \right] \Bigg\}. \end{aligned} \tag{4.17}$$

Performing a sanity check on Eqs. (4.10), (4.14), and (4.17) shows that everything is first-moment ordered. Further, times are contributed in accordance with where it would seem logical (e.g., geometry tracking time is only applied once for the phase-space transform performed by the free-flight transmission kernel $\mathcal{T}$). Thus, the analog FTE according to the linearity described in Eq. (4.5) is

$$\begin{aligned}\Upsilon\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \\ \times \left[ \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \Upsilon\left(\boldsymbol{P}_2\right) + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Upsilon\left(\boldsymbol{P}_3\right) \right] + Q_{\Upsilon}\left(\boldsymbol{P}_0\right) \end{aligned} \tag{4.18a}$$

with corresponding source term

$$Q_{\Upsilon}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \Bigg\{ 3 \int \mathrm{d}\tau_{\mathrm{geom}}\, \tau_{\mathrm{geom}}\, p_{\mathrm{geom}}(\boldsymbol{P}_1, \tau_{\mathrm{geom}}) + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \left[ \int \mathrm{d}\tau_{\mathrm{col}}\, \tau_{\mathrm{col}}\, p_{\mathrm{col}}(\boldsymbol{P}_2, \tau_{\mathrm{col}}) + \int \mathrm{d}\tau_{\mathrm{tally}}\, \tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}_2, \tau_{\mathrm{tally}}) \right] \times \left[ \mathcal{A}(\boldsymbol{P}_2) + \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \int \mathrm{d}\tau_{\mathrm{xs}}\, \tau_{\mathrm{xs}}\, p_{\mathrm{xs}}(\boldsymbol{P}_3, \tau_{\mathrm{xs}}) \right] + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \times \left[ \int \mathrm{d}\tau_{\mathrm{surf}}\, \tau_{\mathrm{surf}}\, p_{\mathrm{surf}}(\boldsymbol{P}_2, \tau_{\mathrm{surf}}) + \int \mathrm{d}\tau_{\mathrm{tally}}\, \tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}_2, \tau_{\mathrm{tally}}) + \int \mathrm{d}\tau_{\mathrm{xs}}\, \tau_{\mathrm{xs}}\, p_{\mathrm{xs}}(\boldsymbol{P}_2, \tau_{\mathrm{xs}}) \right] \Bigg\}. \quad (4.18b)$$

For each of the expected time contribution values, one can introduce the shorthand notation

$$\langle p_{\mathrm{geom}}(\boldsymbol{P}, \tau_{\mathrm{geom}}) \rangle = \int \mathrm{d}\tau_{\mathrm{geom}}\, \tau_{\mathrm{geom}}\, p_{\mathrm{geom}}(\boldsymbol{P}, \tau_{\mathrm{geom}}), \quad (4.19a)$$

$$\langle p_{\mathrm{col}}(\boldsymbol{P}, \tau_{\mathrm{col}}) \rangle = \int \mathrm{d}\tau_{\mathrm{col}}\, \tau_{\mathrm{col}}\, p_{\mathrm{col}}(\boldsymbol{P}, \tau_{\mathrm{col}}), \quad (4.19b)$$

$$\langle p_{\mathrm{tally}}(\boldsymbol{P}, \tau_{\mathrm{tally}}) \rangle = \int \mathrm{d}\tau_{\mathrm{tally}}\, \tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}, \tau_{\mathrm{tally}}), \quad (4.19c)$$

$$\langle p_{\mathrm{xs}}(\boldsymbol{P}, \tau_{\mathrm{xs}}) \rangle = \int \mathrm{d}\tau_{\mathrm{xs}}\, \tau_{\mathrm{xs}}\, p_{\mathrm{xs}}(\boldsymbol{P}, \tau_{\mathrm{xs}}), \quad (4.19d)$$

$$\langle p_{\mathrm{surf}}(\boldsymbol{P}, \tau_{\mathrm{surf}}) \rangle = \int \mathrm{d}\tau_{\mathrm{surf}}\, \tau_{\mathrm{surf}}\, p_{\mathrm{surf}}(\boldsymbol{P}, \tau_{\mathrm{surf}}), \quad (4.19e)$$

to indicate the expected time contributed as a result of each computational event. As such, the analog FTE can be rewritten as

$$\Upsilon(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \times \left[ \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2)\, \Upsilon(\boldsymbol{P}_2) + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3)\, \Upsilon(\boldsymbol{P}_3) \right] + Q_{\Upsilon}(\boldsymbol{P}_0), \quad (4.20a)$$

$$Q_{\Upsilon}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \Bigg\{ 3 \langle p_{\mathrm{geom}}(\boldsymbol{P}, \tau_{\mathrm{geom}}) \rangle + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \left[ \langle p_{\mathrm{col}}(\boldsymbol{P}, \tau_{\mathrm{col}}) \rangle + \langle p_{\mathrm{tally}}(\boldsymbol{P}, \tau_{\mathrm{tally}}) \rangle \right] \left[ \mathcal{A}(\boldsymbol{P}_2) + \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \langle p_{\mathrm{xs}}(\boldsymbol{P}, \tau_{\mathrm{xs}}) \rangle \right] + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \left[ \langle p_{\mathrm{surf}}(\boldsymbol{P}, \tau_{\mathrm{surf}}) \rangle + \langle p_{\mathrm{tally}}(\boldsymbol{P}, \tau_{\mathrm{tally}}) \rangle + \langle p_{\mathrm{xs}}(\boldsymbol{P}, \tau_{\mathrm{xs}}) \rangle \right] \Bigg\}. \quad (4.20b)$$

### 4.4 DXTRAN Future-time Equation

Incorporating DXTRAN into the FTE is similar to doing so with the HSMEs in Chapter 3. Because DXTRAN only applies at non-absorptive collisions in this work, only the non-absorptive collision FTPDF is modified to include additional processing time as a result of playing DXTRAN. As with the HSPDFs, the free-flight transmission kernel $\mathcal{T}$ is changed to the free-flight transmission kernel with DXTRAN truncation, $\mathcal{T}_{\mathrm{DX}}$, for

all FTPDFs. However, this change has no effect on the timing because the integration through $\mathcal{T}_{\mathrm{DX}}$ does not affect timing directly but rather acts on the associated integration $\langle p_{\mathrm{geom}}(\boldsymbol{P}, \tau_{\mathrm{geom}})\rangle$.

However, two new time contribution PDFs are needed. First, $p_{\mathrm{ff}}(\boldsymbol{P}_4, \tau)\,\mathrm{d}\tau$ gives the probability of contributing time $\mathrm{d}\tau_{\mathrm{ff}}$ about $\tau_{\mathrm{ff}}$ to the overall computation from phase-space position $\boldsymbol{P}_4$. This is a result of performing the ray-tracing necessary as part of DXTRAN to determine the optical distance between $\boldsymbol{p}$ and $\boldsymbol{p}_{\mathrm{DX}}$ according to Eq. (2.89). The function $p_{\mathrm{ff}}(\boldsymbol{P}_4, \tau)$ is defined as

$$p_{\mathrm{ff}}(\boldsymbol{P}_4, \tau) = \delta\left(\tau - n_{\mathrm{surf}}(\boldsymbol{P}_4)\,\tau_{\mathrm{ff}}\right), \tag{4.21}$$

where $n_{\mathrm{surf}}(\boldsymbol{P}_4)$ is the number of Monte Carlo geometry surfaces between $\boldsymbol{p}$ and $\boldsymbol{p}_{\mathrm{DX}}$. As noted in Section (2.4.6.2), the time to perform the ray trace scales by the number of Monte Carlo geometry surfaces that the ray trace must step between. In addition, $p_{\mathrm{DX}}(\boldsymbol{P}_4, \tau)$ is defined as

$$p_{\mathrm{DX}}(\boldsymbol{P}_4, \tau) = \delta\left(\tau - \tau_{\mathrm{DX}}\right) \tag{4.22}$$

to represent the balance of the computational time required to perform DXTRAN. As such, it gives the probability of contributing contributing time $\mathrm{d}\tau_{\mathrm{DX}}$ about $\tau_{\mathrm{DX}}$ from phase-space position $\boldsymbol{P}_4$.

As such, the FTPDF for DXTRAN is

$$\begin{aligned}
\upsilon_{\mathrm{DX}}(\boldsymbol{P}_0, \tau)\,\mathrm{d}\tau = &\int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\tau_{\mathrm{geom}}\, p_{\mathrm{geom}}(\boldsymbol{P}_1, \tau_{\mathrm{geom}}) \\
&\times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}\tau_{\mathrm{col}}\, p_{\mathrm{col}}(\boldsymbol{P}_2, \tau_{\mathrm{col}}) \int \mathrm{d}\tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}_2, \tau_{\mathrm{tally}}) \\
&\times \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4) \int \mathrm{d}\tau_{\mathrm{DX}}\, p_{\mathrm{DX}}(\boldsymbol{P}_4, \tau_{\mathrm{DX}}) \int \mathrm{d}\tau_{\mathrm{ff}}\, p_{\mathrm{ff}}(\boldsymbol{P}_4, \tau_{\mathrm{ff}}) \int \mathrm{d}\tau'\, \upsilon(\boldsymbol{P}_4, \tau') \\
&\times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \int \mathrm{d}\tau_{\mathrm{xs}}\, p_{\mathrm{xs}}(\boldsymbol{P}_3, \tau_{\mathrm{xs}})\, \upsilon(\boldsymbol{P}_3, \tau - \tau_{\mathrm{geom}} - \tau_{\mathrm{col}} - \tau_{\mathrm{tally}} - \tau_{\mathrm{DX}} - \tau_{\mathrm{ff}} - \tau' - \tau_{\mathrm{xs}})\,\mathrm{d}\tau,
\end{aligned} \tag{4.23}$$

where temporary working variable $\tau'$ is introduced to represent the future time that the DXTRAN particle contributes. As before, compute the FTE by calculating the first moment of the FTPDF as

$$\begin{aligned}
\Upsilon_{\mathrm{DX}}(\boldsymbol{P}_0) = &\int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\tau_{\mathrm{geom}}\, p_{\mathrm{geom}}(\boldsymbol{P}_1, \tau_{\mathrm{geom}}) \\
&\times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}\tau_{\mathrm{col}}\, p_{\mathrm{col}}(\boldsymbol{P}_2, \tau_{\mathrm{col}}) \int \mathrm{d}\tau_{\mathrm{tally}}\, p_{\mathrm{tally}}(\boldsymbol{P}_2, \tau_{\mathrm{tally}}) \\
&\times \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\mathrm{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4) \int \mathrm{d}\tau_{\mathrm{DX}}\, p_{\mathrm{DX}}(\boldsymbol{P}_4, \tau_{\mathrm{DX}}) \int \mathrm{d}\tau_{\mathrm{ff}}\, p_{\mathrm{ff}}(\boldsymbol{P}_4, \tau_{\mathrm{ff}}) \int \mathrm{d}\tau'\, \upsilon(\boldsymbol{P}_4, \tau') \\
&\times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \int \mathrm{d}\tau_{\mathrm{xs}}\, p_{\mathrm{xs}}(\boldsymbol{P}_3, \tau_{\mathrm{xs}}) \int \mathrm{d}\tau\, \tau\, \upsilon(\boldsymbol{P}_3, \tau - \tau_{\mathrm{geom}} - \tau_{\mathrm{col}} - \tau_{\mathrm{tally}} - \tau_{\mathrm{DX}} - \tau_{\mathrm{ff}} - \tau' - \tau_{\mathrm{xs}})
\end{aligned} \tag{4.24}$$

Make the change of variables $q = \tau - (\tau_{\mathrm{geom}} + \tau_{\mathrm{col}} + \tau_{\mathrm{tally}} + \tau_{\mathrm{DX}} + \tau_{\mathrm{ff}} + \tau' + \tau_{\mathrm{xs}})$ so $\tau = q + \tau_{\mathrm{geom}} + \tau_{\mathrm{col}} +$

$\tau_{\text{tally}} + \tau_{\text{DX}} + \tau_{\text{ff}} + \tau' + \tau_{\text{xs}}$ and $\mathrm{d}q = \mathrm{d}\tau$ so

$$\begin{aligned}
\Upsilon_{\text{DX}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\tau_{\text{geom}}\, p_{\text{geom}}\left(\boldsymbol{P}_1, \tau_{\text{geom}}\right) \\
\times \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\tau_{\text{col}}\, p_{\text{col}}\left(\boldsymbol{P}_2, \tau_{\text{col}}\right) \int \mathrm{d}\tau_{\text{tally}}\, p_{\text{tally}}\left(\boldsymbol{P}_2, \tau_{\text{tally}}\right) \\
\times \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\text{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \int \mathrm{d}\tau_{\text{DX}}\, p_{\text{DX}}\left(\boldsymbol{P}_4, \tau_{\text{DX}}\right) \int \mathrm{d}\tau_{\text{ff}}\, p_{\text{ff}}\left(\boldsymbol{P}_4, \tau_{\text{ff}}\right) \int \mathrm{d}\tau'\, \upsilon\left(\boldsymbol{P}_4, \tau'\right) \\
\times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int \mathrm{d}\tau_{\text{xs}}\, p_{\text{xs}}\left(\boldsymbol{P}_3, \tau_{\text{xs}}\right) \int \mathrm{d}q\ \left(q + \tau_{\text{geom}} + \tau_{\text{col}} + \tau_{\text{tally}} + \tau_{\text{DX}} + \tau_{\text{ff}} + \tau' + \tau_{\text{xs}}\right)\, \upsilon\left(\boldsymbol{P}_3, q\right).
\end{aligned} \tag{4.25}$$

Distributing the summed terms, one obtains

$$\begin{aligned}
\Upsilon_{\text{DX}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Upsilon\left(\boldsymbol{P}_3\right) \\
+ \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\tau_{\text{geom}}\, \tau_{\text{geom}}\, p_{\text{geom}}\left(\boldsymbol{P}_1, \tau_{\text{geom}}\right) \\
+ \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\tau_{\text{col}}\, \tau_{\text{col}}\, p_{\text{col}}\left(\boldsymbol{P}_2, \tau_{\text{col}}\right) \\
+ \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\tau_{\text{tally}}\, \tau_{\text{tally}}\, p_{\text{tally}}\left(\boldsymbol{P}_2, \tau_{\text{tally}}\right) \\
+ \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\text{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \int \mathrm{d}\tau_{\text{DX}}\, \tau_{\text{DX}}\, p_{\text{DX}}\left(\boldsymbol{P}_4, \tau_{\text{DX}}\right) \\
+ \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\text{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \int \mathrm{d}\tau_{\text{ff}}\, \tau_{\text{ff}}\, p_{\text{ff}}\left(\boldsymbol{P}_4, \tau_{\text{ff}}\right) \\
+ \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\text{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \Upsilon\left(\boldsymbol{P}_4\right) \\
+ \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \int \mathrm{d}\tau_{\text{xs}}\, \tau_{\text{xs}}\, p_{\text{xs}}\left(\boldsymbol{P}_3, \tau_{\text{xs}}\right),
\end{aligned} \tag{4.26}$$

written more compactly as

$$\begin{aligned}
\Upsilon_{\text{DX}}\left(\boldsymbol{P}_0\right) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \Bigg\{ \int \mathrm{d}\tau_{\text{geom}}\, \tau_{\text{geom}}\, p_{\text{geom}}\left(\boldsymbol{P}_1, \tau_{\text{geom}}\right) \\
+ \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \left[ \int \mathrm{d}\tau_{\text{col}}\, \tau_{\text{col}}\, p_{\text{col}}\left(\boldsymbol{P}_2, \tau_{\text{col}}\right) + \int \mathrm{d}\tau_{\text{tally}}\, \tau_{\text{tally}}\, p_{\text{tally}}\left(\boldsymbol{P}_2, \tau_{\text{tally}}\right) \right. \\
\left. + \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\text{DX}}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) \left[ \int \mathrm{d}\tau_{\text{DX}}\, \tau_{\text{DX}}\, p_{\text{DX}}\left(\boldsymbol{P}_4, \tau_{\text{DX}}\right) + \int \mathrm{d}\tau_{\text{ff}}\, \tau_{\text{ff}}\, p_{\text{ff}}\left(\boldsymbol{P}_4, \tau_{\text{ff}}\right) + \Upsilon\left(\boldsymbol{P}_4\right) \right] \right. \\
\left. \left. + \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \left[ \int \mathrm{d}\tau_{\text{xs}}\, \tau_{\text{xs}}\, p_{\text{xs}}\left(\boldsymbol{P}_3, \tau_{\text{xs}}\right) + \Upsilon\left(\boldsymbol{P}_3\right) \right] \right] \Bigg\}.
\end{aligned} \tag{4.27}$$

As expected, Eq. (4.27) is identical to Eq. (4.17) except for the addition of the terms corresponding with DXTRAN, included via $\mathcal{B}_{\text{DX}}$, and the associated truncation, included via $\mathcal{T}_{\text{DX}}$.

One can expand the time contribution kernels in Eq. (4.27) to explicitly see how time is contributed analytically. With that, a deterministic solver needs to be modified to incorporate the extra time components. For $\int \mathrm{d}\tau_{\text{DX}}\, \tau_{\text{DX}}\, p_{\text{DX}}\left(\boldsymbol{P}_4, \tau_{\text{DX}}\right)$, the probability of contributing $\mathrm{d}\tau_{\text{DX}}$ about $\tau_{\text{DX}}$ is $\beta\left(\boldsymbol{P}_4\right)$. Similarly, for $\int \mathrm{d}\tau_{\text{ff}}\, \tau_{\text{ff}}\, p_{\text{ff}}\left(\boldsymbol{P}_4, \tau_{\text{ff}}\right)$ the probability of contributing time as a result of performing the DXTRAN free-flight

ray trace calculation is also $\beta(\boldsymbol{P}_4)$. However, in this time contribution $p_{\text{ff}}(\boldsymbol{P}_4, \tau_{\text{ff}})$ also includes the factor $n_{\text{surf}}(\boldsymbol{P}_4)$, which is the number of Monte Carlo surfaces between position $\boldsymbol{x}_4$ and $\delta\Gamma_{\text{DX}}$ along $\boldsymbol{\Omega}_4$. This factor is included to recognize that the time contributed scales linearly with (and thus the PDF is scaled by) the number of free-flight ray trace lookups performed as the ray marches through adjacent Monte Carlo cells. Thus, the DXTRAN time contribution kernels can be written as

$$\beta(\boldsymbol{P}) \langle p_{\text{DX}}(\boldsymbol{P}, \tau_{\text{DX}}) \rangle = \int \mathrm{d}\tau_{\text{DX}}\, \tau_{\text{DX}}\, p_{\text{DX}}(\boldsymbol{P}, \tau_{\text{DX}}), \tag{4.28}$$

$$\beta(\boldsymbol{P})\, n_{\text{surf}}(\boldsymbol{P}) \langle p_{\text{ff}}(\boldsymbol{P}, \tau_{\text{ff}}) \rangle = \int \mathrm{d}\tau_{\text{ff}}\, \tau_{\text{ff}}\, p_{\text{ff}}(\boldsymbol{P}, \tau_{\text{ff}}), \tag{4.29}$$

Considering these time contribution kernels the DXTRAN FTE becomes

$$\begin{aligned}
\Upsilon_{\text{DX}}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \Bigg\{ & \langle p_{\text{geom}}(\boldsymbol{P}, \tau_{\text{geom}}) \rangle \\
& + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \Big[ \langle p_{\text{col}}(\boldsymbol{P}, \tau_{\text{col}}) \rangle + \langle p_{\text{tally}}(\boldsymbol{P}, \tau_{\text{tally}}) \rangle \\
& + \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\text{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4) \Big[ \beta(\boldsymbol{P}_2) \Big[ \langle p_{\text{DX}}(\boldsymbol{P}, \tau_{\text{DX}}) \rangle + n_{\text{surf}}(\boldsymbol{P}_4) \langle p_{\text{ff}}(\boldsymbol{P}_4, \tau_{\text{ff}}) \rangle \Big] + \Upsilon(\boldsymbol{P}_4) \Big] \\
& + \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \Big[ \langle p_{\text{xs}}(\boldsymbol{P}, \tau_{\text{xs}}) \rangle + \Upsilon(\boldsymbol{P}_3) \Big] \Big] \Bigg\}
\end{aligned} \tag{4.30}$$

so the conglomerate FTE incorporating DXTRAN is

$$\begin{aligned}
\Upsilon(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \Bigg\{ & \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \Upsilon(\boldsymbol{P}_2) + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \\
& \times \Big[ \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \Upsilon(\boldsymbol{P}_3) + \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\text{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4) \Upsilon(\boldsymbol{P}_4) \Big] \Bigg\} + Q_{\Upsilon}(\boldsymbol{P}_0)
\end{aligned} \tag{4.31a}$$

$$\begin{aligned}
Q_{\Upsilon}(\boldsymbol{P}_0) = \int \mathrm{d}\Omega_1 \int \mathrm{d}E_1 \int \mathrm{d}w_1 \mathcal{T}_{\text{DX}}(\boldsymbol{P}_0, \boldsymbol{P}_1) \Bigg\{ & 3 \langle p_{\text{geom}}(\boldsymbol{P}, \tau_{\text{geom}}) \rangle \\
& + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \Big[ \langle p_{\text{col}}(\boldsymbol{P}, \tau_{\text{col}}) \rangle + \langle p_{\text{tally}}(\boldsymbol{P}, \tau_{\text{tally}}) \rangle \Big] \Big[ \mathcal{A}(\boldsymbol{P}_2) + \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \langle p_{\text{xs}}(\boldsymbol{P}, \tau_{\text{xs}}) \rangle \\
& + \int \mathrm{d}\boldsymbol{P}_4 \mathcal{B}_{\text{DX}}(\boldsymbol{P}_2, \boldsymbol{P}_4) \Big[ \beta(\boldsymbol{P}_2) \Big[ \langle p_{\text{DX}}(\boldsymbol{P}, \tau_{\text{DX}}) \rangle + n_{\text{surf}}(\boldsymbol{P}_4) \langle p_{\text{ff}}(\boldsymbol{P}_4, \tau_{\text{ff}}) \rangle \Big] \Big] \Big] \\
& + \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \Big[ \langle p_{\text{surf}}(\boldsymbol{P}, \tau_{\text{surf}}) \rangle + \langle p_{\text{tally}}(\boldsymbol{P}, \tau_{\text{tally}}) \rangle + \langle p_{\text{xs}}(\boldsymbol{P}, \tau_{\text{xs}}) \rangle \Big] \Bigg\}.
\end{aligned} \tag{4.31b}$$

Equation (4.31) is the FTE with the DXTRAN biasing kernel and associating timing kernels developed in this work.

### 4.5 Obtaining Monte Carlo Calculation Times

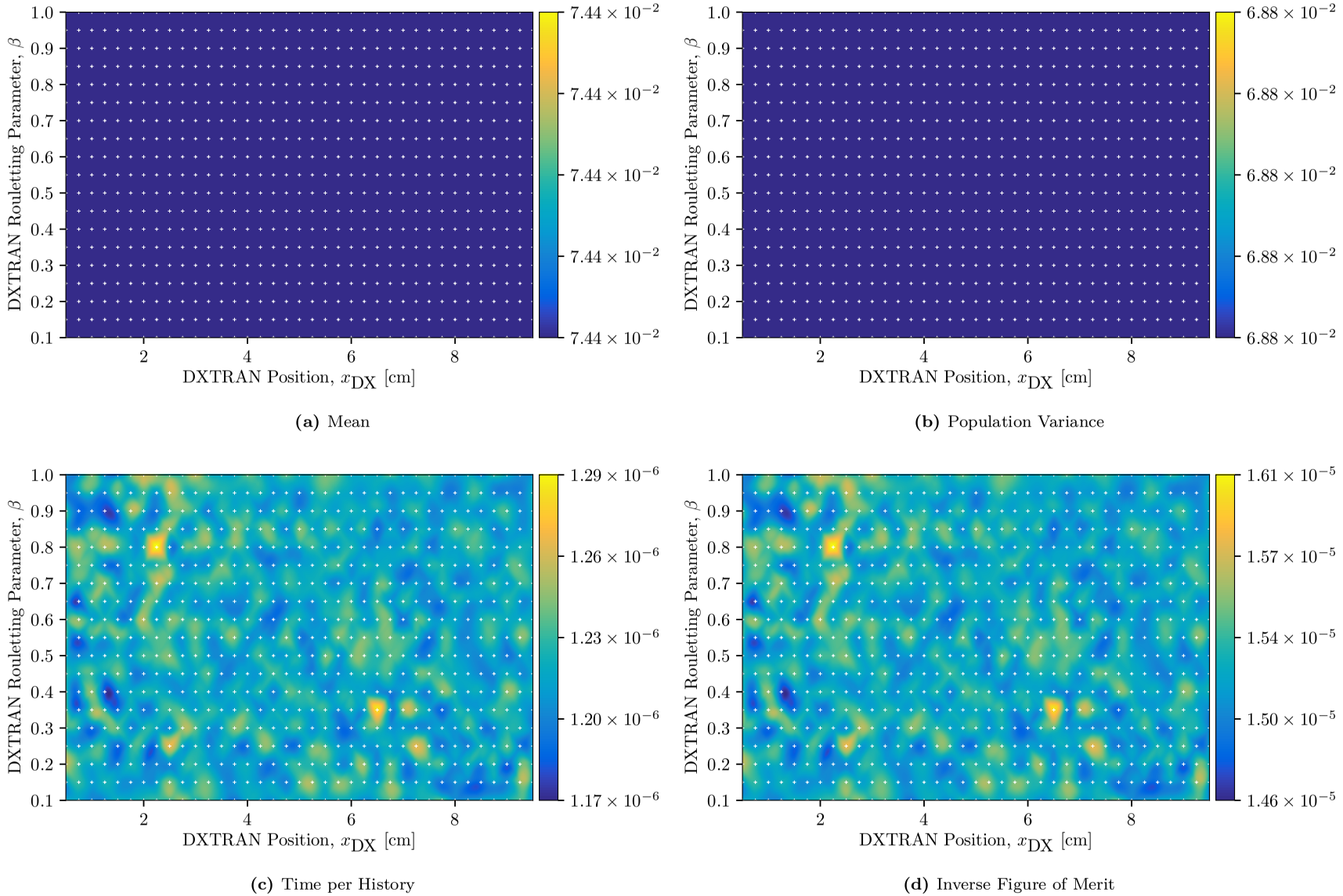
To obtain the Monte Carlo processing times described previously that are required for cost optimization, Valgrind's [105] `callgrind` [106] utility is used. The details of how `callgrind` is used to profile MCNP6 are given in Appendix E. The resulting computational times are given in Table 4.1. Per the notes in Table 4.1, three times are reused from prior work [58]. Note that the weight window times are reused because weight windows are not used in any of the calculations for this work, but this is a required input. The collision and tally times are reused because slightly different values were calculated in Appendix E; however, the times in Table 4.1 gave slightly better agreement for this work. Note that like previous work [58], there is relatively low sensitivity to the precise time values.

#### 4.5.1 Validation of Monte Carlo Calculation Times

The Monte Carlo calculation times shown in Table 4.1 are used with two 1-D test cases to validate that the resulting time and computational cost functions are sensible. The first test case is a mono-energetic homogeneous pure absorber with $\Sigma_{\mathrm{t}} = \Sigma_{\mathrm{a}} = 0.1$ cm$^{-1}$ with $0 \leq x \leq 10$ cm with an isotropic boundary source at $x = 0$ and a leakage current tally at $x = 10$ cm. There are ten 1-cm wide Monte Carlo cells in the problem. In this case, one does not expect any effect from DXTRAN because DXTRAN is only applied at collisions. The second test case is a mono-energetic homogeneous 20% absorber ($\Sigma_{\mathrm{t}} = 0.1$ cm$^{-1}$) with the same size, source, and tally as previously.

In both test cases, the DXTRAN left-hand edge position, $x_{\mathrm{DX}}$, is varied between $x = 0$ and $x = 10$. The MCNP6 cell-wise DXTRAN rouletting parameters are varied between $\beta_c = 0.1$ and $\beta_c = 1.0$. When $\beta_c$ is varied, it is kept constant throughout the problem. By varying these two parameters, computational cost surfaces can be computed with MCNP6 and COVRT and compared. All calculations are performed with one processing thread (serially), on a dedicated computation node on a Los Alamos National Laboratory high-performance computing cluster, one calculation at a time. The calculations are performed in serial to avoid the effect of initializing threading (and subsequent cleanup). Further, the calculations are performed on a dedicated computation node to avoid transient system load caused by other users and/or processes. Finally, the calculations are performed individually to avoid varying system performance caused by the freeing of resources from other calculations. While all calculations fit well within the memory of the system, use only a single thread, and are scheduled to not oversubscribe the node, early work using multiple single-threaded calculations showed a trend in lower computational times at the end when earlier jobs would be finished and less were running on the node.

For the pure absorber test case, the resulting computational cost surface from MCNP calculations is shown in Fig. 4.1d versus the COVRT-predicted computational cost surface in Fig. 4.2d. As expected, the mean and population variance surfaces are constant for both MCNP and COVRT (the problem is analog). However, there is varying time per history despite DXTRAN not being played for the MCNP6 calculations. Despite attempting to make all calculations on as consistent a timing basis as possible, this variation is inescapable. This variation arises from several factors. First, different histories follow different trajectories, leading to different random walks. In principle, this should be captured by the FTE, but the true computational time for a given history depends on memory access times, which depends on locality of the memory that is nearly



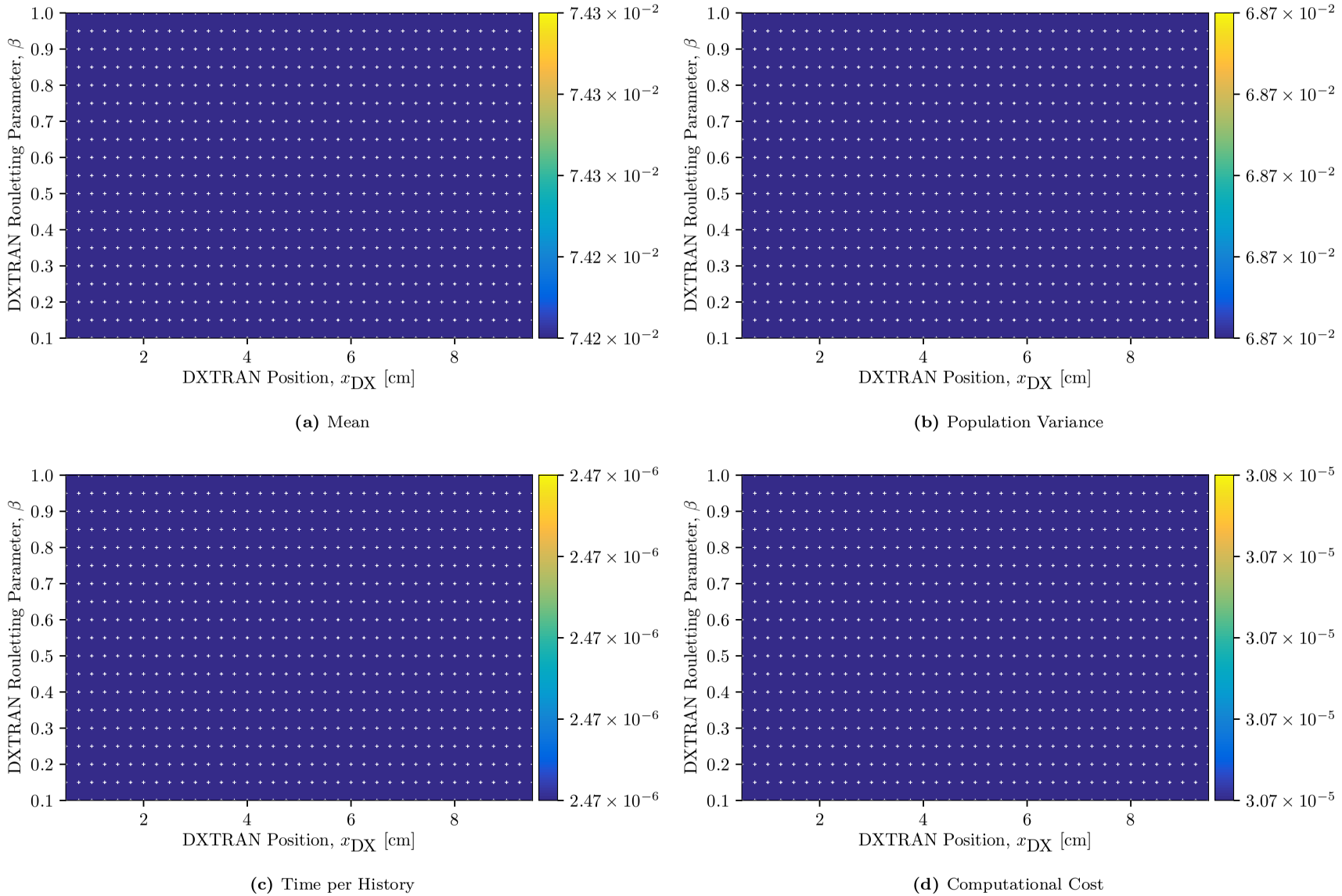

**(a)** Mean

**(b)** Population Variance

**(c)** Time per History

**(d)** Inverse Figure of Merit

**Figure 4.1:** MCNP Pure Absorber Cost Calculation Results




**(a)** Mean

**(b)** Population Variance

**(c)** Time per History

**(d)** Computational Cost

**Figure 4.2:** COVRT Pure Absorber Cost Calculation Results

impossible to predict *a priori*. Furthermore, while the computational node is dedicated to running the MCNP calculation, modern operating systems run numerous background processes that can cause calculating time variation. The FTPDF could capture some of this behavior, but only mean values from `callgrind` in Table E.1 are used. As a result, the maximum-to-minimum ratio of the cost surface in Fig. 4.1 is slightly over 10%. Thus, in a best-case scenario computer timing uncertainty may be as high as 10%. The FTE cannot account for the variation described previously but instead only represents average expected behavior, so the COVRT time and cost surfaces are effectively flat.

For the 20% absorber test case, the resulting computational cost surface from MCNP calculations is shown in Fig. 4.3d versus the COVRT-predicted computational cost surface in Fig. 4.4d. The MCNP mean surface in Fig. 4.3a shows some variation; however, the color bar scale indicates that the surface is effectively flat and agrees with COVRT. Further, the MCNP and COVRT variance surfaces agree to within 1%. The COVRT time surface in Fig. 4.4c shows reasonable behavior. That is, time increases as $x_{\mathrm{DX}}$ increases causing DXTRAN processing to occur more often and causing the number of MCNP6 geometry surfaces to be ray traced through to increase. Further, the time decreases as $\beta$ decreases (causing DXTRAN processing to occur less often). The MCNP time surfaces shows variation such that it is hard to draw an overall conclusion regarding trends. However, the MCNP inverse FOM surface and COVRT computational cost surfaces have matching behaviors though the magnitudes disagree by approximately a factor of three. The agreement in relative behavior is important because the computational cost optimization takes place on the COVRT surface assuming it is an accurate surrogate for the MCNP computational cost behavior. The absolute magnitude doesn't matter but the relationships between the contour, peaks, and valleys is most important to represent accurately.

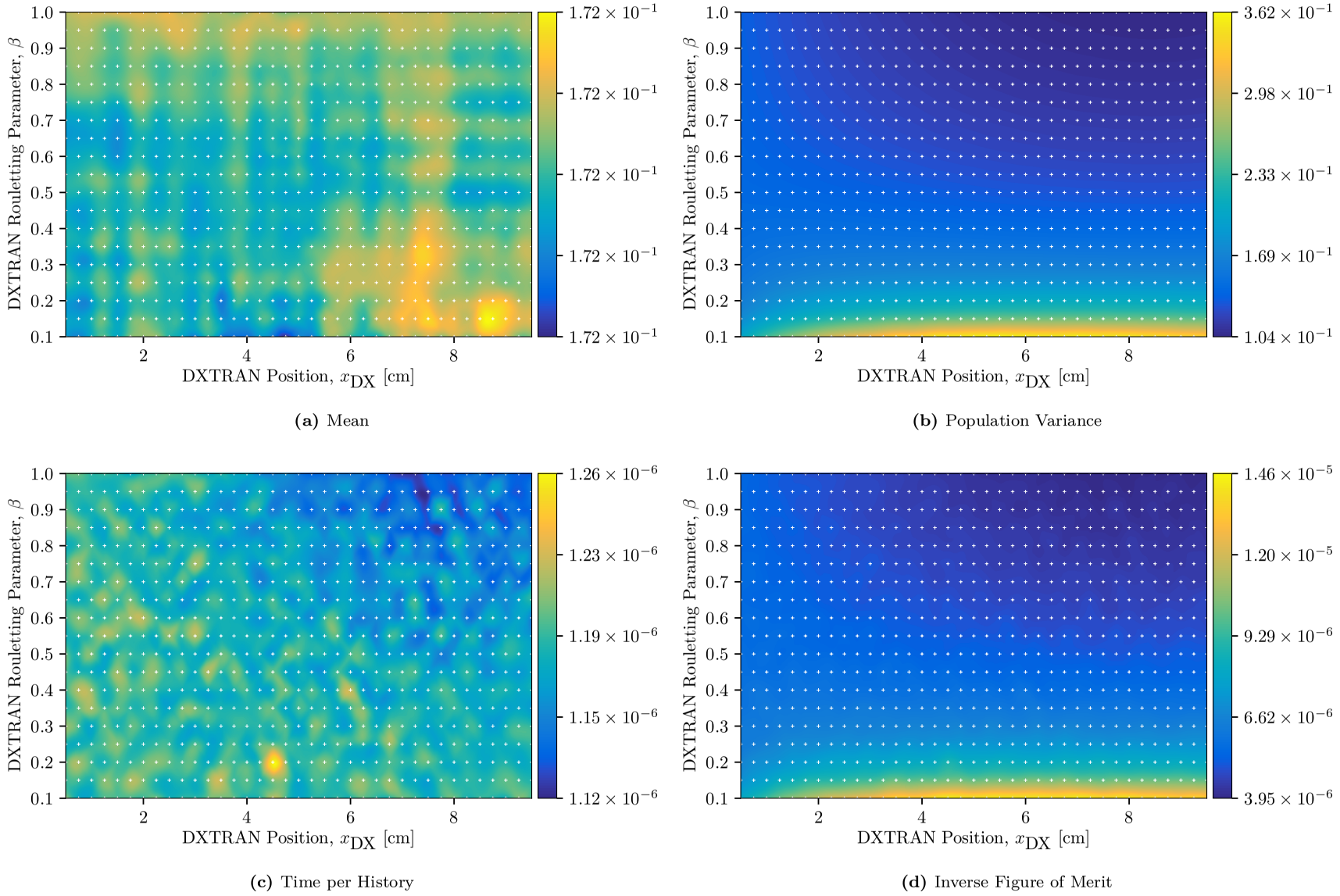
### 4.6 Computational Cost Optimization

With the solution to the HSMEs and the FTE, the computational cost of a Monte Carlo calculation with DXTRAN can be predicted using the analytic equivalent of Eq. (2.82):

$$\$ = \frac{\left(M_2\left\{\mathrm{VR}\right\} - M_1^2\right) T\left\{\mathrm{VR}\right\}}{M_1^2}. \tag{4.32}$$

By varying the set of variance-reduction parameters used ($\{\mathrm{VR}\}$), different computational cost values will result arising from the competing effects of reducing the variance while increasing computational time and vice versa. Thus, one can use an optimization algorithm to identify the set of variance reduction parameters that balances the reduction in variance with the increase in computational time to produce a minimized computational cost. Recognize that $M_1$ is invariant with respect to variance reduction so normalizing by its square is unnecessary to compute a minimum cost, but the form of Eq. (4.32) permits direct comparison with the Monte Carlo computational cost. Further, the minimum computational cost corresponds to the maximum Monte Carlo FOM. Thus, the resulting set of variance-reduction parameters are expected to produce a maximally efficient Monte Carlo calculation in a FOM sense.

Prior work [58] used a gradient descent algorithm to perform optimization. It estimated the local gradient by dithering the variance-reduction parameters, performing the resulting calculations for \$, and then traversing the parameter space in the direction of the most-negative estimated gradient. It continued this process until it determined that all gradients were positive indicating that it had reached a minimum point. This minimum



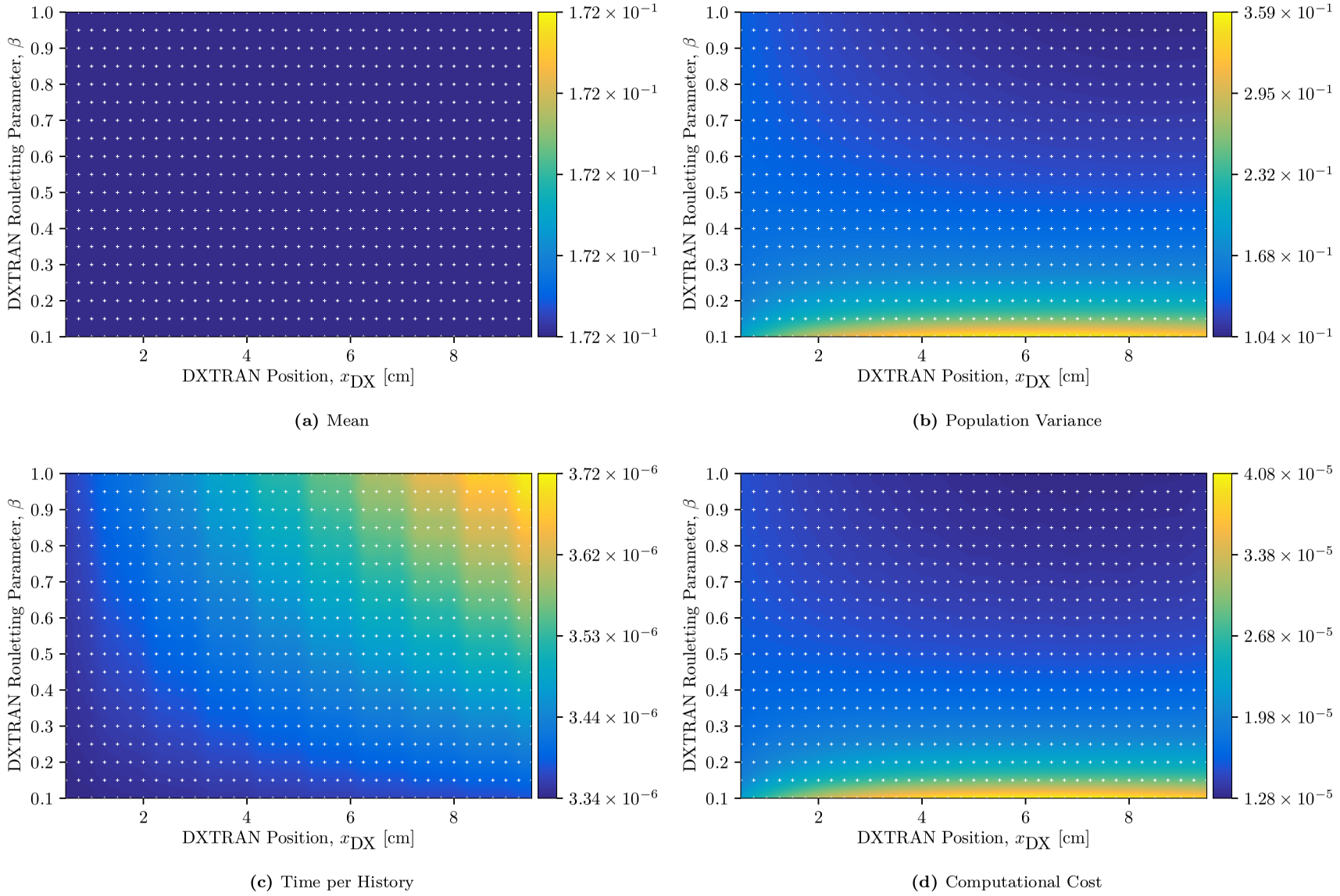

**(a)** Mean

**(b)** Population Variance

**(c)** Time per History

**(d)** Inverse Figure of Merit

**Figure 4.3:** MCNP 20% Absorber Cost Calculation Results




**(a)** Mean

**(b)** Population Variance

**(c)** Time per History

**(d)** Computational Cost

**Figure 4.4:** COVRT 20% Absorber Cost Calculation Results

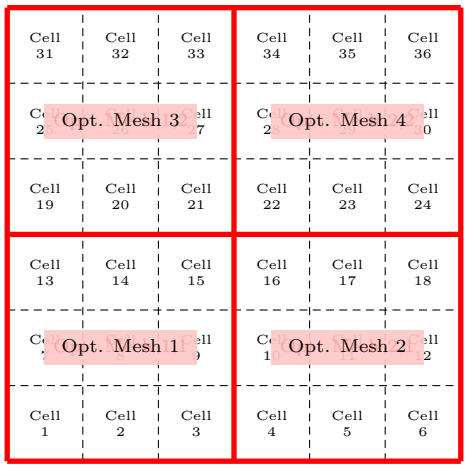



**Figure 4.5:** Representative Monte Carlo Cells and Overlaid Optimization Mesh

is not guaranteed to be global but instead may be local.

Because gradient descent is recognized to be relatively slow compared to competing minimization techniques such as conjugate gradient [107] and GMRES [108], an artificial optimization mesh was overlaid atop the problem. An example of such a mesh in a 2-D problem is shown in Fig. 4.5. The optimization mesh is more coarse than the underlying spatial mesh and is used to define boundaries of basis functions upon which the Monte Carlo cell-specific variance-reduction parameters are assigned. This reduces the number of parameters that must be optimized (from the number of Monte Carlo geometry cells to the number of optimization mesh cells). The coefficients of the basis functions (ultimately, constant-in-cell basis functions) are what the gradient descent algorithm solved for. The Monte Carlo geometry cell-specific variance-reduction parameters are then assigned based upon the basis function values.

For this work, the gradient descent algorithm within COVRT and associated optimization mesh are eschewed in favor of coupling COVRT to an external Python-based optimization tool. Based on empirical studies, PyNOMAD's MADS optimization algorithm [109, 110] was ultimately selected and used outside of COVRT.

Some discussion is provided of the optimization approaches explored but ultimately not pursued further. This is done for two reasons: (1) to illustrate potential pitfalls for future researchers and (2) to credit otherwise capable tools and algorithms that, while not helpful for this work, may benefit future researchers.

### 4.6.1 Constrained & Bounded Python-based Optimization Techniques

Four minimization / optimization techniques from Python's SciPy [111] package are first explored. These approaches are investigated first because SciPy is a commonly available extension to Python and, as such, is expected to be easily retrievable by anyone interested in reproducing or using this work. However, in the course of investigating these packages various deficiencies are identified including extensive premature convergence. As such, three additional techniques provided by independent developers are also evaluated. Ultimately, the PyNOMAD interface to the MADS-based NOMAD package is selected for use with this work. Each technique explored is described briefly in the following subsections.

All optimization algorithms must be capable of optimizing for bounded and constrained parameters. The requirements for bounds exist because $0 < \beta_c \leq 1$ and the DXTRAN region must exist within the problem (i.e., $x_{\min} \leq x_{\text{DX,min}} < x_{\max}$, $x_{\min} < x_{\text{DX,max}} \leq x_{\max}$, $y_{\min} \leq y_{\text{DX,min}} < y_{\max}$, $y_{\min} < y_{\text{DX,max}} \leq y_{\max}$, $z_{\min} \leq z_{\text{DX,min}} < z_{\max}$, and $z_{\min} < z_{\text{DX,max}} \leq z_{\max}$). The requirement for constraints arise from the DXTRAN region needing to have positive volume (i.e., $x_{\text{DX,min}} < x_{\text{DX,max}}$, $y_{\text{DX,min}} < y_{\text{DX,max}}$, and $z_{\text{DX,min}} < z_{\text{DX,max}}$). In addition, the optimization algorithm must not require an analytic gradient (i.e., it must be able to estimate a numeric gradient or be entirely gradient free). These limitations restrict the type of optimization algorithms available.

For each optimization algorithm, several 1-D test cases are used to test its behavior and form an overall opinion on suitability for this work. The reason being that if an optimizer cannot quickly and reasonably accurately predict computational cost minima for simple 1-D test cases then there is little hope that it will succeed with larger and more-difficult 2-D cases.

#### 4.6.1.1 Evaluating the Python-based Optimization Techniques

To evaluate optimization techniques for this work, two homogeneous mono-energetic 1-D test cases are constructed (see Fig. 6.2). Both test cases feature an isotropic surface source at $x = 0$, a leakage current tally at $x = 10$ cm, and the 10-cm region is divided into 10 1-cm Monte Carlo cells. The entire region is filled with a 20% absorbing material with $\Sigma_{\text{t}} = 0.1$ cm$^{-1}$.

Optimization Test Case 1 features a fixed DXTRAN region with left surface at $x_{\text{DX}} = 5$ cm and right surface at $x = 10$ cm. Thus, optimization is only applied to the DXTRAN $\beta_c$ values for the ten cells. Based on the previous work to characterize and validate Monte Carlo times, it is expected that the optimal $\beta_c$ will be one, so the initial values given to the optimization algorithms are $\beta_c = 0.5$ (a bad initial "guess").

Optimization Test Case 2 features a resizable (i.e., optimizable) DXTRAN region with the left surface initially at $x_{\text{DX}} = 5$ cm and fixed right surface at $x = 10$ cm. Again, the initial guess for all rouletting parameters is set to 0.5 for all cells. Based on the previous work to characterize and validate Monte Carlo times, it is expected that the optimal value for $x_{\text{DX}}$ is approximately 7 cm and that all $\beta_c$ values be set to one.

#### 4.6.1.2 Sequential Least SQuares Programming (SLSQP)

The Sequential Least SQuares Programming (SLSQP) [112] algorithm provided by the SciPy `optimize` package is tested first. This technique estimates numerical gradients by varying the optimization parameters. The estimated gradients are then used to iteratively seek a set of optimization parameters.

While this approach showed the most promise relative to other SciPy optimization algorithms, it was ultimately not used because of its inability to accommodate shallow gradient solutions that arise from low-sensitivity to $\beta_c$ variation and from artificial zero-gradient results from small variations in the DXTRAN region boundaries. Often SLSQP would identify a locally flat region on the cost surface as being a local minimum and would falsely converge. Regardless, it did appear the most robust of the SciPy optimization algorithms and is a recommended starting point.

#### 4.6.1.3 Constrained Optimization BY Linear Approximation (COBYLA)

The Constrained Optimization BY Linear Approximation (COBYLA) [113] algorithm provided by the SciPy `optimize` package is tested next. This technique does not require gradients. Instead, it creates a linear approximation of the objective function and optimization parameters over a subset of the whole space (the subset is the trust region). A simplex then processes the parameters in the trust region to find the optimum value. Based on that optimum value, the trust region is modified to broaden the search for the optimal set of parameters according to the linearized objective function.

This algorithm has an execution speed comparable to SLSQP but was ultimately not used because of its tendency to find different optimal values depending on the quality of the initial guess for the optimal parameter values. This represents a type of false convergence and leads to a lack of reproducibility.

#### 4.6.1.4 Basinhopping

The basinhopping [114] algorithm is a global optimization algorithm provided by the SciPy `optimize` package that is also tested. The general approach is to perform global stepping over the optimization parameter space and then performing local minimization within each step. This attempts to prevent the algorithm from falling into a local minimum. However, care must be taken to avoid having multiple local minima in a single global step.

This approach is ultimately not used because of its slow speed to solution and the need for a user to carefully tune the global step size to balance speed to solution with solution quality.

#### 4.6.1.5 Differential Evolution

The differential evolution [115] algorithm is another global optimization algorithm provided by the SciPy `optimize` package that is tested. This algorithm is inspired by the biological theory of evolution where vectors of optimization parameters are followed and those deemed most suitable (most optimal) are allowed to define search vectors for future iterations. Thus, the search vectors for the optimal optimization parameters *evolve* as a function of prior suitability. By working in this way, this method does not require a numeric gradient.

This approach is ultimately not used because of its slow speed to solution relative to SLSQP and COBYLA while producing "optimum" parameters of comparable quality (i.e., not necessarily global minima).

#### 4.6.1.6 Particle Swarm

The particle swarm optimization [116] algorithm is another global optimization algorithm that is tested. It operates by creating "particles" across the optimization parameter space with initially random velocities that define subsequent search vectors. Each particle is tested (evaluated for the objective function value) and the individual results are communicated to all members of the swarm. Then, the velocities are updated to steer the particles toward the swarm-obtained set of parameters that produce the minimum value, the particles are moved according to the new velocity, and the objective functions are re-evaluated. Thus, the particles of the swarm traverse the parameter space but eventually congregate around the swarm-found minimum value.

This approach is ultimately not used because of its slow speed to solution.

#### 4.6.1.7 Markov-chain Monte Carlo

A Markov-chain Monte Carlo algorithm is commonly used to construct Markov chains to find parameters that fit an analytic model to a series of experimental values. This is done by defining a function to compute the sum of squared differences between the experimental values and the analytic model given the optimization parameters tested at the particular step of the random walk. For this work the approach is modified to set the objective function as the sum-of-squares function. The MCMC algorithm then produces Markov chains normally but rather than trying to minimize the sum-of-squares function it is truly minimizing the objective function. For this work, the Delayed Rejection Adaptive Metropolis (DRAM) approach is used exclusively in the PyMCMC package [117].

This approach is ultimately not used because of its inability to adequately statistically converge the set of optimization parameters in a reasonable amount of time.

#### 4.6.1.8 Mesh Adaptive Direct Search (MADS)

The Mesh Adaptive Direct Search (MADS) algorithm for constrained optimization [118] implemented in the NOMAD software suite [109, 110] was ultimately selected for use in this work. The main reason for this is that it operates using a black-box optimization function and also does not estimate numeric gradients. However, it is deterministic in its execution so it is not subject to random fluctuations introduced in techniques such as particle swarm optimization or Markov-chain Monte Carlo. When the 1-D test case calculations are made, the answers are found quickly (as fast as the quickest of the other methods) and consistently (without random fluctuation as noted previously and without variation regardless of the quality of the initial guess). Further, MADS does not exhibit false convergence for the 1-D test cases used to examine it.

The NOMAD user guide [109] states that NOMAD is not a suitable tool if the number of optimization parameters is large. Naturally, this quantity is associated with the time needed to evaluate the cost function, but private communication with a colleague [119] noted that NOMAD can be challenged by problems with 100s of parameters. Thus, for nominal Monte Carlo calculations NOMAD is expected to be acceptable. Indeed, in Chapter 7 using MADS via NOMAD for this work is shown to work for up to 104 parameters. However, Chapter 7 also demonstrates some undesirable MADS behavior for some 2-D calculations that merits future investigation.

# Chapter 5

## Numerical Implementation

This chapter acts as a bridge between the integral HSMEs and FTE derived in Chapters 3 and 4, respectively, and the discretization and solution approaches described in Chapter 2[1]. In general, this chapter focuses on the HSMEs because the FTE is solved using similar techniques. However, it is noted when the FTE requires special treatment.

This chapter begins with an overview of the methods used to recast the HSMEs into an integro-differential form. This is demonstrated by example using a 1-D, mono-energetic, $S_2$ calculation with a homogeneously distributed and isotropically scattering material in $0 \leq x \leq X$, containing a forward isotropic surface source at the left-hand boundary, a leakage current tally at the right-hand boundary, and a DXTRAN region occupying the right-hand portion of the geometry in the region $x_{\mathrm{DX}} \leq x \leq X$.

Next, two algorithms are introduced to solve the DXTRAN HSMEs numerically and demonstrated using the aforementioned example. The first algorithm, used early during this work but later abandoned, directly represents Monte Carlo particle behavior in the deterministic solver but necessitates both significant modification to the sweep-based solver and additional bookkeeping (referred to as the separated-moment approach). The second algorithm, used throughout this work, implicitly treats the free-flight truncation required by DXTRAN (referred to as the combined-moment approach). The numeric solution of $\Psi_1$ is described in detail for these two approaches. Next, the algorithms to solve for $Q_2$ and $\Psi_2$ are given.

Next, this chapter describes the modifications necessary to incorporate DXTRAN regions into the COVRT software [58]. An overview is given for how COVRT operates in a once-through manner to predict Monte Carlo mean, population variance, time, and computational cost. Finally, the chapter concludes with a brief summary of the modifications made to COVRT that allow it to be driven by Python, using Python-based optimizers as described in Chapter 4, to compute computational-cost optimal DXTRAN size, position, and rouletting parameters.

[1] One should recognize that when these topics are brought together, there is an opportunity for confusion because the term "moment" is used to mean two different things. In Chapter 2, angular moments of cross sections and fluxes are computed to calculate the scattering source. In Chapter 3, statistical moments (i.e., the HSMEs) are the quantity of interest. In this chapter, a reasonable effort is made to differentiate between these two types of moments, but unfortunately doing so can be quite verbose. As such, if a "moment" is referred to it generally means the statistical moment.

## 5.1 Recasting HSMEs into an Integro-differential Form

To recast the path integral forms of the HSMEs into an integro-differential form, a three-step process is used. The process is

1. Simplify the integro-differential equation using any knowledge of the problem at hand. For example, the aforementioned motivating example permits limiting the scattering order to isotropic, reducing the spatial dependence to 1-D, and eliminating energy dependence. The inhomogeneous volume source, $Q$, also disappears.
2. Simplify the HSMEs to remove any tallies not under consideration. In this case, there are no collision or expected track-length estimators, so any associated scoring terms disappear leaving only one for the terminal leakage-current estimator. For the FTE, similar simplifications are made such that no time is contributed for events that do not happen (e.g., time is contributed to process collisions but not to record scores to collision estimators).
3. Incorporate the HSME and FTE random walks modifications into the integro-differential migration operators. For example, the random walk concluding with non-absorptive collision and scattering emergence is incorporated by modifying the scattering source term. Similarly, random walks that depend on surface crossings are incorporated by modifying the streaming operator.

These three steps are described next. They are followed by the algorithms to numerically solve for $\Psi_1$, $Q_2$, and $\Psi_2$. In the upcoming discussion, because strictly weight-independent variance reduction is considered the technique of extracting the weight coefficient as a scaling on the statistical moment [Section 3.6] is used and a weight index is eschewed except when the final inner product to calculate $M_m$, as a reminder. Furthermore, in this section only the combined-moment approach is considered because it eliminates the need to alter the streaming operator and because it is the technique ultimately applied in this work.

### 5.1.1 Forming an Appropriate, and Simplified, Integro-differential Adjoint Equation

Using the simplifications described previously, one may begin recasting the HSMEs into an integro-differential form by transforming the 1-D particle transport equations from Chapter 2, Eqs. (2.48)–(2.52), to the adjoint form using Section 2.2.3 to obtain:

$$-\mu_n \left( \frac{f^*_{i+1/2,n} - f^*_{i-1/2,n}}{\Delta x_i} \right) + \Sigma_\mathrm{t} f^*_{i,n} = q^*_{i,n}\,,\ i = 1, 2, \ldots, I\,,\ n = 1, 2, \tag{5.1a}$$

$$f^*_{1/2,n} = f^*_{\mathrm{b},n}\,,\ n = 1, \tag{5.1b}$$

$$f^*_{1/2,n} = f^*_{\mathrm{b},n}\,,\ n = 2, \tag{5.1c}$$

$$f^*_{I+1/2,n} = f^*_{\mathrm{b},n}\,,\ n = 1, \tag{5.1d}$$

$$f^*_{I+1/2,n} = f^*_{\mathrm{b},n}\,,\ n = 2, \tag{5.1e}$$

where the $S_2$ Gauss-Legendre quadrature (see Appendix C) has been applied with

$$\mu_1 = -\sqrt{\frac{1}{3}}, \tag{5.2a}$$

$$\mu_2 = \sqrt{\frac{1}{3}}, \tag{5.2b}$$

$$\omega_1 = \omega_2 = \frac{1}{2}, \tag{5.2c}$$

and the boundary conditions are written for each boundary and direction but are left unspecified for now. The combined source term is

$$q^*_{i,n} = \sum_{r=1}^{1} \Sigma_{\mathrm{s},l_r} f^*_{i,r} + Q^*_{i,n} = \Sigma_{\mathrm{s},0} f^*_{i,r=1} + Q^*_{i,n} \tag{5.3}$$

where $\Sigma_{\mathrm{s},l_r}$ are the angular moments of the non-absorptive collision cross section given by Eq. (2.66) and $f_{i,r}$ are the angular moments of the flux given by Eq. (2.67b). For the isotropic case considered here, $r = 1$ so $l_r = 0$. The inhomogeneous source term, $Q^*_{i,n}$, is kept and left unspecified because there is no inhomogeneous source for $\Psi_1$ but there is for $\Psi_2$. The auxiliary equation used to relate cell-edge and cell-center values is Eq. (2.49),

$$f^*_{i,n} = \frac{f^*_{i+1/2,n} - f^*_{i-1/2,n}}{2}. \tag{5.4}$$

### 5.1.2 Casting $\Psi_1$ into an Integro-differential Form

One can simplify $\Psi_1$ and $Q_1$ for the example case. By considering a 1-D mono-energetic problem, one can temporarily redefine $\boldsymbol{P} = (x, \mu, w)$. Simplifying Eq. (3.45) for the example case, one obtains

$$\Psi_1(\boldsymbol{P}_0) = \int \mathrm{d}\mu_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}(\boldsymbol{P}_2, \boldsymbol{P}_3) \Psi_1(\boldsymbol{P}_3) + Q_1(\boldsymbol{P}_0), \tag{5.5a}$$

$$Q_1(\boldsymbol{P}_0) = \int \mathrm{d}\mu_1 \int \mathrm{d}w_1 \mathcal{T}(\boldsymbol{P}_0, \boldsymbol{P}_1) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}(\boldsymbol{P}_1, \boldsymbol{P}_2) \int \mathrm{d}s\, s\, p_{\mathcal{S}}(\boldsymbol{P}_2, s), \tag{5.5b}$$

because the surface-crossing kernel, $\mathcal{S}$, in $\Psi_1$ does not modify $\Psi_1$. Because there are no collision estimators in $Q_1$, several terms within also disappear. Indeed, the only source term is the scoring function on the right-hand boundary (consistent with a traditional adjoint flux scenario). Recasting Eq. (5.5) using Section 2.2.4 to the integro-differential form given previously, one obtains

$$-\mu_n \left( \frac{\Psi_{1,i+1/2,n} - \Psi_{1,i-1/2,n}}{\Delta x_i} \right) + \Sigma_{\mathrm{t}} \Psi_{1,i,n} = q^*_{i,n}\,,\ i = 1, 2, \ldots, I\,,\ n = 1, 2, \tag{5.6a}$$

$$\Psi_{1,1/2,n} = 0\,,\ n = 1, \tag{5.6b}$$

$$\Psi_{1,1/2,n} = 0\,,\ n = 2, \tag{5.6c}$$

$$\Psi_{1,I+1/2,n} = 0\,,\ n = 1, \tag{5.6d}$$

$$\Psi_{1,I+1/2,n} = w\,,\ n = 2, \tag{5.6e}$$

where $\Psi_{1,I+1/2,n} = w$ in Eq. (5.6e) from $Q_1$ because the leakage current tally defined at the right-hand boundary will score $w$ when a particle passes through the corresponding surface. The combined source is

$$q^*_{i,n} = \Sigma_{\mathrm{s},0}\Phi_{1,i,r=1} \tag{5.7}$$

because there is no cell-centered inhomogeneous source in $Q_1$. The scattering source described by Eq. (5.7) corresponds to the term

$$\int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right)$$

because it moves particles through the non-absorptive collision process to transition them from (in a forward sense) phase spaces $\boldsymbol{P}_1$ to $\boldsymbol{P}_3$, which they then continue their random walk from. The scattering source iteration process used accumulates the effect of collision 1, 2, 3, and so on until convergence (in a forward sense). For the adjoint-like calculations performed here, the scattering source iteration process accumulates the effect of collision $N$ before scoring, then $N-1$, $N-2$, and so on.

### 5.1.3 Casting $\Psi_2$ into an Integro-differential Form

One can also simplify $\Psi_2$ and $Q_2$ in Eq. (3.74) for the motivating case to obtain

$$\begin{aligned}
\Psi_2\left(\boldsymbol{P}_0\right) = & \underbrace{\int \mathrm{d}\mu_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_2\left(\boldsymbol{P}_3\right)}_{\textcircled{1}} \\
& + \int \mathrm{d}\mu_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \\
& \underbrace{\quad \times \int \mathrm{d}\mu_1 \frac{p^2_{\mathrm{ff}}\left(x_2, x_{\mathrm{DX}}, \mu_4\right)}{\beta\left(x_2\right)} \frac{p^2\left(\mu_2 \rightarrow \mu_4\right)}{\tilde{p}\left(\mu_2 \rightarrow \mu_4\right)} \Psi_2\left(x_{\mathrm{DX}}, \mu_4, w_2\right)}_{\textcircled{2}} \\
& + Q_2\left(\boldsymbol{P}_0\right),
\end{aligned} \tag{5.8a}$$

$$\begin{aligned}
Q_2\left(\boldsymbol{P}_0\right) = & \underbrace{\int \mathrm{d}\mu_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{S}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}s\, s^2\, p_{\mathcal{S}}\left(\boldsymbol{P}_2, s\right)}_{\textcircled{3}} \\
& + 2 \int \mathrm{d}\mu_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \\
& \underbrace{\quad \times \int \mathrm{d}\mu_4 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) p_{\mathrm{ff}}\left(x_2, x_{\mathrm{DX}}, \mu_4\right) \Psi_1\left(x_{\mathrm{DX}}, \mu_4, w_2\right),}_{\textcircled{4}}
\end{aligned} \tag{5.8b}$$

where

(1) is the analog non-absorptive collision operator that applies inside the DXTRAN region and to non-DXTRAN particles in the non-DXTRAN region,

(2) is the non-analog (DXTRAN) non-absorptive collision operator that applies *outside* the DXTRAN

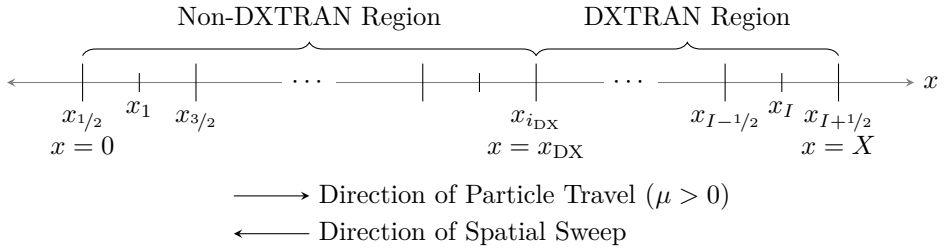



**Figure 5.1:** DXTRAN Region Segregation and Direction of Spatial Sweep versus Direction of Particle Travel

region to non-DXTRAN particles as calculated by their contribution to the second moment at $x_{\mathrm{DX}}$,

(3) is the second-moment ordered surface-scoring function comparable to the first-moment ordered source term in Eq. (5.5),

(4) is the joint source term that drives $\Psi_2$ caused by DXTRAN particles generating additional DXTRAN particles as described in Section 2.4.6.6.

Again, in $\Psi_2$ the surface-crossing kernel, $\mathcal{S}$, does not modify $\Psi_2$ so those terms disappear. Also, all non-scoring terms disappear from $Q_2$ because there are no collision or expected track-length estimators used. Because the leakage estimator is terminal there is no joint surface-scoring term. Recasting Eq. (5.8) using Section 2.2.4 to the integro-differential forms given previously, one obtains

$$-\mu_n \left( \frac{\Psi_{2,i+1/2,n} - \Psi_{2,i-1/2,n}}{\Delta x_i} \right) + \Sigma_{\mathrm{t}} \Psi_{2,i,n} = q^*_{i,n} \,,\; i = 1, 2, \ldots, I \,,\; n = 1, 2, \tag{5.9a}$$

$$\Psi_{2,1/2,n} = 0 \,,\; n = 1, \tag{5.9b}$$

$$\Psi_{2,1/2,n} = 0 \,,\; n = 2, \tag{5.9c}$$

$$\Psi_{2,I+1/2,n} = 0 \,,\; n = 1, \tag{5.9d}$$

$$\Psi_{2,I+1/2,n} = w^2 \,,\; n = 2, \tag{5.9e}$$

where $\Psi_{1,I+1/2,n} = w^2$ in Eq. (5.9e) from $Q_2$ Term (3) because the leakage current tally defined at the right-hand boundary will score $w^2$ to the second moment when a particle passes through the corresponding surface. The combined source $q^*_{i,n}$ differs depending on whether mesh cell $i$ is inside or outside the DXTRAN region. The combined source is

$$q^*_{i,n} = \Sigma_{\mathrm{s},0} \Phi_{2,i,r=1} \tag{5.10}$$

if $i$ is inside the DXTRAN region and is

$$q^*_{i,n} = \Sigma_{\mathrm{s},0} \Phi_{2,i,r=1} + \sum_{n=1}^{2} \Sigma_{\mathrm{s},n,n'} \frac{p^2_{\mathrm{ff},i,i_{\mathrm{DX}},n}}{\beta_i} \frac{p^2_{n,n'}}{\tilde{p}_{n,n'}} \Psi_{2,i_{\mathrm{DX}},n} + Q_{2,i,n} \tag{5.11}$$

if $i$ is outside the DXTRAN region. The arrangement of mesh cells inside and outside the DXTRAN region is shown in Fig. 5.1, which also shows the direction of the spatial sweep progression versus the direction of particle travel.

Within the DXTRAN region, all non-absorptive collisions are treated as analog because this work considers only a single DXTRAN region. Note that if multiple DXTRAN regions are considered, non-absorptive collisions in one DXTRAN region are capable of initiating the DXTRAN process for other DXTRAN regions that the collision is external to. Outside the DXTRAN region there are now cell-centered DXTRAN-biased scattering and inhomogeneous sources from Terms ② and ④, respectively.

The numeric solution algorithms for $\Psi_1$, $Q_2$, and $\Psi_2$ are described next.

### 5.2 Algorithms to Solve the DXTRAN HSMEs and FTE

Because $\Psi_1$ is a component of $Q_2$, $\widehat{\Psi}_1$ must be solved before $\widehat{Q}_2$ and thus, $\widehat{\Psi}_2$. Using scattering source iteration described in Algorithm 2.1, there are two ways to accomplish this. First, one can iteratively solve for $\widehat{\Psi}_1$, $\widehat{Q}_2$, and $\widehat{\Psi}_2$ in sequence within the same scattering source iteration. Alternatively, one may first solve for a converged $\widehat{\Psi}_1$, use the converged solution to compute $\widehat{Q}_2$, and then perform scattering source iterations to solve for $\widehat{\Psi}_2$. The latter approach is taken in this work, consistent with prior work [58].

Initially this work took the separated-moment approach by applying DXTRAN variance reduction while solving for $\widehat{\Psi}_1$ because it directly represents the Monte Carlo code behavior. That is, when the DXTRAN particle is generated it effectively splits the initiating particle by representing the uncollided component that reaches the DXTRAN region [Section 2.4.6.1]. However, this approach (a) requires extensively modifying the streaming operator, and thus the spatial mesh sweeper, to perform the truncation described by $\mathcal{T}_{\text{DX}}$ in Section 3.3b and (b) poses significant bookkeeping challenges when DXTRAN is combined with another variance-reduction technique. Thus, this work then applied the combined-moment approach where DXTRAN truncation is treated implicitly, so only the solution of $\widehat{Q}_2$, $\widehat{\Psi}_2$, and $\widehat{\Upsilon}$ are modified to account for DXTRAN.

The separated-moment approach is described next to solve $\widehat{\Psi}_1$ to demonstrate the significant modification to the spatial mesh sweeper and associated considerations. The transition from it to the combined-moment approach is then described. Using the combined-moment approach, the solution algorithms for $\widehat{Q}_2$ and $\widehat{\Psi}_2$ are then given. Because $\widehat{\Upsilon}$ is effectively a first-moment ordered quantity with a solution method comparable to $\widehat{\Psi}_1$, the solution of the HSMEs are focused on.

As noted previously, a simple example problem is used to motivate the upcoming discussion. That problem can be described as a 1-D, mono-energetic, $S_2$ calculation with a homogeneously distributed and isotropically scattering material in $0 \leq x \leq X = 10$ cm. A forward isotropic surface source is placed at the left-hand boundary at $x = 0$, a leakage current tally is placed at the right hand boundary at $x = 10$ cm, and a DXTRAN region occupies the right-hand portion of the problem. The DXTRAN region is defined with left-hand boundary $x_{\text{DX}} = 8$ cm. For the upcoming discussion, one should assume that the probability of free-flight from each cell-edge and cell-center position, along each $S_N$ ray to $x_{\text{DX}}$, $p_{\text{ff},i,i_{\text{DX}},n}$, is pre-computed using Eq. (2.89). To supplement this discussion, Appendix F has examples of Python-based solvers that can calculate $\widehat{M}_1$ and $\widehat{M}_2$ for this scenario, which corresponds to Test Case 1-2 described later in Section 6.2.2.

#### 5.2.1 Separated-moment Approach to Calculate $\widehat{\Psi}_1$

The separated-moment approach described here is supplemented by Listing F.1.

#### 5.2.1.1 First Iteration, $u = 1$, Left-to-right Sweep with $\mu_{n=1} < 0$

To apply the separated-moment approach, one can begin by setting the iteration index $u = 1$, setting the initial guesses for $\widehat{M_1}$ to one (i.e., $\widehat{M}_{1,\text{old}} = 1$) and $\Psi_1$ to zero, and starting with $\mu_{n=1} = -\sqrt{1/3}$, known boundary condition $\widehat{\Psi}_{1,u=1,i=1/2,n=1} = 0$, and initially guessed $\widehat{\Phi}_{1,u=0,i=1,r=1} = 0 \implies \widehat{q}_{u=0,i=1,n=1} = 0$, the cell-center moment for the left-most spatial cell is solved with Eq. (2.51) as

$$\widehat{\Psi}_{1,1,1,1} = \frac{\frac{2|\mu_1|}{\Delta x_1}\widehat{\Psi}_{1,1,1/2,1}^{\,0} + \widehat{q}_{0,1,1}^{\,0}}{\frac{2|\mu_1|}{\Delta x_1} + \Sigma_{\text{t},1}} = 0. \tag{5.12}$$

The cell-edge outgoing moment is then solved using Eq. (2.52) as

$$\widehat{\Psi}_{1,1,3/2,1} = 2\widehat{\Psi}_{1,1,1,1} - \widehat{\Psi}_{1,1,1/2,1} = 0, \tag{5.13}$$

and so on for each cell $i = 2, 3, \ldots I$. Because the incoming cell-edge values, inhomogeneous source, and the initial scattering sources are guessed as zero, the moment for all cells for $n = 1$ for this iteration is uniformly zero. Because the sweeper is proceeding from left to right, it exits the problem at the same time that it exits the DXTRAN region, so there is no modification necessary because of DXTRAN.

#### 5.2.1.2 First Iteration, $u = 1$, Right-to-left Sweep with $\mu_{n=2} > 0$

Next, one can consider $\mu_2 = \sqrt{1/3}$ with known boundary condition $\widehat{\Psi}_{1,1,I+1/2,2} = 1$ (as an arbitrary weight normalization) and with initially guessed $\widehat{\Phi}_{1,u=0,i=I,r=1} = 0 \implies \widehat{q}_{u=0,i=I,n=2} = 0$. The adjacent cell-center moment in the right-most spatial cell is again solved with Eq. (2.51) as

$$\widehat{\Psi}_{1,1,I,2} = \frac{\frac{2|\mu_2|}{\Delta x_I}\widehat{\Psi}_{1,1,I+1/2,2} + \widehat{q}_{0,I,1}^{\,0}}{\frac{2|\mu_2|}{\Delta x_I} + \Sigma_{\text{t},I}}, \tag{5.14}$$

which is now non-zero because of the boundary condition. Thus, the cell-outgoing angular moment is solved with Eq. (2.52) as

$$\widehat{\Psi}_{1,1,I-1/2,2} = 2\widehat{\Psi}_{1,1,I,2} - \widehat{\Psi}_{1,1,I+1/2,2}, \tag{5.15}$$

which is positive if the spatial mesh is defined so the cell is not too optically thick (see [88, page 132] for more information). Therefore, $\widehat{\Psi}_{1,1,I-1/2,2}$ becomes the cell-incoming moment for cell $I - 1$ such that Eq. (2.51) is solved for the corresponding cell-center moment, used with Eq. (2.52) to solve for the next cell-edge outgoing angular moment, and so on.

However, as the sweep exits the DXTRAN region at $x_{i_{\text{DX}}}$ the moment must be separated into a swept and analytic component corresponding to the initiating and DXTRAN particles, respectively. The swept component of the moment is truncated to zero corresponding to the initiating particle being killed upon entry into the DXTRAN region because as the sweep proceeds from right to left it represents particles directed from left-to-right, that is, into the DXTRAN region. Thus, the incoming cell-edge moment at $x_{i_{\text{DX}}}$, $\widehat{\Psi}_{1,1,i_{\text{DX}},2}$, is stored for later use to represent $\widehat{\Psi}_1\left(x_{\text{DX}}, \mu > 0\right)$ and then set to zero. Note that setting the moment to zero can conflict with a zero-moment fix-up scheme, if present. The sweep then proceeds, but because the incoming cell-edge moment is zero and no internal scattering sources have been computed (and no homogeneous sources

exist), the swept moment is zero for the remaining cells.

Only the swept moment has been focused on. The accumulation of scattering source within each spatial cell is now described. Following each sweep through each cell, the angular moment of the statistical moment in that cell is computed using Eq. (2.70c), which for this 1-D isotropic case with $R_{r=1}(\mu_n) = P_{\ell=0}(\mu_n) = 1$ is

$$\widehat{\Phi}_{1,1,i,r=1} = \sum_{n=1}^{2} \omega_n \widehat{\Psi}_{1,1,i,n}. \tag{5.16}$$

For each of the cells in the sweep that moves from left to right (with $n = 1$), the angular moment of the statistical moment, $\widehat{\Phi}_1$, remains zero because the angular statistical moment, $\widehat{\Psi}_1$, is zero, For the sweep in the opposite direction with $n = 2$, the accumulation differs inside and outside the DXTRAN region. Within the DXTRAN region, Eq. (5.16) is used and is non-zero because the swept moment is non-zero. Outside the DXTRAN region, the swept moment is zero but the analytic moment corresponding to the DXTRAN particle is

$$\widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n} = p_{\mathrm{ff},i,i_{\mathrm{DX}},n} \widehat{\Psi}_{1,1,i_{\mathrm{DX}},n}. \tag{5.17}$$

One should recognize that $p_{\mathrm{ff},i,i_{\mathrm{DX}},n} \neq 0$ when $\mu_n$ is directed toward the DXTRAN region and zero otherwise. Thus, outside the DXTRAN region the angular moment of the statistical moment is accumulated as

$$\widehat{\Phi}_{1,1,i,r=1} = \sum_{n=1}^{2} \omega_n \left( \widehat{\Psi}_{1,1,i,n} + \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n} \right). \tag{5.18}$$

As a result, the swept moment is zero but the analytic moment is non-zero, so the angular moment of the statistical moment is accumulated as non-zero to drive the scattering source on the subsequent iteration. Outside the DXTRAN region, two angular moments corresponding to the swept and analytic components of the moment are necessary for calculating $\widehat{Q}_2$ later and must be calculated as

$$\widehat{\Phi}_{\text{1-a},1,i,r=1} = \sum_{n=1}^{2} \omega_n \widehat{\Psi}_{1,1,i,n}, \tag{5.19}$$

$$\widehat{\Phi}_{\text{1-b},1,i,r=1} = \sum_{n=1}^{2} \omega_n \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n}, \tag{5.20}$$

respectively.

Having computed $\widehat{\Phi}_{1,1,i,r=1}$, $\widehat{\Phi}_{\text{1-a},1,i,r=1}$, and $\widehat{\Phi}_{\text{1-b},1,i,r=1}$ in all spatial cells, one can compute the new value of $\widehat{M}_1$ by recombining the separated moment and using Eq. (3.75) as

$$\begin{aligned} \widehat{M}_{1,\mathrm{new}} &= \int \mathrm{d}\boldsymbol{P} \widehat{Q}(\boldsymbol{P}) \widehat{\Psi}_1(\boldsymbol{P}) \\ &\approx \sum_i \sum_n \sum_w \widehat{Q}_{i,n,w} \omega_n \left( \widehat{\Psi}_{1,1,i,n} + \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n} \right) \\ &= \sum_i \sum_n \sum_w \delta_{i,1/2} \delta_{n,2} \delta_{w,1} \omega_n \left( \widehat{\Psi}_{1,1,i,n} + \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n} \right), \end{aligned} \tag{5.21}$$

where $\widehat{Q}_{i,n} = \delta_{i,1/2} \delta_{n,2} \delta_{w,1}$ is the discretized forward source term. The source term is a product of three

Kronecker delta functions at $i = 1/2$, $n = 2$, and $w = 1$ consistent with a forward source that exists only at the system left-hand boundary in the incident direction emitting particles with a weight of one. Finally, one can compute the relative change in the response as

$$\epsilon_{\text{rel}} = \frac{\left|\widehat{M}_{1,\text{new}} - \widehat{M}_{1,\text{old}}\right|}{\widehat{M}_{1,\text{new}}} \tag{5.22}$$

and compare $\epsilon_{\text{rel}}$ to the user-defined convergence criterion, $\epsilon$. If $\epsilon_{\text{rel}} < \epsilon$, then the calculation of $\widehat{M}_1$ is converged, iterations cease, and the calculation moves to calculating $\widehat{Q}_2$. If not, then one must update $\widehat{M}_{1,\text{old}} = \widehat{M}_{1,\text{new}}$ and iterate.

#### 5.2.1.3 Second Iteration, $u = 2$, Left-to-right Sweep with $\mu_{n=1} < 0$

Iterations proceed by incrementing $u = u + 1$ and computing the new cell-center source for all spatial cells using Eq. (5.7) as

$$\widehat{q}_{u=1,i,n} = \Sigma_{\text{s},0}\widehat{\Phi}_{1,u=1,i,r=1}. \tag{5.23}$$

This cell-center source is non-zero in all cells. It is non-zero outside the DXTRAN region because of the analytic component of the separated moment and it is non-zero inside because of the swept component. Thus, one can solve starting again at the left boundary for the left-most cell-center moment as the cell-center moment for the left-most spatial cell is solved with Eq. (2.51) as

$$\widehat{\Psi}_{1,2,1,1} = \frac{\frac{2|\mu_1|}{\Delta x_1}\cancelto{0}{\widehat{\Psi}_{1,1,1/2,1}} + \widehat{q}_{1,1,1}}{\frac{2|\mu_1|}{\Delta x_1} + \Sigma_{\text{t},1}}, \tag{5.24}$$

which is now non-zero because of the scattering source. The cell-center moment is calculated with

$$\widehat{\Psi}_{1,2,3/2,1} = 2\widehat{\Psi}_{1,2,1,1} - \widehat{\Psi}_{1,2,1/2,1} \tag{5.25}$$

and the sweep continues for all subsequent cells moving from left to right. Along the way, the angular moment of the statistical moment is accumulated with Eq. (5.18) (with $\widehat{\Phi}_{\text{1-a},2,i,r=1} \neq 0$ and $\widehat{\Phi}_{\text{1-b},2,i,r=1} = 0$ because $p_{\text{ff},i,i_{\text{DX}},n} = 0$).

#### 5.2.1.4 Second Iteration, $u = 2$, Right-to-left Sweep with $\mu_{n=2} > 0$

When the right-hand boundary is reached, the sweep restarts in the opposite direction as before. Similarly, when the sweep passes out of the DXTRAN region, the new DXTRAN-edge moment value is updated, stored, and the swept moment is again set to zero to compute the cell-center moment. Because there is now scattering source in the mesh cell adjacent to and outside the DXTRAN region, a non-zero swept moment builds in from scattering. As such, the sweep continues with non-zero moment driven by scattering source and streaming. The cell-center angular moment of the statistical moment is then accumulated from both swept and analytic components as described by Eq. (5.18). The sweep continues until the left-hand boundary is reached. The first moment response, $\widehat{M}_{1,\text{new}}$, is recomputed, $\epsilon_{\text{rel}}$ is recomputed, and and so on until convergence is achieved.

### 5.2.2 Combined-moment Approach to Calculate $\Psi_1$

The combined-moment approach described here is supplemented by Listing F.2. Because this approach is also used for the remainder of this work, it is summarized at the conclusion of this section in Algorithm 5.1.

#### 5.2.2.1 First Iteration, $u = 1$, Left-to-right Sweep with $\mu_{n=1} < 0$

The process to sweep from left to right for the first iteration using the combined-moment approach proceeds identically to Section 5.2.1.1 and is not repeated. To summarize: the angular statistical moment and accumulated angular moment of the statistical moment are both zero at the conclusion of this sweep.

#### 5.2.2.2 First Iteration, $u = 1$, Right-to-left Sweep with $\mu_{n=2} > 0$

The sweep from right to left from the right-hand boundary is the same as in Section 5.2.1.2 until the DXTRAN boundary is crossed at $x_{i_{\mathrm{DX}}}$. When crossed, the edge moment is stored but *not* truncated to zero. However, the angular moment of the statistical moment is accumulated as

$$\widehat{\Phi}_{1,1,i,r=1} = \sum_{n=1}^{2} \omega_n \left( \widehat{\Psi}_{1,1,i,n} - \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n} \right) \tag{5.26}$$

and the two angular moments of the statistical moment needed to calculate $\widehat{Q}_2$ are accumulated as

$$\widehat{\Phi}_{\text{1-a},1,i,r=1} = \sum_{n=1}^{2} \omega_n \left( \widehat{\Psi}_{1,1,i,n} - \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n} \right), \tag{5.27a}$$

$$\widehat{\Phi}_{\text{1-b},1,i,r=1} = \sum_{n=1}^{2} \omega_n \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n}, \tag{5.27b}$$

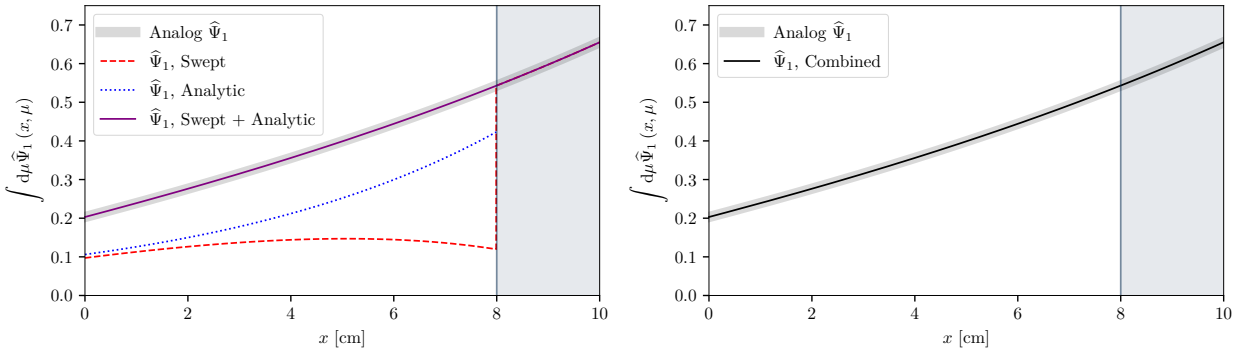
where $\widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n}$ is defined by Eq. (5.17). By subtracting $\widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n}$ there is an implicit truncation of the swept-moment terms and the sweeper does not need to be modified. This introduces the additional benefit that any zero-moment fix-up scheme [Section 2.3.2] does not need to be adjusted. Otherwise, this sweep proceeds as in Section 5.2.1.2. At the conclusion, $\widehat{M}_{1,\mathrm{new}}$ is evaluated as

$$\begin{aligned} \widehat{M}_{1,\mathrm{new}} &= \int \mathrm{d}\boldsymbol{P} \widehat{Q}\left(\boldsymbol{P}\right) \widehat{\Psi}_1\left(\boldsymbol{P}\right) \\ &\approx \sum_i \sum_n \sum_w \widehat{Q}_{i,n,w} \omega_n \widehat{\Psi}_{1,1,i,n} \\ &= \sum_i \sum_n \sum_w \delta_{i,1/2} \delta_{n,2} \delta_{w,1} \omega_n \widehat{\Psi}_{1,1,i,n}, \end{aligned} \tag{5.28}$$

convergence is assessed, and iterations continue, if needed. Subsequent iterations proceed similarly without any moment separation or truncation except to accumulate and store $\widehat{\Phi}_{\text{1-a},1,i,r=1}$ and $\widehat{\Phi}_{\text{1-b},1,i,r=1}$ for $\widehat{Q}_2$. In this way, the solution of $\widehat{\Psi}_1$ and $\widehat{\Phi}_1$ proceeds identically to the analog case, as expected, as shown in Algorithm 5.1.

**Algorithm 5.1:** Combined-moment Approach to Calculate $\widehat{\Psi}_1$ with DXTRAN

$\Sigma_{\mathrm{s},r} \leftarrow$ Eq. (2.66) for all mesh cells; // Material scattering moments
*converged* $\leftarrow$ False; $u \leftarrow 0$; $\widehat{\Psi}_1 \leftarrow 0$; $\widehat{\Phi}_1 \leftarrow 0$; $\widehat{M}_{1,\mathrm{old}} \leftarrow 0$; $\widehat{M}_{1,\mathrm{new}} \leftarrow 0$
$\widehat{Q}_{\mathrm{s},g} \leftarrow 0$; // Scattering source
$\widehat{\Phi}_{1\text{-a}} \leftarrow 0$; $\widehat{\Phi}_{1\text{-b}} \leftarrow 0$ // For $\widehat{Q}_2$
**while not** converged **do**
  $u \leftarrow u + 1$
  **forall** Energy groups **do**
    // Recompute scattering sources using latest angular moments.
    $\widehat{Q}_{\mathrm{s}} \leftarrow$ Eq. (2.65); // Scattering source
    // Perform scattering source iteration sweep for all directions and cells.
    **forall** $S_N$ directions **do**
      **forall** $S_N$ spatial cells **do**
        **if** $\boldsymbol{\Omega}_n \cdot \boldsymbol{n} > 0$ **and** on problem boundary **then**
          Set incoming cell-edge value to boundary source
        **else**
          Set incoming cell-edge value to previous cell's outgoing cell-edge value
        // Compute cell-center and cell-edge outgoing quantities.
        $\widehat{q} \leftarrow$ Eq. (2.44); // Total cell source
        $\widehat{\Psi}_1 \leftarrow$ Eq. (2.45); // Cell-center $\widehat{\Psi}_1$
        $\widehat{\Phi}_{1,r} \leftarrow$ Eq. (2.70c); // Update cell-center $\widehat{\Psi}_1$ angular moment
        // Update local and DXTRAN-edge angular moments for $\widehat{Q}_2$.
        $\widehat{\Phi}_{1\text{-a}} \leftarrow \widehat{\Psi}_1 - p_{\mathrm{ff}}\widehat{\Psi}_{1,\mathrm{DX}}$; // Local $\widehat{\Psi}_1$ angular moment
        $\widehat{\Phi}_{1\text{-b}} \leftarrow p_{\mathrm{ff}}\widehat{\Psi}_{1,\mathrm{DX}}$; // DXTRAN-edge $\widehat{\Psi}_1$ angular moment
        Calculate outgoing cell-edge value with Eq. (2.46)
        Advance to next cell along $-\boldsymbol{\Omega}_n$
  // Update DXTRAN-edge values for mesh outside DXTRAN region.
  **forall** $n$ **do**
    **forall** $i, j, k$ **do**
      **if** Outside DXTRAN Region **and** Directed Toward DXTRAN Region **then**
        Store $\Psi_{1,\mathrm{DX}} \leftarrow$ DXTRAN-edge moment value
  // Compute response relative convergence.
  $\widehat{M}_{1,\mathrm{new}} \leftarrow$ Eq. (3.75)
  **if** $\left|\widehat{M}_{1,\mathrm{new}} - \widehat{M}_{1,\mathrm{old}}\right| / \widehat{M}_{1,\mathrm{new}} < \epsilon$ **then**
    *converged* $\leftarrow$ True
  $\widehat{M}_{1,\mathrm{old}} \leftarrow \widehat{M}_{1,\mathrm{new}}$




**(a)** Separated $\widehat{\Psi}_1$ **(b)** Combined $\widehat{\Psi}_1$

**Figure 5.2:** Separated and Combined DXTRAN $\widehat{\Psi}_1$ versus Analog $\widehat{\Psi}_1$ for a Homogeneous 20% Scatterer, $\Sigma_\mathrm{t} = 0.1$ cm$^{-1}$ with a DXTRAN region in $8 \leq x \leq 10$ cm

### 5.2.3 Summary of the Separated- and Combined-moment Approaches

As noted at the beginning of Section 5.2, when this work began a separated-moment approach was used to directly represent MCNP DXTRAN behavior in the deterministic solver. That is, the HSME solutions are separated into swept and analytic components. The swept component represents the free-flight of particles following non-absorptive collision emergence. Such particles are killed if they attempt to enter the DXTRAN region during the Monte Carlo calculation. For the adjoint-like moment quantities being solved by COVRT, this means that as the sweeper exits the DXTRAN region the swept moment must be truncated to zero. Truncation happens upon exit because the sweep travels opposite the direction of the particle for adjoint-like quantities, so as the sweeper exits the DXTRAN region it represents the backwards streaming of particles facing/entering the DXTRAN region. Correspondingly, the analytic component represents the DXTRAN particles created through collision-induced DXTRAN, which have their weight adjusted by an analytic calculation [Eq. (2.93)].

Figure 5.2 shows $\widehat{\Psi}_1$ obtained using the separated- and combined-moment approaches. As expected, when the analytic and swept components are summed for Fig. 5.2a the analog $\widehat{\Psi}_1$ is recovered. The converged swept $\widehat{\Psi}_1$ is not zero outside the DXTRAN region because the analytic term generates scattering sources external to the DXTRAN region that are carried along by the sweeper in subsequent iterations. Regardless, the effect of truncating the moment at the DXTRAN edge is apparent. This treatment faithfully reproduces the behavior of Monte Carlo DXTRAN generation and truncation in the deterministic solver; however, the separation necessitates additional information storage and bookkeeping and complicates the use of DXTRAN with other variance-reduction techniques such as importance splitting and rouletting. This complication arises because $Q_2$ and $\Psi_2$ for other variance reduction techniques require the combined $\Psi_1$ and $\Psi_2$, respectively.

As this work progressed, it became apparent that the combined-moment approach is available that does not require (a) modification to the sweeper to truncate moment to zero upon exiting the DXTRAN region and (b) separately tracking the swept and analytic components of the moments, as described in Section 5.2.2. While the combined-moment approach does not directly represent the Monte Carlo DXTRAN behavior, it

has the advantages that it

1. Reduces the information that needs to be stored (the analytic component is not stored separately),
2. Eliminates the bookkeeping to store and apply the swept and analytic statistical moments to calculate $\widehat{M_1}$ and $\widehat{M_2}$, and
3. Eases incorporating other variance-reduction techniques with DXTRAN.

The combined-moment approach treats DXTRAN truncation implicitly by subtracting away free-flight contribution components when calculating scattering contributions in $\widehat{Q}_2$ and $\widehat{\Psi}_2$. This has the effect of treating $\widehat{\Psi}_1$ and $\widehat{\Upsilon}$ as analog (shown in Fig. 5.2b for $\widehat{\Psi}_1$), which is consistent with how COVRT treats $\widehat{\Psi}_1$ and $\widehat{\Upsilon}$ for all other variance-reduction techniques. Because truncation is treated implicitly for $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$, the effect of DXTRAN on $\widehat{Q}_2$ and $\widehat{\Psi}_2$ is easier to reconcile with other variance-reduction techniques. For example, importance splitting that occurs at Monte Carlo cell surfaces depends on $\widehat{\Psi}_1$ to compute $\widehat{Q}_2$ so $\widehat{\Psi}_1$ needs to be reconstructed if separated into swept and analytic components. In addition, because $\widehat{\Psi}_2$ is not separated it is straightforward to adjust it based on the effect of other variance-reduction techniques.

### 5.2.4 Combined-moment Approach to Calculate $\widehat{Q}_2$

As shown in Eq. (5.8), $Q_2$ requires calculating the product of two collision emergence integrals as

$$2 \times \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_1\left(\boldsymbol{P}_3\right) \times \int \mathrm{d}\Omega_4 \int \mathrm{d}E_4 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_4\right) p_{\mathrm{ff}}\left(\boldsymbol{x}_2, \boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4\right) \Psi_1\left(\boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4, E_4, w_2\right).$$

Physically, in a forward sense, the first term represents those particles that emerge from non-absorptive collision that do not reach the DXTRAN region on their next free flight while the second term represents those particles that emerge from non-absorptive collision that *do* reach the DXTRAN region on their next free flight (which are killed and represented instead by DXTRAN particles). These two terms correspond to the terms calculated with Eqs. (5.27a) and (5.27b), respectively, while calculating $\widehat{\Psi}_1$.

Because these are scattering emergence terms, they can be recast into an adjoint integro-differential scattering-like source for $\Psi_2$ as

$$\begin{aligned} Q_2\left(\boldsymbol{x}, \boldsymbol{\Omega}\right) = 2 \times & \int \mathrm{d}\boldsymbol{\Omega}' \int \mathrm{d}E' \Sigma_s\left(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E \rightarrow E'\right) \Psi_1\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right) \\ & \times \int \mathrm{d}\boldsymbol{\Omega}' \int \mathrm{d}E' \Sigma_s\left(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E \rightarrow E'\right) p_{\mathrm{ff}}\left(\boldsymbol{x}_2, \boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4\right) \Psi_1\left(\boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}', E'\right), \end{aligned} \quad (5.29)$$

which becomes

$$\begin{aligned} Q_2\left(\boldsymbol{x}, \boldsymbol{\Omega}\right) = 2\Sigma_{s,0}\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right) \times & \int \mathrm{d}\boldsymbol{\Omega}' \int \mathrm{d}E' p\left(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E \rightarrow E'\right) \Psi_1\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right) \\ & \times \int \mathrm{d}\boldsymbol{\Omega}' \int \mathrm{d}E' p\left(\boldsymbol{x}, \boldsymbol{\Omega} \cdot \boldsymbol{\Omega}', E \rightarrow E'\right) p_{\mathrm{ff}}\left(\boldsymbol{x}_2, \boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}_4\right) \Psi_1\left(\boldsymbol{x}_{\mathrm{DX}}, \boldsymbol{\Omega}', E'\right) \end{aligned} \quad (5.30)$$

by separating the magnitude of scattering from the analog double-differential direction-energy transition PDF. Thus, using the angular moment expansion given in Eq. (2.62), the angular moment expansion of the

statistical moment given in Eq. (2.67b) applied in Eq. (5.27) with the simple motivating case, one obtains

$$\widehat{Q}_{2,\mathrm{DX},u,i,n} = 2\Sigma_{s,i,\ell=0}\widehat{\Phi}_{\text{1-a},u,i,r=1}\widehat{\Phi}_{\text{1-b},u,i,r=1}. \tag{5.31}$$

As implemented, the scattering cross-section is not separated from the analog double-differential direction-energy transition PDF. Instead, it is included in each integral (kept as individual summations) but then the result is normalized by the scalar scattering cross-section magnitude to prevent double-counting scattering. This permits treating higher-order scattering (left to be confirmed in future work). The implementation used in this work is shown in Algorithm 5.2.

**Algorithm 5.2:** Combined-moment Approach to Calculate $\widehat{Q}_2$ with DXTRAN

```
1  forall Energy groups do
2      Q̂_{2,DX} ← 0
3      forall S_N directions do
4          forall S_N spatial cells do
5              if Using DXTRAN and not Void then
6                  t_1 ← 0, t_2 ← 0;                          // Initialize temporary accumulators
7                  forall Angular Moments and Incoming Energy Groups do
8                      t_1 ← t_1 + Σ_{s,r} Φ̂_{1-a}
9                      t_2 ← t_2 + Σ_{s,r} Φ̂_{1-b}
10                 Q̂_{2,DX} ← Q̂_{2,DX} + 2 t_1 t_2 / Σ_{s,0}
```

### 5.2.5 Combined-moment Approach to Calculate $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$

The calculation of $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$ is similar to $\widehat{\Psi}_1$. The main change arises in how the angular moments of $\widehat{\Psi}_2$, $\widehat{\Phi}_2$, are computed. Term ① of Eq. (3.74a) requires the cell-center swept flux and Term ② requires the cell-center analytic flux. Thus,

$$\begin{aligned}
&\int \mathrm{d}\mu_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\boldsymbol{P}_3 \mathcal{E}\left(\boldsymbol{P}_2, \boldsymbol{P}_3\right) \Psi_2\left(\boldsymbol{P}_3\right) \\
&\quad + \int \mathrm{d}\mu_1 \int \mathrm{d}w_1 \mathcal{T}_{\mathrm{DX}}\left(\boldsymbol{P}_0, \boldsymbol{P}_1\right) \int \mathrm{d}\boldsymbol{P}_2 \mathcal{K}\left(\boldsymbol{P}_1, \boldsymbol{P}_2\right) \int \mathrm{d}\mu_1 \frac{p_{\mathrm{ff}}^2\left(x_2, x_{\mathrm{DX}}, \mu_4\right)}{\beta\left(x_2\right)} \frac{p^2\left(\mu_2 \rightarrow \mu_4\right)}{\tilde{p}\left(\mu_2 \rightarrow \mu_4\right)} \Psi_2\left(x_{\mathrm{DX}}, \mu_4, w_2\right)
\end{aligned}$$

is recast into the integro-differential form and discretized to become

$$\widehat{\Phi}_{2,1,i,r=1} = \sum_{n=1}^{2} \omega_n \left(\widehat{\Psi}_{1,1,i,n} - \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n}\right) + \sum_{n=1}^{2} \omega_{n'} \frac{\left(p_{\mathrm{ff},i,i_{\mathrm{DX}},n}\right)^2}{\beta_i} \frac{p^2\left(\mu_n\right)}{\widetilde{p}\left(\mu_n\right)} \widehat{\Psi}^{\mathrm{DX}}_{1,1,i,n}, \tag{5.32}$$

where $p\left(\mu_n\right) = 1/2$ for isotropic scattering and $\widetilde{p}\left(\mu_n\right) = 1$ as long as $\mu_n$ is pointed toward the DXTRAN region. The DXTRAN rouletting parameter, $\beta_i$, is assumed to be one for the demonstration case.

Meanwhile, $\widehat{\Upsilon}$ (as a first-moment ordered quantity) is computed similar to $\widehat{\Psi}_1$. The main modification is to incorporate the average time contribution functions ($\langle p_{\mathrm{xs}}\left(\boldsymbol{P}, \tau_{\mathrm{xs}}\right)\rangle$, $\langle p_{\mathrm{col}}\left(\boldsymbol{P}, \tau_{\mathrm{col}}\right)\rangle$, $\langle p_{\mathrm{DX}}\left(\boldsymbol{P}, \tau_{\mathrm{DX}}\right)\rangle$, $\langle p_{\mathrm{ff}}\left(\boldsymbol{P}_4, \tau_{\mathrm{ff}}\right)\rangle$, etc.) as additional source terms associated with surface crossings, non-absorptive collisions, etc. For surface-crossing terms, the time contribution functions are added to the incoming cell-edge angular future time. For non-absorptive collision terms, the time contribution functions are added to the cell-center

angular future time to compute the angular moment future time used to compute the subsequent cell-center scattering source. However, for the non-absorptive collision terms the same implicit truncation is performed as in the case of $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$.

The combined-moment approach used in this work to compute $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$ is shown in Algorithm 5.3.

**Algorithm 5.3:** Combined-moment Approach to Calculate $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$ with DXTRAN

*converged* $\leftarrow$ False; $u \leftarrow 0$; $\widehat{\Psi}_2 \leftarrow 0$; $\widehat{M}_{2,\text{old}} \leftarrow 0$; $\widehat{M}_{2,\text{new}} \leftarrow 0$; $\widehat{\Upsilon} \leftarrow 0$; $\widehat{\Upsilon}_{\text{old}} \leftarrow 0$; $\widehat{\Upsilon}_{\text{new}} \leftarrow 0$
$\widehat{Q}_{\Psi_2,\text{s},g} \leftarrow 0$; $\widehat{Q}_{\Upsilon,\text{s},g} \leftarrow 0$; // $\widehat{\Psi}_2$ and $\Upsilon$ scattering sources
**while not** converged **do**
  $u \leftarrow u + 1$
  **forall** Energy groups **do**
    // Recompute scattering sources using latest angular moments.
    $\widehat{Q}_{\Psi_2,\text{s},g} \leftarrow$ Eq. (2.65); $\widehat{Q}_{\Upsilon,\text{s},g} \leftarrow$ Eq. (2.65);
    // Perform scattering source iteration sweep for all directions and cells.
    **forall** $S_N$ directions **do**
      **forall** $S_N$ spatial cells **do**
        **if** $\boldsymbol{\Omega}_n \cdot \boldsymbol{n} > 0$ **and** on problem boundary **then**
          Set incoming cell-edge value to boundary source
        **else**
          Set incoming cell-edge value to previous cell's outgoing cell-edge value
        // Compute cell-center and cell-edge outgoing quantities.
        $\widehat{q}_{\Psi_2} \leftarrow$ Eq. (2.44); $\widehat{q}_{\Upsilon} \leftarrow$ Eq. (2.44); // Total cell $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$ source
        $\widehat{\Psi}_2 \leftarrow$ Eq. (3.74a); $\widehat{\Upsilon} \leftarrow$ Eq. (4.31); // Cell-center $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$
        // Update cell-center $\widehat{\Psi}_2$ and $\Upsilon$ angular moments
        **if** Using DXTRAN **and** Outside DXTRAN Region **then**
          $\widehat{\Phi}_{2,r} \leftarrow$ Eq. (3.69);
        **else**
          $\widehat{\Phi}_{2,r} \leftarrow$ Eq. (2.70c);
        $\widehat{\Upsilon}_r \leftarrow$ Eq. (2.70c)
        Calculate outgoing cell-edge values with Eq. (2.46)
        Advance to next cell along $-\boldsymbol{\Omega}_n$
  // Update DXTRAN-edge values for mesh outside DXTRAN region.
  **forall** $n$ **do**
    **forall** $i, j, k$ **do**
      **if** Outside DXTRAN Region **and** Directed Toward DXTRAN Region **then**
        Store $\widehat{\Psi}_{2,\text{DX}} \leftarrow$ DXTRAN-edge moment value
  // Compute response relative convergence.
  $\widehat{M}_{2,\text{old}} \leftarrow \widehat{M}_{2,\text{new}}$; $\widehat{M}_{2,\text{new}} \leftarrow$ Eq. (3.75)
  $\widehat{\Upsilon}_{\text{old}} \leftarrow \widehat{\Upsilon}_{\text{new}}$; $\widehat{\Upsilon}_{\text{new}} \leftarrow$ Eq. (4.2)
  **if** $\left|\widehat{M}_{2,\text{new}} - \widehat{M}_{2,\text{old}}\right| / \widehat{M}_{2,\text{new}} < \epsilon$ **and** $\left|\widehat{\Upsilon}_{\text{new}} - \widehat{\Upsilon}_{\text{old}}\right| / \widehat{\Upsilon}_{\text{new}} < \epsilon$ **then**
    *converged* $\leftarrow$ True

### 5.3 Modifications to COVRT to Incorporate DXTRAN

This section provides a review of the changes to COVRT to incorporate DXTRAN variance reduction and follows with a description of the once-through COVRT program flow. COVRT is written in object-oriented

Fortran and was originally developed to solve the HSMEs and FTE for weight-window variance reduction but also supports importance splitting/rouletting, weight cutoffs, and implicit capture. COVRT computes these solutions iteratively using scattering source iterations with the multi-group and discrete ordinates approximations on a Cartesian spatial mesh as described previously. This discussion is provided to supplement prior work [58] for the benefit of future researchers that work with this tool.

### 5.3.1 Modifications to COVRT to Incorporate DXTRAN

The main modifications to COVRT for this work are to extend it to accommodate DXTRAN variance reduction. The changes required include:

1. Adding a DXTRAN object data structure to store DXTRAN region properties such as spatial extents and corresponding mesh indices,
2. Augmenting spatial mesh cell center and cell edges with the following pre-computed or periodically re-computed values:
   (a) An ID flag to indicate whether the cell center is inside or outside the DXTRAN region,
   (b) A flag to indicate whether the cell edge is on the DXTRAN region boundary,
   (c) $\beta_c$ as a user-assigned DXTRAN rouletting parameter assigned by the user to Monte Carlo cells and applied to all deterministic spatial mesh cells corresponding to the particular Monte Carlo spatial cell,
   (d) $p_{\text{ff}}(\mathbf{\Omega}_n, E_g)$ to indicate the free-flight probability from the cell center or cell edge position to the DXTRAN region along discrete ordinate direction $\mathbf{\Omega}_n$ in energy group $g$,
   (e) $\widetilde{p}(\mathbf{\Omega}_n)$ to indicate the value of the arbitrary DXTRAN emergence PDF along $\mathbf{\Omega}_n$,
   (f) $\widehat{\Psi}_{1,\text{edge}}(\mathbf{\Omega}_n, E_g)$ to indicate the DXTRAN region edge first moment value in energy group $g$ as viewed from the cell center or cell edge position along direction $\mathbf{\Omega}_n$,
   (g) $\widehat{\Psi}_{2,\text{edge}}(w_w, \mathbf{\Omega}_n, E_g)$ to indicate the DXTRAN region edge second moment value in statistical weight group $w$ and energy group $g$ as viewed from the cell center or cell edge position along direction $\mathbf{\Omega}_n$,
   (h) $\widehat{\Upsilon}_{\text{edge}}(w_w, \mathbf{\Omega}_n, E_g)$ to indicate the DXTRAN region edge future time value in statistical weight group $w$ and energy group $g$ as viewed from the cell center or cell edge position along direction $\mathbf{\Omega}_n$,
   (i) $n_{\text{surf.}}(\mathbf{\Omega}_n, E_g)$ to indicate the number of Monte Carlo geometry surfaces between the cell center or cell edge position and the DXTRAN region along direction $\mathbf{\Omega}_n$,
3. Adding a function to compute the optical distance between two positions as a part of calculating $p_{\text{ff}}(\mathbf{\Omega}_n, E_g)$,
4. Adding a function to compute the number of Monte Carlo geometry surfaces between two positions to assign $n_{\text{surf.}}(\mathbf{\Omega}_n, E_g)$,
5. Adding a subroutine to check whether the DXTRAN region has its boundaries defined collinear with a the spatial cell mesh edges, and if not, to resize the DXTRAN region to position its boundaries on the nearest edge,

6. Adding a subroutine to recompute $\widehat{\Psi}_{1,\text{edge}}$, $\widehat{\Psi}_{2,\text{edge}}$, and $\widehat{\Upsilon}_{\text{edge}}$ following a complete source iteration spatial sweep over all angles or after the DXTRAN region is moved or resized (e.g., following an optimization iteration),

7. Modifying the spatial sweep behavior when DXTRAN is applied (discussed in greater detail in Section 5.2.3),

8. Updating the calculation of $\widehat{Q}_2$ to accommodate DXTRAN, and

9. Introducing C-bindings that allows COVRT to be compiled into and called as a shared object library.

Other program execution and user efficiency enhancements are also made, but these modifications are not directly germane to this work so they are not discussed further.

### 5.3.2 Description of Once-through COVRT Program Flow

To perform a single COVRT calculation to predict the mean, population variance, future time, and ultimately the computational cost of a Monte Carlo calculation incorporating DXTRAN variance reduction, the once-through approach is most appropriate. With this approach, COVRT is compiled from its Fortran source code into a binary executable that takes a variety of command line arguments and an input file to define how it behaves. The command line arguments are meant to specify overall calculation parameters (global spatial mesh size, toggling between a weight-dependent or weight-independent calculation, enforcing a certain number of processing threads, etc.). The input file is similar in structure to an MCNP input file describing Monte Carlo surface type and position, surface-cell relationships, materials, forward source parameters, and tally parameters. The overall once-through program flow of COVRT is shown in Fig. 5.3. A brief description of the once-through program flow follows.

When COVRT executes, it first parses the command line to find execution parameters (most notably, the name of the input file). COVRT then performs two passes to read and cross-validate the input values to mitigate user error as summarized in Fig. 5.4. The same types of steps are used later to read and cross-validate the input file. Then, memory for the calculation is allocated and pointers to the allocated memory addresses are assigned to initialize the calculation (see Fig. 5.5).

The major data structure within COVRT is the mesh object, which contains all cell-center and cell-edge information (dimensions, material information pointers, HSME and FTE solution pointers, etc.). Using the memory space allocated during initialization, the mesh object is constructed as summarized by Algorithm 5.4. Construction of the mesh object is the second-most time consuming component of COVRT (second to actually solving for $\widehat{\Psi}_1$, $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$). In addition, because the mesh object is monolithic, most other program components depend on it and it takes a relatively long time to compile and link. The slowness during compiling and linking is exacerbated when compiler optimization is employed. Future work that will benefit both developers and users includes separating the mesh object into small program components.

With all memory allocated, the HSMEs and FTE are solved using the scattering source iteration approach described in Chapter 2 used as described in Sections 5.2.2, 5.2.4, and 5.2.5. The solutions to the HSMEs and FTE are then used to compute the computational cost (see Fig. 5.6). A summary of the source iteration

algorithm for $\widehat{\Psi}_1$, with special consideration for the effect of DXTRAN, is summarized in Algorithm 5.1. The computed $\widehat{\Psi}_1$ is then used to calculate $\widehat{Q}_2$ (summarized in Algorithm 5.2), which is then used to compute $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$ (summarized in Algorithm 5.3).

Additional discussion regarding the DXTRAN setup algorithms follow.

### 5.3.3 Algorithm 5.4 (Construct Mesh) and Algorithm 5.5 (Setup DXTRAN on Mesh)

For this work only a single DXTRAN region is considered. However, the DXTRAN and spatial mesh data structures in COVRT are designed so only a few changes in the future should be necessary to include multiple, potentially nested, DXTRAN regions. The $S_N$ mesh cell-center and cell-edge objects include integer-valued DXTRAN identifier lists. If the $S_N$ cell is within a DXTRAN region, the identifier list is set to contain the DXTRAN region IDs for all DXTRAN regions the $S_N$ cell is within. For this work, with only a single DXTRAN region, the ID is always one if the $S_N$ cell is within the DXTRAN region. If the $S_N$ cell is outside the DXTRAN region the ID is set to zero. A simple Boolean flag is also included to indicate whether the $S_N$ cell is within any DXTRAN regions. The same approach is used to indicate if an $S_N$ cell edge is also a boundary of one or more DXTRAN regions.

Otherwise, Algorithm 5.4 initializes DXTRAN-edge moment values for cell-center mesh to zero (these values are used to compute collision angular moments). Algorithm 5.5 initializes the values of all $S_N$ cell-center and cell-edge values external to the DXTRAN region called for by the DXTRAN biasing kernel in Eq. (3.52).

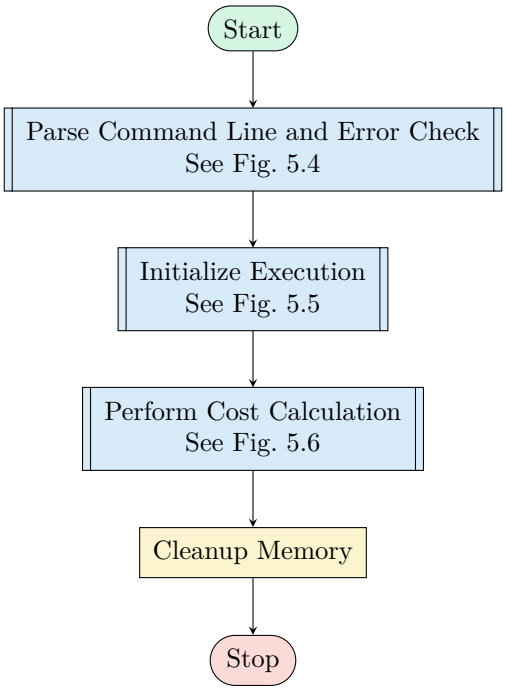



**Figure 5.3:** Overall Once-through COVRT Program Flow

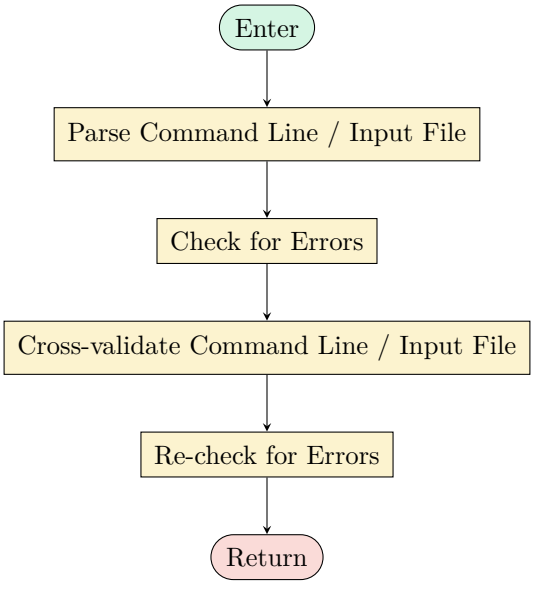



**Figure 5.4:** Command Line and Input File Parsing Process

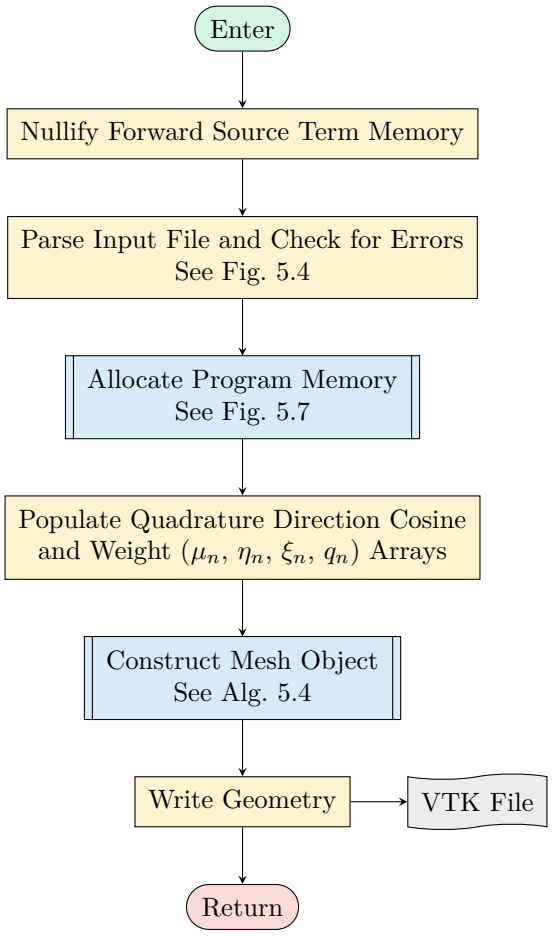



**Figure 5.5:** Initialize Calculation Memory

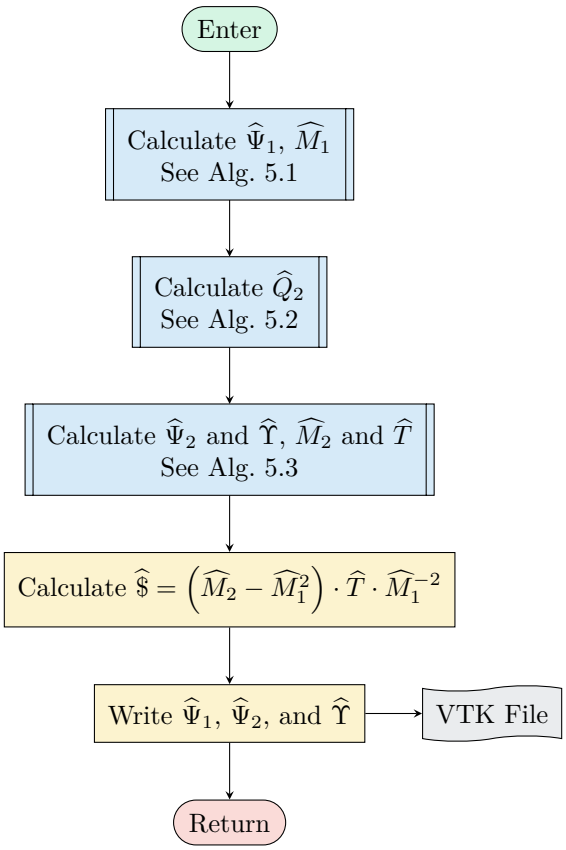



**Figure 5.6:** Compute Quantities of Interest

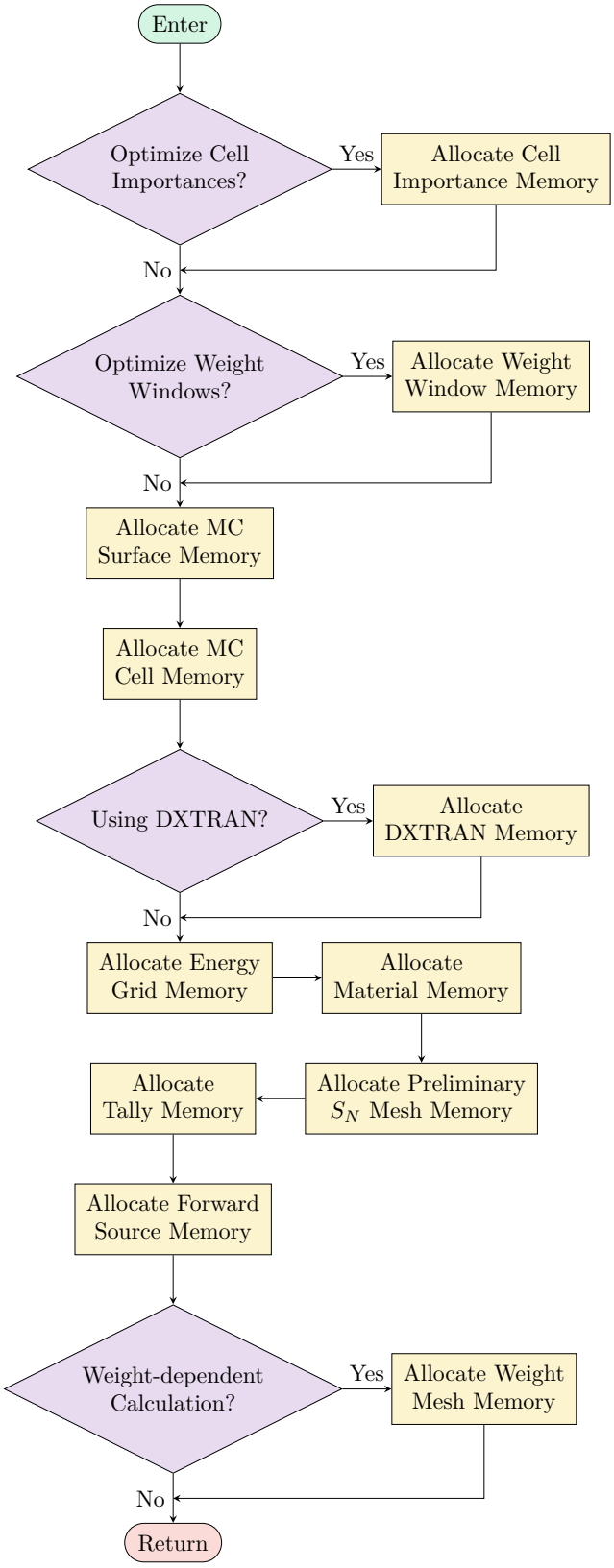



**Figure 5.7:** Allocate Program Memory

**Algorithm 5.4:** Construct Mesh with DXTRAN

Remove duplicate user-input Monte Carlo surfaces
**if** `sm` assigned on command line **then**
  $\mathrm{d}x$, $\mathrm{d}y$, $\mathrm{d}z \leftarrow$ `sm`
**else**
  `// Size mesh to avoid moment negativity.`
  $\mathrm{d}x \leftarrow \min(\mu_n) / \max(\Sigma_t)$; $\mathrm{d}y \leftarrow \min(\eta_n) / \max(\Sigma_t)$; $\mathrm{d}z \leftarrow \min(\xi_n) / \max(\Sigma_t)$
Fit Monte Carlo surfaces to nearest $S_N$ cell-edge nodes defined with $\mathrm{d}x$, $\mathrm{d}y$, $\mathrm{d}z$
Set number of cell-center nodes in $x$, $y$, and $z$ directions
**forall** $S_N$ spatial cells **do**
  Assign cell-center $x$, $y$, and $z$ coordinates
  Compute cell volume
  Assign corresponding Monte Carlo cell ID
  Assign corresponding $i$, $j$, and $k$ $S_N$ cell indices
  $\widehat{\Psi}_1 \leftarrow 0$; $\widehat{\Psi}_2 \leftarrow 0$; $\widehat{\Upsilon} \leftarrow 0$
  **if** material not void **then**
    `// Set material, flux, and source collisional moments to zero.`
    $\Sigma_{s,g'\rightarrow g,l} \leftarrow 0$; $\widehat{\Psi}_{1,g,r}$; $\widehat{\Psi}_{2,g,r}$; $\widehat{\Upsilon}_{g,l}$; $\widehat{Q}_{s,g} \leftarrow 0$
    **if** DXTRAN is being used and outside DXTRAN region **then**
      $\widehat{\Psi}_{1,\mathrm{DX}} \leftarrow 0$; $\widehat{\Psi}_{2,\mathrm{DX}} \leftarrow 0$; $\widehat{\Upsilon}_{\mathrm{DX}} \leftarrow 0$; `// DXTRAN-edge` $\widehat{\Psi}_1$, $\widehat{\Psi}_2$, `and` $\widehat{\Upsilon}$
      $\widehat{Q}_{\mathrm{DX\text{-}a},g,l} \leftarrow 0$; $\widehat{Q}_{\mathrm{DX\text{-}b},g,l} \leftarrow 0$ `// Set DXTRAN` $\widehat{Q}_2$ `source moments to zero`
  **if** Monte Carlo cell contains a track-length tally **then**
    Increment $S_N$ cell tally counter and allocate memory for $\widehat{Q}_1$ and $\widehat{Q}_2$
  **if** DXTRAN is being used **then**
    Allocate memory to store $\widehat{Q}_2$ from DXTRAN
  **if** $S_N$ cell contains a volumetric forward source **then**
    Allocate memory to store computed responses $\widehat{M}_1$, $\widehat{M}_2$, and $\widehat{T}$
  **if** Collisional weight windows are being used **then**
    Allocate memory to store $\widehat{Q}_2$ from weight windows
  **if** Implicit capture is being used **then**
    Allocate memory to store $\widehat{Q}_2$ from implicit capture
  **forall** $S_N$ mesh edges **do**
    Allocate memory to store edge node object
    **if** Importance splitting surface or surface-based weight windows **then**
      Allocate memory for associated $\widehat{Q}_2$ terms
    **if** Any bounding surface is a surface tally **then**
      Increment $S_N$ surface tally counter and allocate memory for $\widehat{Q}_1$ and $\widehat{Q}_2$
    **if** Any bounding surface is a surface source **then**
      Allocate memory to store computed responses $\widehat{M}_1$, $\widehat{M}_2$, and $\widehat{T}$
**if** DXTRAN is being used **then**
  Setup DXTRAN information on mesh (see Alg. 5.5)
Allocate memory for forward source on mesh

**Algorithm 5.5:** Setup DXTRAN on Mesh

**forall** $S_N$ spatial cells **do**
  $\beta \leftarrow 1$
  `// Setup cell-center and cell-edge DXTRAN IDs and flags.`
  **if** $S_N$ cell inside DXTRAN region **then**
    Set cell center DXTRAN ID $\leftarrow 1$
    Set cell center inside DXTRAN flag $\leftarrow$ True
  **else**
    Set cell center DXTRAN ID $\leftarrow 0$
    Set cell center inside DXTRAN flag $\leftarrow$ False
    $\beta \leftarrow \beta_c$
  **if** $S_N$ cell edge is DXTRAN boundary **then**
    Set cell edge DXTRAN ID $\leftarrow 1$
    Set cell edge is DXTRAN boundary flag $\leftarrow$ True
  **else**
    Set cell edge DXTRAN ID $\leftarrow 0$
    Set cell edge is DXTRAN boundary flag $\leftarrow$ False
    $\beta \leftarrow \beta_c$
  `// Calculate cell-center and cell-edge DXTRAN biasing kernel values.`
  **forall** $S_N$ quadrature directions **do**
    Set cell-center and cell-edge $p_{\text{ff}} \leftarrow 0$
    Set cell-center and cell-edge $\widetilde{p} \leftarrow 0$
    Set cell-center and cell-edge $n_{\text{surf.}} \leftarrow 0$
    `// Calculate cell-center and cell-edge` $p_{\text{ff}}$ `values.`
    **if** $S_N$ cell can see DXTRAN region along $S_N$ direction **then**
      Compute optical distance from $S_N$ cell to DXTRAN region, $\lambda$
      $p_{\text{ff}} \leftarrow \exp(-\lambda)$
      $n_{\text{surf.}} \leftarrow$ Monte Carlo geometry surfaces between $S_N$ cell to DXTRAN region
      Set $\widetilde{p} \leftarrow q_n$ to indicate that the direction is valid for DXTRAN emergence
    Renormalize $\widetilde{p}$ to be valid PDF

## 5.4 Python-COVRT Optimization Linkage

When COVRT was originally developed, a gradient-descent based optimization scheme was implemented as described in Chapter 4. Because this approach is generally considered slow and out of the desire to investigate other optimization schemes, COVRT is coupled to Python. By performing this coupling, a rich set of optimization algorithms such as those described in Chapter 4 become immediately available with minimal additional effort. The coupling to Python is performed using the Python `ctypes` capability, which allows Python to directly call C-bound shared objects. As such, the COVRT functions and subroutines that need to be called to perform problem setup and to calculate the computational cost are modified to include `iso_c_binding` functionality [120].

The overall Python-driven COVRT (dubbed PyCOVRT) calculation is shown in Fig. 5.8. As indicated, the major calculation steps described in Section 5.3.2 are encapsulated as a Fortran shared objected called by Python. In short, Python executes the once-through program flow shown in Fig. 5.3 but when the computational cost is first calculated Python determines if optimization should be performed. If so, and if a minimum value has not been identified, the MADS algorithm varies the Monte Carlo cell-wise DXTRAN rouletting parameters, $\beta_c$, and the DXTRAN spatial extents and then uses COVRT to recompute an updated $\widehat{\Psi}_2$ and $\widehat{\Upsilon}$ to obtain updated $\widehat{M_2}$ and $\widehat{T}$ to ultimately obtain an updated $\widehat{\$}$. The new computational cost is then used as the black box objective function value to continue executing the MADS algorithm until a minimum is identified or the process aborted.

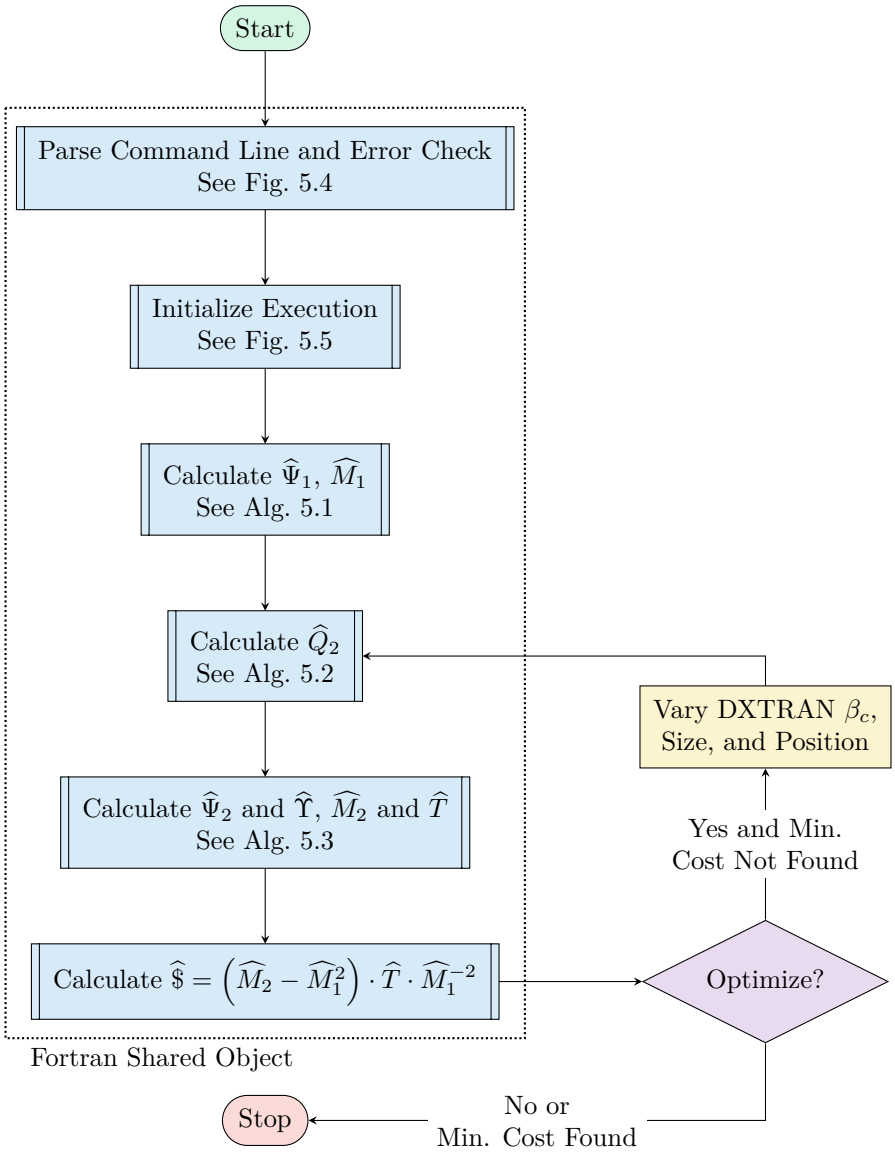



**Figure 5.8:** Python-driven COVRT Optimization Process Flow

# Chapter 6

# Computational Verification of Analytic Equations

Before computational cost optimization is performed, it is necessary to verify that the equations derived in Chapter 3 are implemented consistent with the MCNP code according to Chapter 5. The FTE was derived and validated in Chapter 4, so it is not addressed in this chapter. A variety of 1-D and 2-D test cases are created, described below, that verify the values of mean and population variance values predicted by COVRT, $\widehat{\mu}$ and $\widehat{\sigma}^2$, match reference solutions from MCNP calculations. The MCNP calculations are statistically well-converged, so it is expected that COVRT mean and variance values will agree with the MCNP values similar to prior work [58].

This chapter begins with a brief overview of all 1-D and 2-D test cases. The chapter continues with a detailed definition of all test cases. Finally, summary tables with comparisons to MCNP results and representative plots of statistical moment spatial distributions are given for all test cases. For the 1-D cases, the MCNP and COVRT mean and variances agree within 1.4% when DXTRAN is the only variance-reduction technique. When importance splitting and DXTRAN are used in concert, the agreement is within 3.4%. For the 2-D cases, the agreement is generally well within 10% and never worse than 13%, which is consistent with prior analyses for other variance-reduction techniques.

## 6.1 Introduction and Test Case Summary

Unless otherwise noted, test cases are executed in two ways (1) using strictly analog transport (including disabling MCNP6's default implicit capture, weight cutoff, and point-detector / DXTRAN weight rouletting behavior) and (2) using a DXTRAN region but otherwise keeping the transport analog as in the previous configuration.

In all cases, the dimensions are in centimeters unless otherwise noted. Outside of the problem is a vacuum. Because all systems are convex, there are no re-entrant particles. As such, when a Monte Carlo particle leaves the geometry it is killed, so the region outside the system is referred to as the graveyard.

All 1-D test cases are contained in a 10-cm region. All materials have $\Sigma_{\mathrm{t}} = 0.1$ cm$^{-1}$. Each case varies by type of material (absorber vs. scatterer), tally (leakage current, expected track length, or internal current), source (boundary or distributed), DXTRAN space-dependent rouletting parameter $\beta(\boldsymbol{x})$, and/or Monte Carlo cell-based importance parameter. Additional details on the 1-D test cases are given in Sections 6.2.1–6.2.14.

The 2-D test cases attempt to examine a variety of streaming and attenuation behaviors. Test Cases 2-1 and 2-2 act as simple verification cases while Test Cases 2-4–2-7 are representative of angle-dependent problems. This is because the variously configured ducts in these problems provide preferential directions of travel relative to the attenuating material surrounding the ducts. Additional details are given in Sections 6.3.1–6.3.7.

For convenience, all test cases are also summarized in Table 6.1.

**Table 6.1:** Summary of COVRT Test Cases

| # | Description |
|---|---|
| 1-1 | 1-D, mono-energetic, homogeneous pure absorber, $0 \leq x \leq 10$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$. See Fig. 6.1. |
| 1-2 | 1-D, mono-energetic, homogeneous 20% absorber, $0 \leq x \leq 10$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$. See Fig. 6.2. |
| 1-3 | 1-D, mono-energetic, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$. See Fig. 6.3. |
| 1-4 | 1-D, mono-energetic, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$, with non-DXTRAN cells incorporating a $\beta(x)$ value based on the Monte Carlo cell-center optical distance from the DXTRAN surface. |
| 1-5 | 1-D, mono-energetic, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, track-length flux tally in $8 < x < 10$. See Fig. 6.4. |
| 1-6 | 1-D, mono-energetic, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, distributed isotropic unit source in $0 < x < 3$, track-length flux tally in $8 < x < 10$. See Fig. 6.5. |
| 1-7 | 1-D, two-group, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$. |
| 1-8 | 1-D, two-group, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, distributed isotropic unit source in $0 < x < 3$, track-length flux tally in $8 < x < 10$, with non-DXTRAN cells incorporating a $\beta(x)$ value based on the Monte Carlo cell-center optical distance from the DXTRAN surface. |
| 1-9 | 1-D, mono-energetic, homogeneous pure absorber, $0 \leq x \leq 10$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$, with DXTRAN and Monte Carlo cell-based $2:1$ importance splitting. |
| 1-10 | 1-D, mono-energetic, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$, with DXTRAN and importance splitting with strictly $2:1$ importance-splitting ratios. |

continued on next page...

Table 6.1, continued from previous page...

| # | Test Case Description |
|---|---|
| 1-11 | 1-D, mono-energetic, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$, with DXTRAN and importance splitting with various integer and non-integer importance-splitting ratios. |
| 1-12 | 1-D, mono-energetic, three-region problem with 20% absorber in $0 \leq x < 3$ and $7 < x \leq 10$ and pure absorber in $3 \leq x \leq 7$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, leakage current tally at $x = 10$, with DXTRAN and importance splitting with different various integer and non-integer importance-splitting ratios. |
| 1-13 | 1-D, mono-energetic, homogeneous 20% absorber, $0 \leq x \leq 10$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, surface current tally at $x = 6$. |
| 1-14 | 1-D, mono-energetic, homogeneous 20% absorber, $0 \leq x \leq 10$, $\Sigma_t = 0.1$, isotropic boundary source at $x = 0$, expected track length tally in $6 < x < 8$. |
| 2-1 | 2-D, mono-energetic, homogeneous pure absorber, $-5 \leq x, y \leq 5$, $\Sigma_t = 0.1$, isotropic boundary source along $x = -5$, leakage current tally along $x = 5$. See Fig. 6.6. |
| 2-2 | 2-D, mono-energetic, homogeneous 20% absorber, $-5 \leq x, y \leq 5$, $\Sigma_t = 0.1$, isotropic boundary source along $x = -5$, leakage current tally along $x = 5$. See Fig. 6.7. |
| 2-3 | 2-D, mono-energetic, homogeneous block problem featuring a square distributed isotropic source and track-length tally in opposite corners (from [58]). See Fig. 6.8. |
| 2-4 | 2-D, void two-legged duct problem (from [58]) with a DXTRAN region around the tally region. See Fig. 6.9. |
| 2-5 | 2-D, void three-legged duct problem (from [58]). See Fig. 6.10. |
| 2-6 | 2-D, void two-legged duct problem (from [58]) with a DXTRAN region around the duct elbow (to "plug" the duct). See Fig. 6.11. |
| 2-7 | 2-D, void straight duct problem with a DXTRAN region in the duct to "plug" the duct. See Fig. 6.13. |

## 6.2 One-dimensional Test Cases

The approach to developing the 1-D test cases used to verify this work is incremental in nature. That is, the test cases get progressively more complex in terms of the physics and Monte Carlo features they incorporate. In this way, the tests act as

1. development aids while capabilities are introduced into COVRT,
2. a pseudo unit test suite where specific features are exercised in only certain cases, and
3. a regression test suite that can be quickly rerun to identify unintended consequences to code changes (that can be localized relatively easily using the earlier point on feature segregation).

For each of the 1-D test cases, a variety of DXTRAN positions are used to assess the resulting behavior (to verify the theory for a variety of configurations) and to identify any trends in mean and variance behavior.

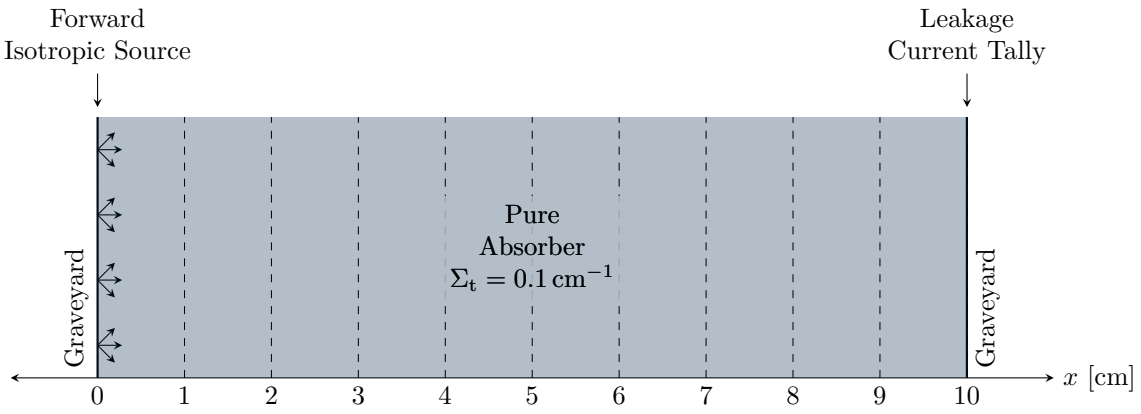



**Figure 6.1:** Test Case 1-1 Configuration

All 1-D test cases have a spatial extent of $0 \leq x \leq 10$ cm and an optical thickness no greater than one mean free path for any given energy. The test cases are kept relatively optically thin to ensure statistically converged MC results are obtained in a reasonable amount of time. However, these test cases have been examined to ensure that DXTRAN particles account for at least 10% of the scoring particles with DXTRAN particles produced by other DXTRAN particles accounting for at least 4% of scoring particles. Thus, there are sufficient DXTRAN-producing collisions within these test cases to allow them to act as verification cases.

All MCNP6 calculations are made with $10^7$ histories unless otherwise noted. This many histories provide sufficiently converged sample mean and variance values (evaluated with the estimated sample variance-of-the-variance) to act as population mean and population variance values to provide meaningful comparisons with COVRT.

All 1-D COVRT executions are made with an $S_{32}$ Gauss-Legendre quadrature [Section 2.3.3] with each spatial cell width specified as $\Delta x = 0.01$ cm. These parameters are chosen to minimize discretization errors. Furthermore, 1-D calculations do not exhibit ray effects making the verification process more simple than 2-D and 3-D geometries.

All calculations take place on a LANL workstation, which has four Intel Xeon CPU E5-4650 processors providing 32 physical cores (64 hyper-threaded cores) and 512 GB of memory. This computer's operating system is Redhat Enterprise Linux 7 (RHEL7) running Linux kernel version 3.10.0-862.el7.x86_64.

### 6.2.1 Test Case 1-1

Test Case 1-1 contains a homogeneous mono-energetic purely absorbing material with $\Sigma_t = \Sigma_a = 0.1$ cm$^{-1}$ as shown in Fig. 6.1. An incident isotropic surface source is positioned at $x = 0$ cm with an arbitrary magnitude of one. A leakage current tally is positioned at $x = 10$.

When $x_{DX} = 0$, strictly analog behavior is expected throughout the problem. Otherwise, analog behavior is expected to take place within the DXTRAN region. Because this problem is purely absorbing and because DXTRAN for this work is only initiated during collisions, only particles that undergo free-flight from the source to the tally are expected to score. Because these particles are not killed as they attempt to enter

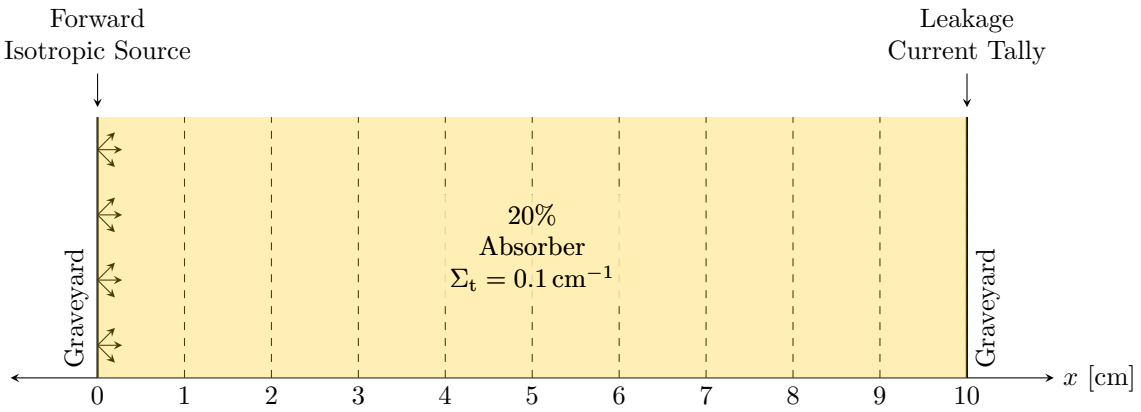



**Figure 6.2:** Test Case 1-2 Configuration

DXTRAN (because DXTRAN is not played on source emission for this work), analog behavior is observed external to the DXTRAN region. These behaviors are demonstrated when the test case results are described in Section 6.4.

### 6.2.2 Test Case 1-2

Test Case 1-2 is identical to Test Case 1-1 except that the purely absorbing material is replaced by a 20% absorbing material (with $\Sigma_t = 0.1$ cm$^{-1}$) as shown in Fig. 6.2. Thus, 80% of collisions external to the DXTRAN region result in DXTRAN particles. This collision-to-absorption ratio was selected to

1. ensure that the Monte Carlo calculations run in a reasonable amount of time,
2. force both absorptive and non-absorptive collisions, and
3. yield a sufficient number of collision-generated DXTRAN particles to act as a suitable test.

To point #3, examining the Monte Carlo tally behavior shows that DXTRAN daughter particles (particles with $> 2$ collisions scoring as a result of being born by DXTRAN particles that left the DXTRAN region and collided as described in Section 2.4.6.6) comprise just over 4% of the tally. If the MCNP and COVRT results agree to well within this value then the effect of these daughter particles has been accurately accounted for.

### 6.2.3 Test Case 1-3

Test Case 1-3 is mono-energetic but heterogeneous, combining the materials used in Test Cases 1-1 and 1-2, as shown in Fig. 6.3. The purely absorbing material (still with $\Sigma_t = \Sigma_a = 0.1$ cm$^{-1}$) is placed in the region $3 < x < 7$ and the 20% absorbing material ($\Sigma_t = 0.1$ cm$^{-1}$) is placed elsewhere. This test case represents a step toward reality in that it contains heterogeneity with the purely absorbing region representing space through which DXTRAN is hoped to draw particles that might otherwise be lost to absorptive collision.

In the early stage of this work, it was previously observed that this test case has some resemblance to the "Reed" test problem [122], so it was suggested that the Reed test problem be used in this work. The Reed

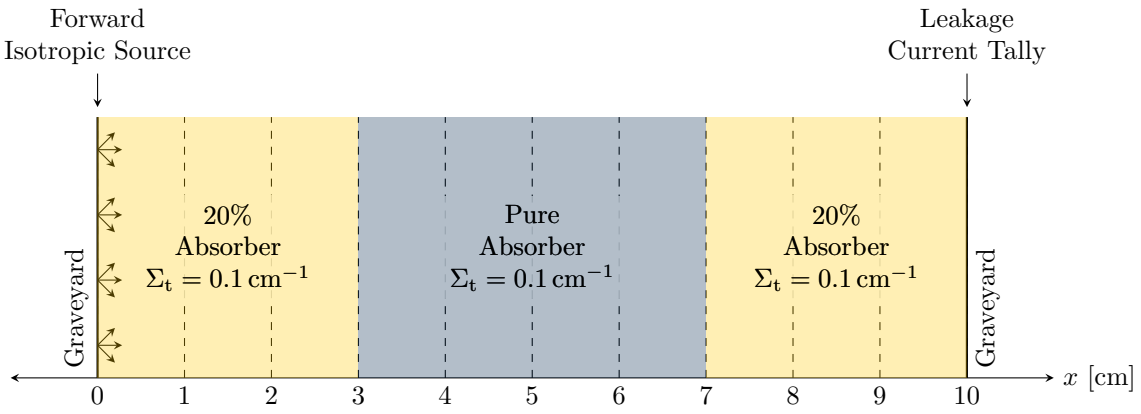



**Figure 6.3:** Test Case 1-3 Configuration

test problem is a 1-D planar heterogeneous non-multiplying artificial test case developed to expose weaknesses in deterministic differencing schemes. However, the Reed test problem is not used in this work. The current work applies the diamond-difference approximation, which has well characterized weaknesses. Furthermore, to use the Reed test problem requires modification to COVRT (to accommodate discontinuous distributed sources, an enhancement not necessary to verify the theory developed herein). Most significantly, the Reed test problem requires modifying both MCNP6 and COVRT because reflective boundaries cannot be used with DXTRAN. This inability is because there is no known method to perform the ray tracing necessary as part of the weight correction when DXTRAN particles are created. Because of this, reflective boundaries cannot be used with point detectors either.

While the discontinuous source issue could be addressed, the latter point regarding ray tracing is nontrivial. As such, the Reed test problem is not used herein but is mentioned for the benefit of future work.

### 6.2.4 Test Case 1-4

Test Case 1-4 uses the same heterogeneous material distribution as Test Case 1-3 but has a spatially varying DXTRAN rouletting parameter assigned. In theory, the DXTRAN rouletting parameter can vary continuously in space, but in practice (at least, in MCNP6) it is assigned on a geometry cell-wise basis. As such, to the left of $x_{\text{DX}}$ the DXTRAN parameter $\beta(x)$ is kept constant in each Monte Carlo cell but varied according to an approximate optical distance between the cell center and the left-hand side of the DXTRAN region, $x_{\text{DX}}$, as

$$\beta_c = \exp\left(-\Sigma_t \left(x_{\text{DX}} - x_c\right)\right), \tag{6.1}$$

where $\beta_c$ is the DXTRAN rouletting parameter applied to cell $c$ outside the DXTRAN region and $x_c$ is the midpoint of cell $c$. These values are selected because it is reasonable to assume that the probability of contribution to a tally within the DXTRAN region falls off exponentially according to the optical distance between the collision site that produces a DXTRAN particle and the DXTRAN region on which the particle is created. The value for $\beta_c$ is set to one for cells within the DXTRAN region.

As an example, for the 1-cm wide cell with total cross section $\Sigma_t = 0.1$ cm$^{-1}$ adjacent to the DXTRAN

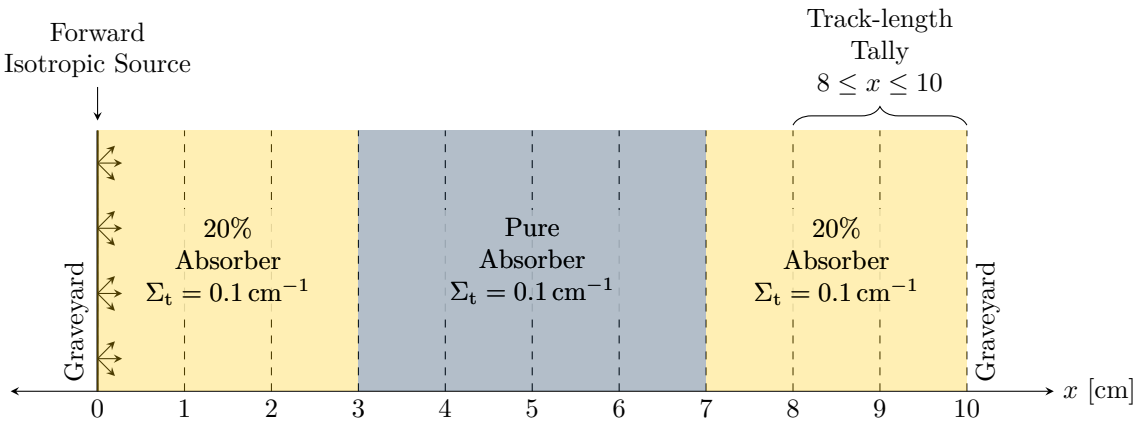



**Figure 6.4:** Test Case 1-5 Configuration

region with left hand boundary located at $x_{\mathrm{DX}} = 8$ cm,

$$\beta_c = \exp\left(-\Sigma_t \left(x_{\mathrm{DX}} - x_c\right)\right) = \exp\left(-0.1\left(8.0 - 7.5\right)\right) \approx 0.95. \tag{6.2}$$

### 6.2.5 Test Case 1-5

Test Case 1-5 uses the same heterogeneous material distribution as Test Case 1-3 but has an expected track-length tally in $8 < x < 10$ as shown in Fig. 6.4. Information on the expected track length tally's derivation and potential pitfalls is available in [58, Section 3.6]. In all MCNP calculations that incorporate the expected track-length tally, the particle score is calculated as

$$s = \frac{w}{\Sigma_{t,c}}\left[1 - \exp\left(-d_c \cdot \Sigma_{t,c}\right)\right], \tag{6.3}$$

where $w$ is the particle's weight, $d_c$ is the distance to the cell boundary, and $\Sigma_{t,c}$ is the macroscopic total cross section of the cell. This test case ensures that COVRT correctly accounts for track-length generated by DXTRAN particles in the tally region.

### 6.2.6 Test Case 1-6

Test Case 1-6 is the same as Test Case 1-5 except that instead of an isotropic boundary source, a distributed isotropic volumetric source is used in the region $0 \leq x \leq 2$ as shown in Fig. 6.5. This test case ensures that the inner product performed in Eq. (5.3) is performed correctly.

### 6.2.7 Test Case 1-7

Test Case 1-7 is the same as Test Case 1-3 except that there are two energy groups instead of one. The materials are defined by the cross sections given in Table 6.2. The forward source emits entirely in group 1. This provides a check that adjoint particles starting in group 2 scattering (via the adjoint scattering process) into group 1 are accurately accounted for.

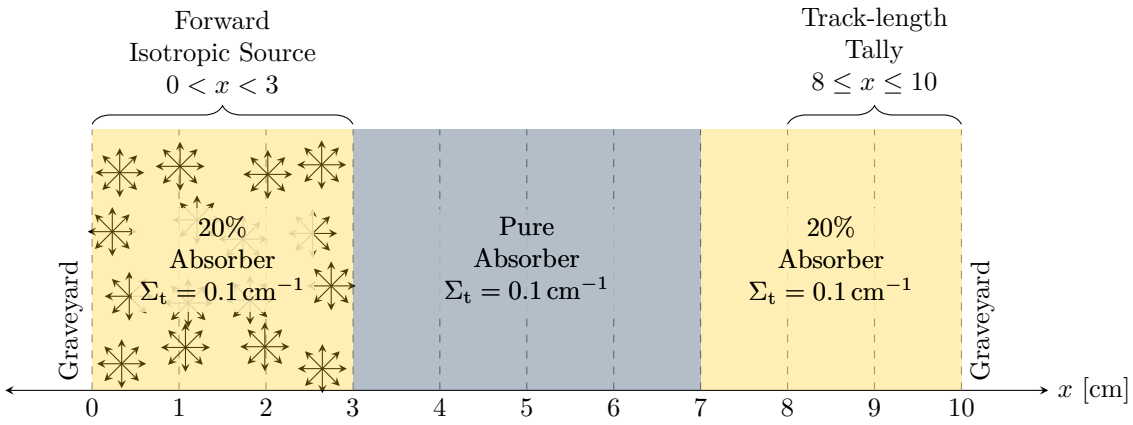



**Figure 6.5:** Test Case 1-6 Configuration

**Table 6.2:** Test Case 1-8 Two-group Cross Sections

| Material 1 | | | | | |
|---|---|---|---|---|---|
| Group, $g$ | $\Sigma_{t,g}$ | $\Sigma_{a,g}$ | $\Sigma_{s,g}$ | $\Sigma_{s,g\to1}$ | $\Sigma_{s,g\to2}$ |
| 1 | 0.10 | 0.01 | 0.09 | 0.06 | 0.03 |
| 2 | 0.07 | 0.02 | 0.05 | 0 | 0.05 |
| Material 2 | | | | | |
| Group, $g$ | $\Sigma_{t,g}$ | $\Sigma_{a,g}$ | $\Sigma_{s,g}$ | $\Sigma_{s,g\to1}$ | $\Sigma_{s,g\to2}$ |
| 1 | 0.08 | 0.05 | 0.03 | 0.01 | 0.02 |
| 2 | 0.03 | 0.03 | 0 | 0 | 0 |

### 6.2.8 Test Case 1-8

Test Case 1-8 is an amalgamation of Test Cases 1-4, 1-6, and 1-7. It consists of the two-group heterogeneous material distribution of Test Case 1-7, incorporates the same $\beta_c$ values as Test Case 1-4, and uses the same forward source and expected track-length tally as Test Case 1-6. As such, it represents the most complicated 1-D DXTRAN test case tested.

### 6.2.9 Test Case 1-9

Test Case 1-9 is the same as Test Case 1-1, except that each Monte Carlo cell is assigned an importance to produce $2:1$ splitting at each Monte Carlo surface as particles travel from left-to-right. This test case verifies the correct implementation of importance splitting by disallowing DXTRAN (it is a pure absorber). Furthermore, as a pure absorber, there will be no importance rouletting (particles cannot travel from right-to-left).

### 6.2.10 Test Case 1-10

Test Case 1-10 is the same as Test Case 1-3, except that each Monte Carlo cell is assigned an importance to produce $2:1$ splitting at each Monte Carlo surface as particles travel from left-to-right. This test case verifies the correct implementation of importance splitting with DXTRAN. While not a direct goal of this work, it is expected that other variance reduction techniques like importance splitting or weight windows will be used with DXTRAN in practice so it is valuable to assess the compatibility of these techniques.

### 6.2.11 Test Case 1-11

Test Case 1-11 is the same as Test Case 1-10, except various integer and non-integer splitting ratios are induced by varying adjacent cell importances. In this case, a maximum of $7:1$ splitting is induced near the tally region. This variation is used to ensure that a variety of splitting/rouletting ratios are compatible with DXTRAN.

### 6.2.12 Test Case 1-12

Test Case 1-12 is the same as Test Case 1-10, except various integer and non-integer splitting ratios are induced by varying adjacent cell importances. In this case, a maximum of $7:1$ splitting is induced near the forward source. This variation is used to ensure that a variety of splitting/rouletting ratios are compatible with DXTRAN.

### 6.2.13 Test Case 1-13

Test Case 1-13 is the same as Test Case 1-2 except that instead of a leakage current tally an internal current tally is positioned at $x = 6$. A variety of two-sided DXTRAN regions are defined around the tally from intimately fitting it to covering the entire problem. This test case tests internal current tallies with DXTRAN and ensures that two-sided DXTRAN regions work correctly.

### 6.2.14 Test Case 1-14

Test Case 1-14 is the same as Test Case 1-2 except that instead of a leakage current tally an internal expected track-length tally is positioned in the region $6 < x < 8$. A variety of two-sided DXTRAN regions are defined around the tally from intimately fitting it to covering the entire problem. This test case tests internal expected track-length tallies with DXTRAN and ensures that two-sided DXTRAN regions work correctly.

## 6.3 Two-dimensional Test Cases

The 1-D test cases described previously verify that the HSMEs are implemented correctly in COVRT but they cannot readily provide angle-dependent problems to examine. As such, seven 2-D test case geometries are used. Test Cases 2-1–2-3 act as additional simple verification cases but Test Cases 2-4–2-7 provide streaming regions that cause anisotropic behavior characteristic of angle-dependent transport problems. In addition, Test Cases 2-2–2-7 are used in Chapter 7 to examine the behavior of DXTRAN when optimization is performed. Each test case is described in more detail next.

### 6.3.1 Test Case 2-1

Test Case 2-1 is a mono-energetic homogeneous pure absorber ($\Sigma_t = \Sigma_a = 0.1$ cm$^{-1}$) with $-5 \leq x, y \leq 5$ as shown in Fig. 6.6. An isotropic surface source is positioned along the left-hand boundary at $x = -5$ between $-5 < y < 5$. A leakage current tally is positioned along the right-hand boundary at $x = 5$ between $-5 < y < 5$. A DXTRAN region is defined between $0 < x < 5$ and $-5 < y < 5$. As with Test Case 1-1, this test case is expected to behave like an analog calculation because there is no opportunity for DXTRAN to be initiated.

### 6.3.2 Test Case 2-2

Test Case 2-2 is identical to Test Case 2-1 except that the pure absorber material is replaced by a 20% absorber ($\Sigma_t = 0.1$ cm$^{-1}$) as shown in Fig. 6.7.

### 6.3.3 Test Case 2-3

Test Case 2-3 is a mono-energetic homogeneous 20% absorber ($\Sigma_t = 0.5$ cm$^{-1}$) with $-5 \leq x, y \leq 5$. However, a isotropic distributed volumetric source is defined in the region $-5 < x, y < -3$ and an expected track length tally is defined in the region $3 < x, y < 5$. A DXTRAN region closely surrounds the expected track length tally as shown in Fig. 6.8. This test case is comparable to the homogeneous block problem from [58]. It is used to verify that isotropic volumetric sources and expected track-length tallies work correctly with a DXTRAN region in 2-D.

### 6.3.4 Test Case 2-4

Test Case 2-4 is characterized by a 2-cm wide void duct with a 90-degree bend (forming two legs) within a mono-energetic 20% absorber ($\Sigma_t = 0.1$ cm$^{-1}$) with a total problem extent of $-5 \leq x, y \leq 5$ as shown in Fig. 6.9. An isotropic boundary source exists in the region $-2 < x < 0$ at $y = -5$. A leakage current tally is placed between $0 < y < 2$ at $x = 5$. This problem is comparable to the two-legged duct problem from [58]. A DXTRAN region closely surrounds the leakage current tally.

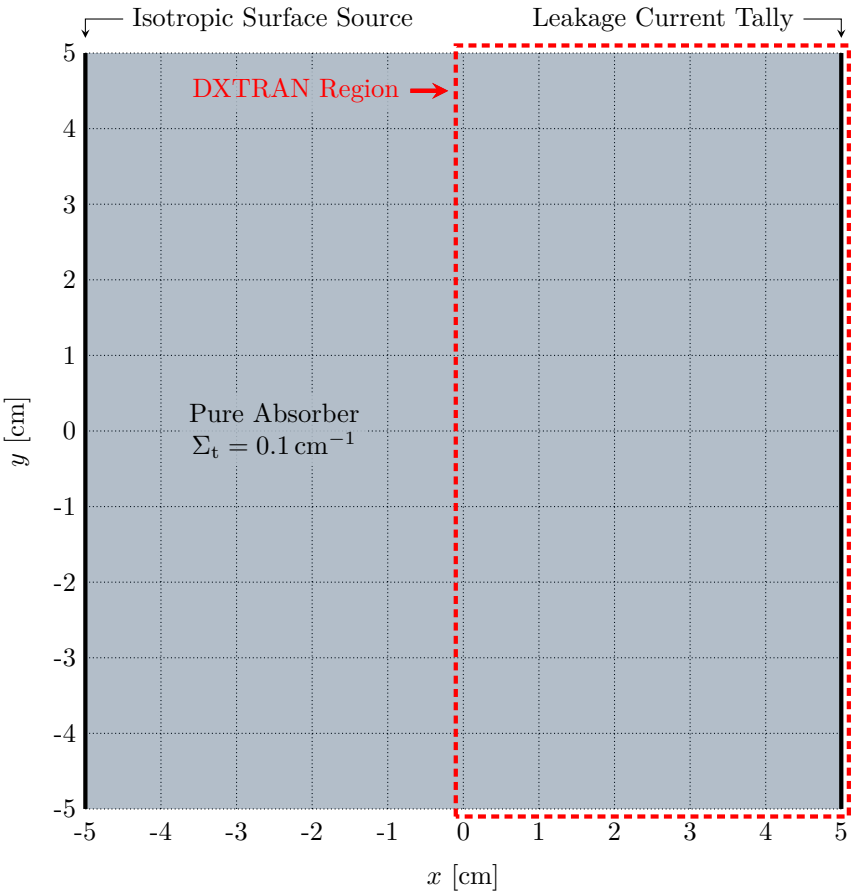



**Figure 6.6:** Test Case 2-1 Configuration

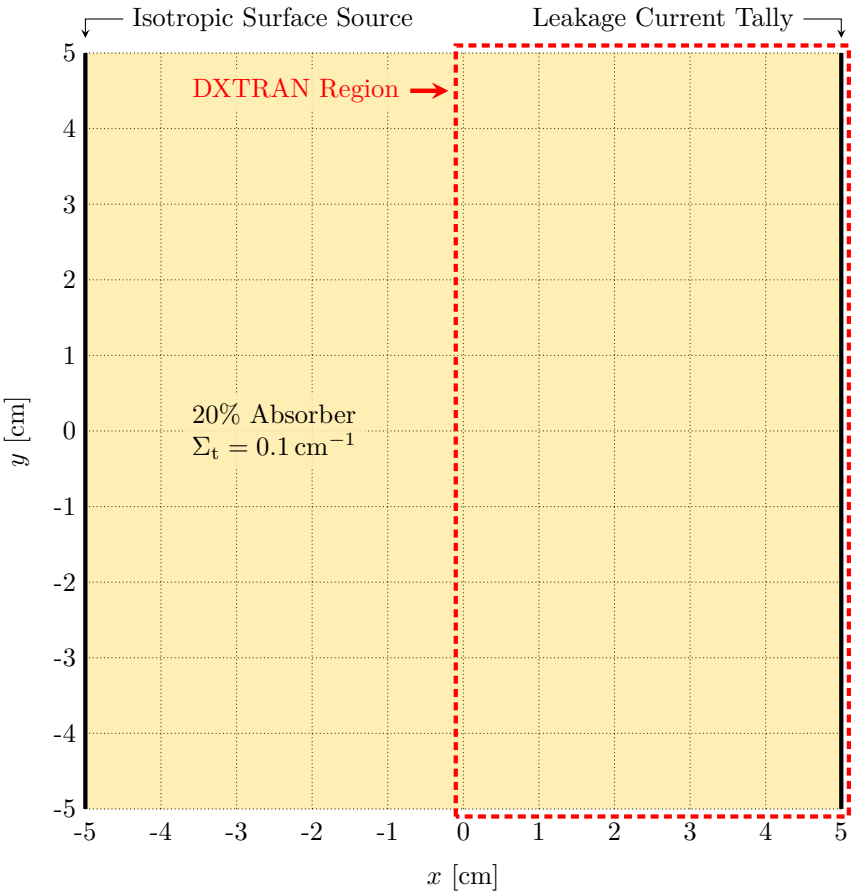



**Figure 6.7:** Test Case 2-2 Configuration

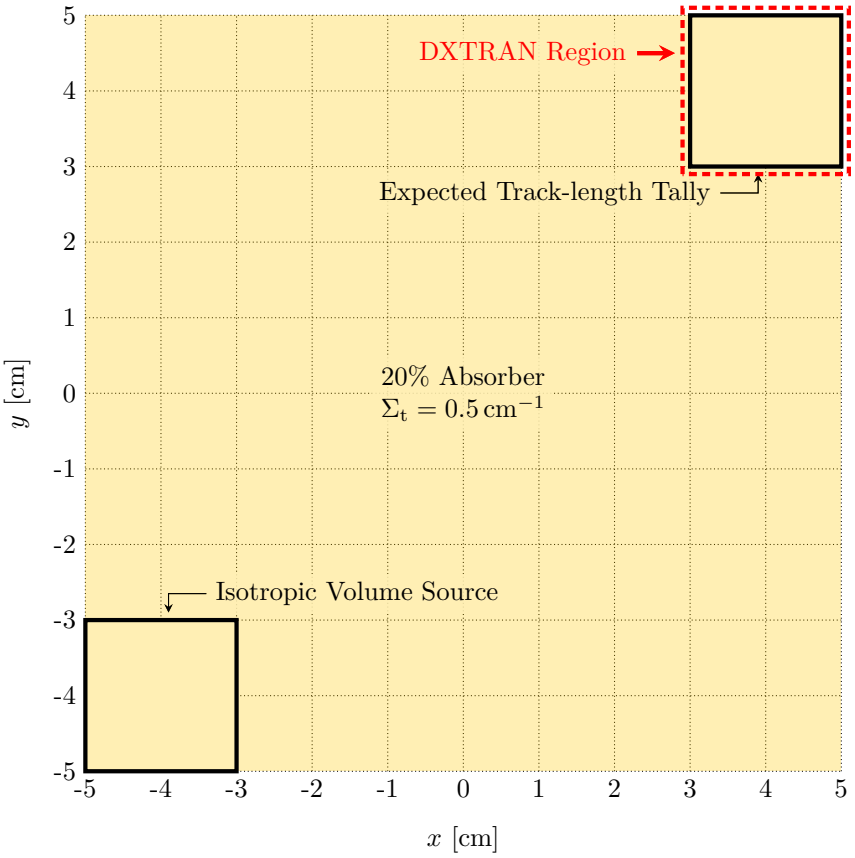



**Figure 6.8:** Test Case 2-3 Configuration

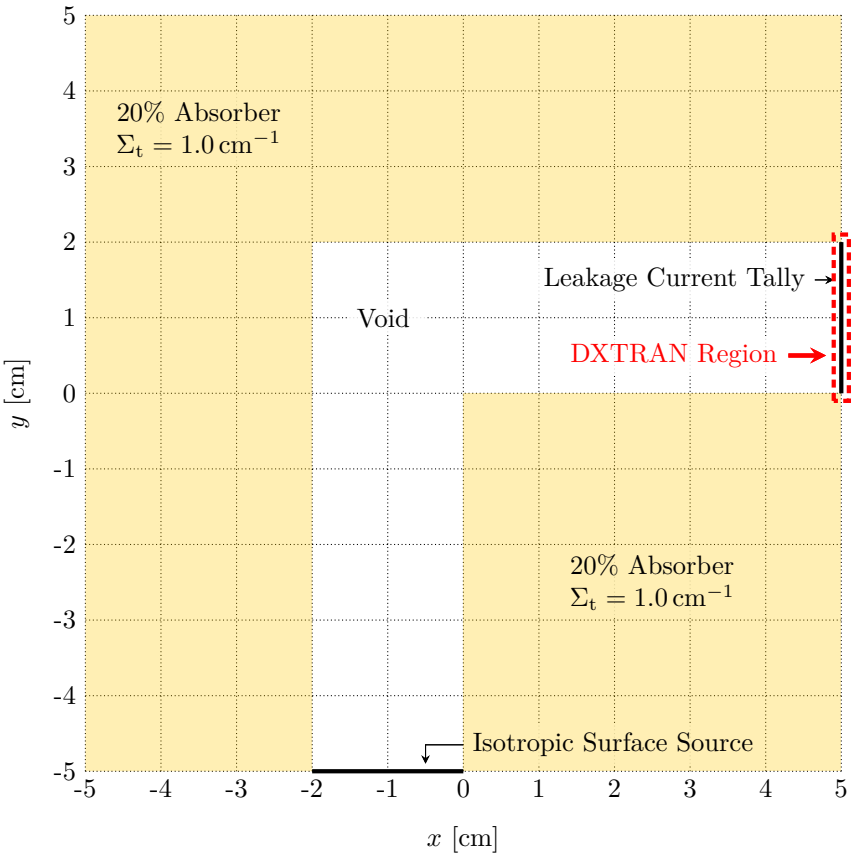



**Figure 6.9:** Test Case 2-4 Configuration

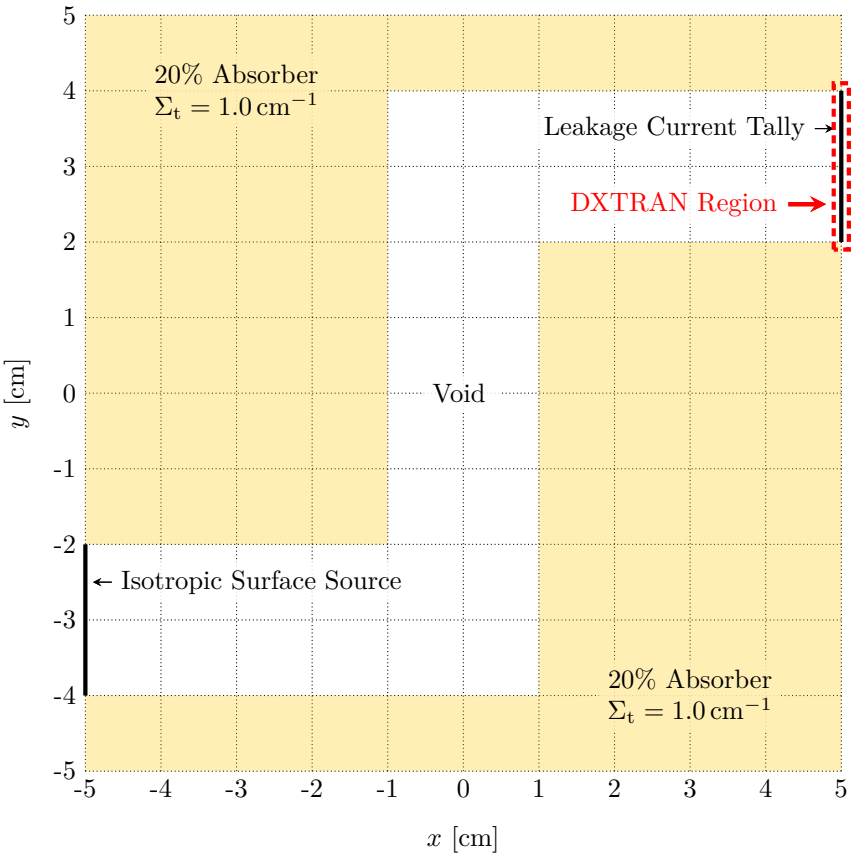



**Figure 6.10:** Test Case 2-5 Configuration

### 6.3.5 Test Case 2-5

Test Case 2-5 is characterized by a 2-cm wide void duct with two 90-degree bends (forming three legs) within a mono-energetic 20% absorber ($\Sigma_t = 0.1$ cm$^{-1}$) with a total problem extent of $-5 \leq x, y \leq 5$ as shown in Fig. 6.10. An isotropic boundary source exists at $x = -5$ in the region $-4 < y < -2$. A leakage current tally is placed at $x = 5$ between $2 < y < 4$. This problem is comparable to the three-legged / "dogleg" duct problem from [58]. A DXTRAN region closely surrounds the leakage current tally.

### 6.3.6 Test Case 2-6

Test Case 2-6 is identical to Test Case 2-4 except that the DXTRAN region is used to "plug the duct" by being placed in the elbow of the duct as shown in Fig. 6.11. This test case attempts to explore the alternative application of DXTRAN where it acts as a "shield" (to block particles from transmitting through a particular region) rather than as a "magnet" (drawing particles toward a region).

For this test case, the MCNP6 calculations were initially too long to be practical even before considering optimization. This is because particles projected to the DXTRAN region would pass through the DXTRAN region, collide, and generate subsequent DXTRAN collisions. This behavior can be contrasted to other cases that position the DXTRAN region near the problem boundary so DXTRAN-generated particles

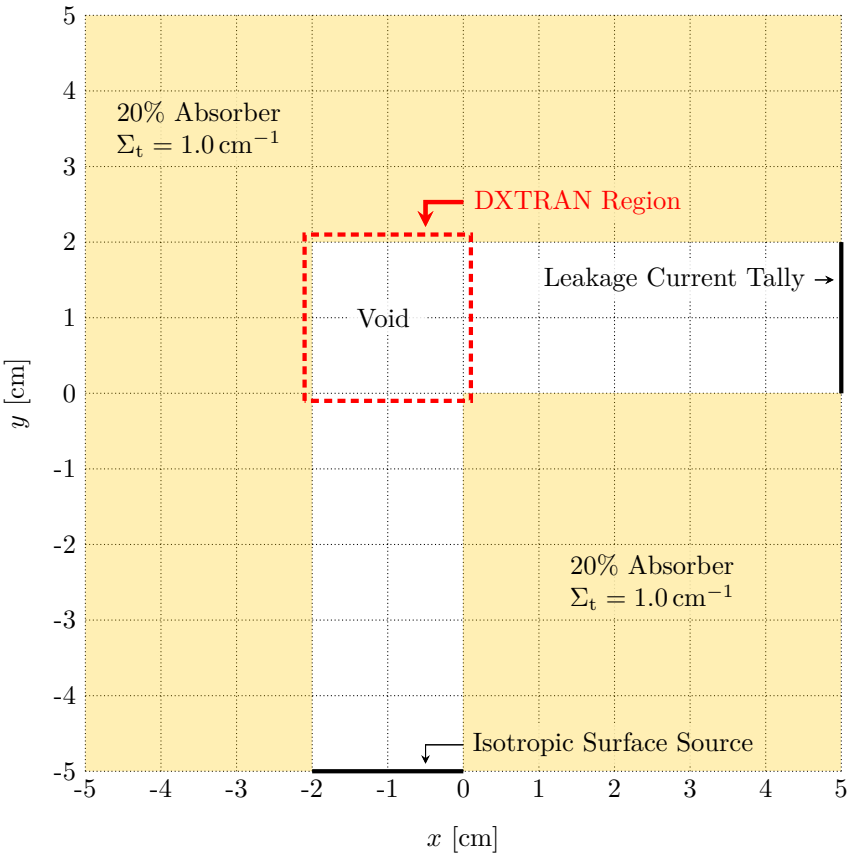



**Figure 6.11:** Test Case 2-6 Configuration

that penetrated through the region did not have the opportunity to create subsequent DXTRAN particles. Regardless, the resulting cascade of DXTRAN particles in this case leads to intractably long calculation Monte Carlo times.

A standard solution to this behavior is to use a weight control mechanism to roulette low-weight particles. In all prior test cases, this (default) behavior is disabled with the MCNP `dd` card entry "`dd1 0`," which disables rouletting. Recognizing that such rouletting is necessary but with the desire to not unduly modify the sample variance results with a weight-dependent rouletting game, a sensitivity study is performed on the `dd` card for this test case. The dd card is changed to "`dd1` $w_c$" with $w_c = -\left\{10^{-1}, 10^{-2}, 10^{-3}, \ldots, 10^{-8}\right\}$. Specifying the cutoff values as negative causes them to be absolute and applied to all histories (cf. default MCNP behavior where an average weight is determined from the first 200 histories and then periodically recalculated).

The resulting mean, variance, and execution times are shown in Fig. 6.12. Note that Fig. 6.12b is log-scaled. Considering the convergence to the mean and variance observed in Fig. 6.12a, the `dd` card entry used for Test Case 2-6 is "`dd1 1e-4`".

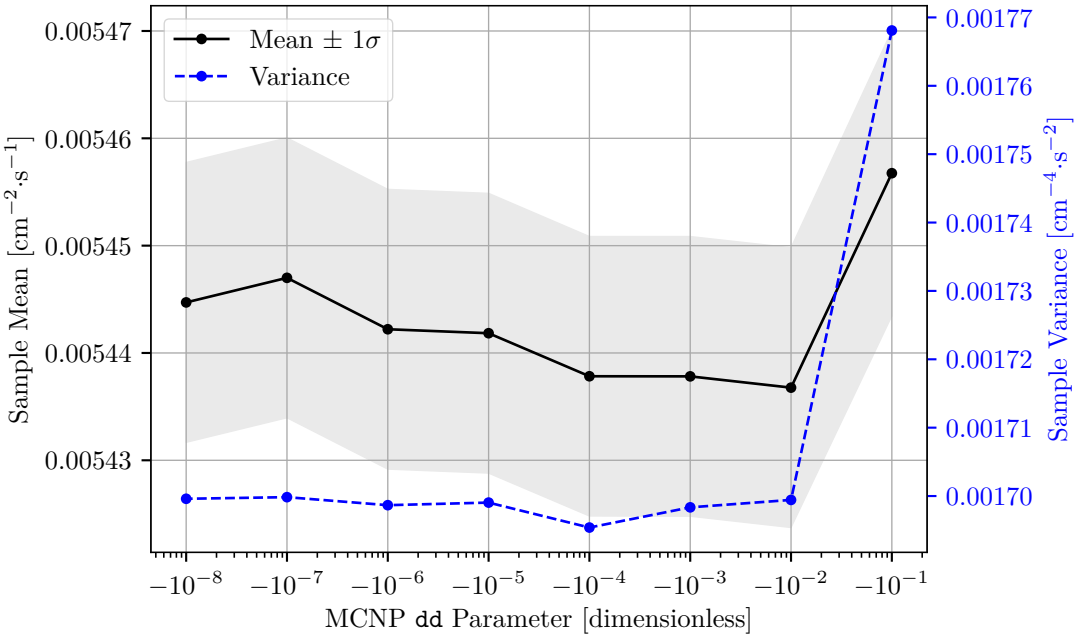
### 6.3.7 Test Case 2-7

Test Case 2-7 is characterized by a straight duct with a DXTRAN region used to plug the duct. However, it features an offset forward source (the forward source is not aligned with the end of the duct) and a leakage current tally along the opposite boundary of the problem (rather than just along the duct exit). The arrangement of these features is shown in Fig. 6.13. This test case is loosely inspired by the application of a DXTRAN region in [123].

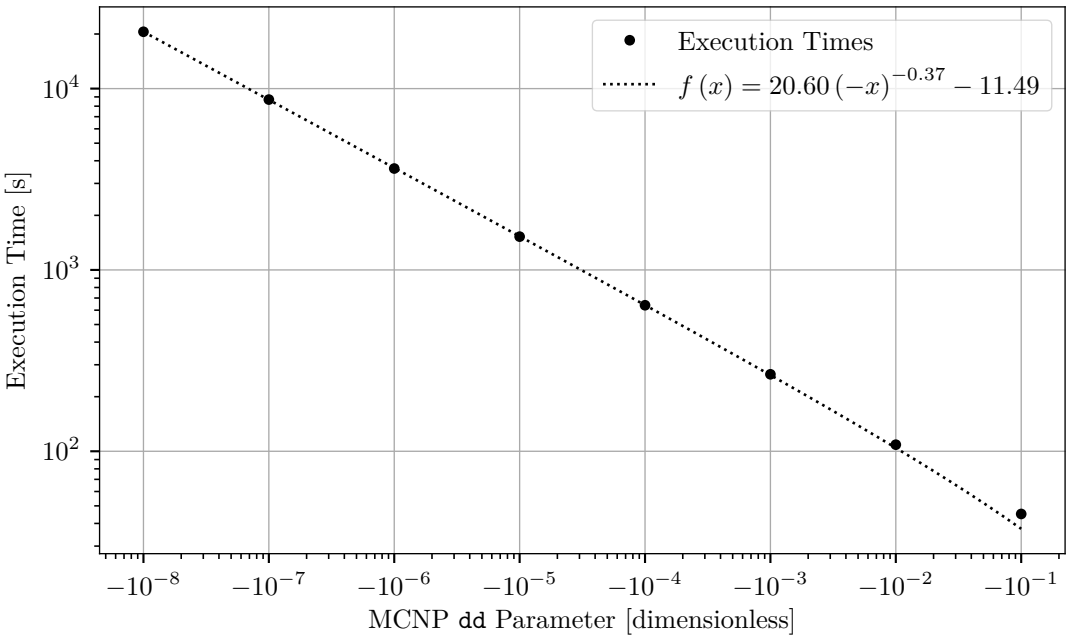
## 6.4 One-dimensional Test Case Results

Each 1-D test case is executed using both MCNP6 and COVRT. The MCNP6 sample mean ($\mu_{\mathrm{MC}}$) and relative standard fractional uncertainty in the sample mean ($R_{\mathrm{MC}}$) values are retrieved and the Monte Carlo sample variance is calculated with Eq. (2.80). The MCNP6 sample mean and sample variance results are then compared with the COVRT-produced values by dividing the MCNP6 values by the COVRT values. The resulting ratios by DXTRAN position for Test Cases 1-1–1-14 are given in Tables 6.3–6.16. Figures showing representative scalar $\widehat{\Psi}_1(x)$ and $\widehat{\Psi}_2(x)$ traverses for each test case are given in Figures 6.14–6.27. Because all variance reduction is weight-independent, there is no need to consider weight groups (i.e., all moments can be represented in a single weight group). For all test cases, the ratio of the Monte Carlo and deterministic mean values is 1.00 indicating agreement within a half percent.

For Test Cases 1-1 through 1-4, the variance values also agree within a half percent across the range of DXTRAN positions. This is important because it demonstrates that increased collisions outside the DXTRAN region (as the DXTRAN region shrinks) that initiate DXTRAN are treated appropriately. Test Case 1-1 exhibits the pure absorber analog behavior as described in Section 6.1. Because the MCNP and COVRT mean and variance for Test Case 1-2 agree well within 1% the effect of DXTRAN daughter particles (that account for 4% of the tally value) is accurately accounted for. Test Case 1-5 and 1-6 incorporate an expected track length tally and agree within 1.5 percent. This is a reasonable level of agreement given the modifications to the Monte Carlo tally to accommodate the expected track length tally (see Section 6.2.5) and is consistent




**(a)** Mean and Variance




**(b)** Execution Time (using 36 OpenMP Threads)

**Figure 6.12:** Test Case 2-6 dd Absolute Cutoff Sensitivity Study Results

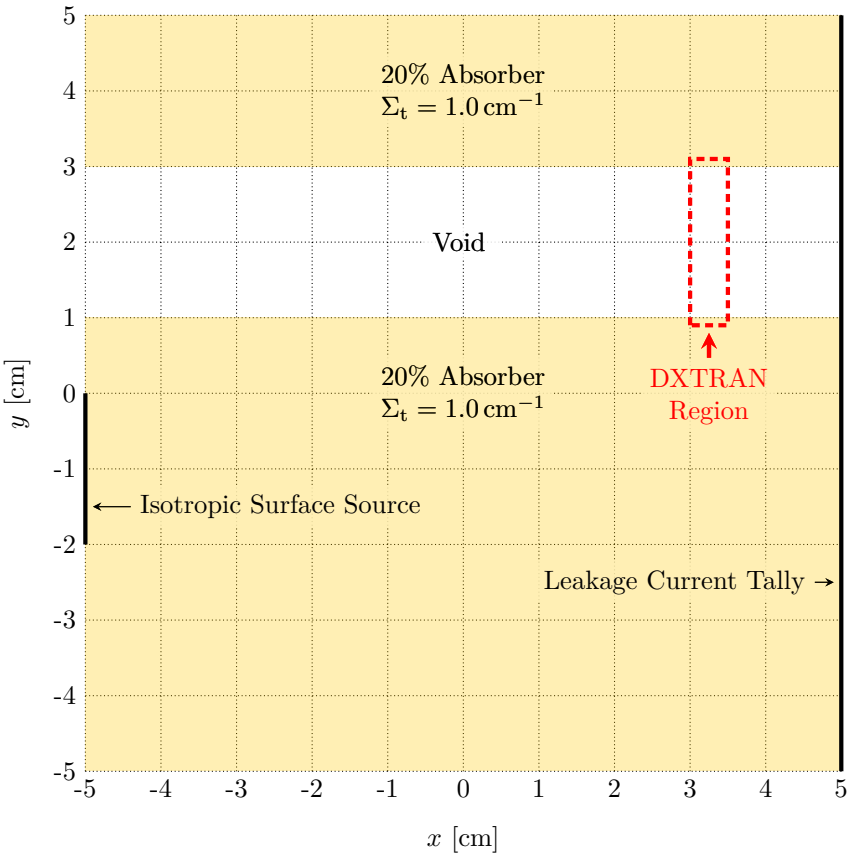



**Figure 6.13:** Test Case 2-7 Configuration

with prior work [58]. The test cases with two energy groups (Test Cases 1-7 and 1-8) show consistent levels of variance agreement.

Test Cases 1-10–1-12 combines importance splitting/rouletting with a DXTRAN region. The agreement in these cases between the MCNP and COVRT results is no worse than 3.4%. Test Cases 1-10 and 1-11 suggests a trend of increasing disagreement as the DXTRAN region shrinks; however, Test Cases 1-12 does not exhibit a clear trend. While this level of agreement between the MCNP and COVRT results is not as good as the other, DXTRAN-only, cases it is consistent with an alternative implementation of importance splitting combined with DXTRAN developed as a course of this work.

**Table 6.3:** Test Case 1-1 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 1 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 2 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 3 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 4 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 5 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 6 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 7 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 8 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 9 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |
| 10 | 10 | 7.437e-02 | 6.884e-02 | 7.425e-02 | 6.873e-02 | 1.00 | 1.00 |

**Table 6.4:** Test Case 1-2 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 1.721e-01 | 1.425e-01 | 1.721e-01 | 1.425e-01 | 1.00 | 1.00 |
| 1 | 10 | 1.722e-01 | 1.302e-01 | 1.721e-01 | 1.302e-01 | 1.00 | 1.00 |
| 2 | 10 | 1.722e-01 | 1.226e-01 | 1.721e-01 | 1.225e-01 | 1.00 | 1.00 |
| 3 | 10 | 1.722e-01 | 1.168e-01 | 1.721e-01 | 1.168e-01 | 1.00 | 1.00 |
| 4 | 10 | 1.721e-01 | 1.124e-01 | 1.721e-01 | 1.123e-01 | 1.00 | 1.00 |
| 5 | 10 | 1.721e-01 | 1.090e-01 | 1.721e-01 | 1.090e-01 | 1.00 | 1.00 |
| 6 | 10 | 1.721e-01 | 1.067e-01 | 1.721e-01 | 1.066e-01 | 1.00 | 1.00 |
| 7 | 10 | 1.721e-01 | 1.051e-01 | 1.721e-01 | 1.050e-01 | 1.00 | 1.00 |
| 8 | 10 | 1.721e-01 | 1.043e-01 | 1.721e-01 | 1.043e-01 | 1.00 | 1.00 |
| 9 | 10 | 1.721e-01 | 1.045e-01 | 1.721e-01 | 1.043e-01 | 1.00 | 1.00 |
| 10 | 10 | 1.721e-01 | 1.053e-01 | 1.721e-01 | 1.053e-01 | 1.00 | 1.00 |

**Table 6.5:** Test Case 1-3 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 1.216e-01 | 1.068e-01 | 1.215e-01 | 1.067e-01 | 1.00 | 1.00 |
| 1 | 10 | 1.216e-01 | 9.852e-02 | 1.215e-01 | 9.846e-02 | 1.00 | 1.00 |
| 2 | 10 | 1.216e-01 | 9.354e-02 | 1.215e-01 | 9.345e-02 | 1.00 | 1.00 |
| 3 | 10 | 1.216e-01 | 8.998e-02 | 1.215e-01 | 8.991e-02 | 1.00 | 1.00 |
| 4 | 10 | 1.216e-01 | 8.822e-02 | 1.215e-01 | 8.814e-02 | 1.00 | 1.00 |
| 5 | 10 | 1.216e-01 | 8.675e-02 | 1.215e-01 | 8.666e-02 | 1.00 | 1.00 |
| 6 | 10 | 1.216e-01 | 8.551e-02 | 1.215e-01 | 8.541e-02 | 1.00 | 1.00 |
| 7 | 10 | 1.216e-01 | 8.445e-02 | 1.215e-01 | 8.435e-02 | 1.00 | 1.00 |
| 8 | 10 | 1.216e-01 | 8.207e-02 | 1.215e-01 | 8.196e-02 | 1.00 | 1.00 |
| 9 | 10 | 1.216e-01 | 8.068e-02 | 1.215e-01 | 8.055e-02 | 1.00 | 1.00 |
| 10 | 10 | 1.216e-01 | 8.017e-02 | 1.215e-01 | 8.004e-02 | 1.00 | 1.00 |

**Table 6.6:** Test Case 1-4 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 1.216e-01 | 1.068e-01 | 1.215e-01 | 1.067e-01 | 1.00 | 1.00 |
| 1 | 10 | 1.216e-01 | 9.892e-02 | 1.215e-01 | 9.885e-02 | 1.00 | 1.00 |
| 2 | 10 | 1.216e-01 | 9.471e-02 | 1.215e-01 | 9.463e-02 | 1.00 | 1.00 |
| 3 | 10 | 1.216e-01 | 9.211e-02 | 1.215e-01 | 9.205e-02 | 1.00 | 1.00 |
| 4 | 10 | 1.216e-01 | 9.139e-02 | 1.215e-01 | 9.135e-02 | 1.00 | 1.00 |
| 5 | 10 | 1.216e-01 | 9.071e-02 | 1.215e-01 | 9.067e-02 | 1.00 | 1.00 |
| 6 | 10 | 1.216e-01 | 9.013e-02 | 1.215e-01 | 9.007e-02 | 1.00 | 1.00 |
| 7 | 10 | 1.215e-01 | 8.958e-02 | 1.215e-01 | 8.954e-02 | 1.00 | 1.00 |
| 8 | 10 | 1.216e-01 | 8.757e-02 | 1.215e-01 | 8.750e-02 | 1.00 | 1.00 |
| 9 | 10 | 1.216e-01 | 8.658e-02 | 1.215e-01 | 8.653e-02 | 1.00 | 1.00 |
| 10 | 10 | 1.216e-01 | 8.651e-02 | 1.215e-01 | 8.648e-02 | 1.00 | 1.00 |

**Table 6.7:** Test Case 1-5 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 2.509e-01 | 6.235e-01 | 2.506e-01 | 6.196e-01 | 1.00 | 1.01 |
| 1 | 10 | 2.510e-01 | 5.740e-01 | 2.506e-01 | 5.709e-01 | 1.00 | 1.01 |
| 2 | 10 | 2.509e-01 | 5.412e-01 | 2.506e-01 | 5.384e-01 | 1.00 | 1.01 |
| 3 | 10 | 2.508e-01 | 5.156e-01 | 2.506e-01 | 5.132e-01 | 1.00 | 1.00 |
| 4 | 10 | 2.508e-01 | 5.035e-01 | 2.506e-01 | 5.011e-01 | 1.00 | 1.00 |
| 5 | 10 | 2.507e-01 | 4.938e-01 | 2.506e-01 | 4.915e-01 | 1.00 | 1.00 |
| 6 | 10 | 2.508e-01 | 4.862e-01 | 2.506e-01 | 4.839e-01 | 1.00 | 1.00 |
| 7 | 10 | 2.509e-01 | 4.801e-01 | 2.506e-01 | 4.776e-01 | 1.00 | 1.01 |
| 8 | 10 | 2.509e-01 | 4.482e-01 | 2.506e-01 | 4.449e-01 | 1.00 | 1.01 |

**Table 6.8:** Test Case 1-6 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 3.347e-01 | 8.287e-01 | 3.340e-01 | 8.220e-01 | 1.00 | 1.01 |
| 1 | 10 | 3.347e-01 | 7.864e-01 | 3.340e-01 | 7.809e-01 | 1.00 | 1.01 |
| 2 | 10 | 3.348e-01 | 7.371e-01 | 3.340e-01 | 7.314e-01 | 1.00 | 1.01 |
| 3 | 10 | 3.350e-01 | 6.907e-01 | 3.340e-01 | 6.846e-01 | 1.00 | 1.01 |
| 4 | 10 | 3.349e-01 | 6.729e-01 | 3.340e-01 | 6.671e-01 | 1.00 | 1.01 |
| 5 | 10 | 3.349e-01 | 6.592e-01 | 3.340e-01 | 6.536e-01 | 1.00 | 1.01 |
| 6 | 10 | 3.350e-01 | 6.486e-01 | 3.340e-01 | 6.427e-01 | 1.00 | 1.01 |
| 7 | 10 | 3.349e-01 | 6.397e-01 | 3.340e-01 | 6.339e-01 | 1.00 | 1.01 |
| 8 | 10 | 3.349e-01 | 5.962e-01 | 3.340e-01 | 5.889e-01 | 1.00 | 1.01 |

**Table 6.9:** Test Case 1-7 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 1.726e-01 | 1.428e-01 | 1.726e-01 | 1.428e-01 | 1.00 | 1.00 |
| 1 | 10 | 1.726e-01 | 1.288e-01 | 1.726e-01 | 1.287e-01 | 1.00 | 1.00 |
| 2 | 10 | 1.726e-01 | 1.208e-01 | 1.726e-01 | 1.208e-01 | 1.00 | 1.00 |
| 3 | 10 | 1.727e-01 | 1.153e-01 | 1.726e-01 | 1.154e-01 | 1.00 | 1.00 |
| 4 | 10 | 1.726e-01 | 1.124e-01 | 1.726e-01 | 1.126e-01 | 1.00 | 1.00 |
| 5 | 10 | 1.727e-01 | 1.097e-01 | 1.726e-01 | 1.100e-01 | 1.00 | 1.00 |
| 6 | 10 | 1.727e-01 | 1.072e-01 | 1.726e-01 | 1.075e-01 | 1.00 | 1.00 |
| 7 | 10 | 1.727e-01 | 1.047e-01 | 1.726e-01 | 1.050e-01 | 1.00 | 1.00 |
| 8 | 10 | 1.726e-01 | 1.017e-01 | 1.726e-01 | 1.021e-01 | 1.00 | 1.00 |
| 9 | 10 | 1.727e-01 | 1.002e-01 | 1.726e-01 | 1.006e-01 | 1.00 | 1.00 |
| 10 | 10 | 1.727e-01 | 9.973e-02 | 1.726e-01 | 1.002e-01 | 1.00 | 1.00 |

**Table 6.10:** Test Case 1-8 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 5.210e-01 | 1.400e+00 | 5.198e-01 | 1.382e+00 | 1.00 | 1.01 |
| 1 | 10 | 5.210e-01 | 1.316e+00 | 5.198e-01 | 1.300e+00 | 1.00 | 1.01 |
| 2 | 10 | 5.211e-01 | 1.223e+00 | 5.198e-01 | 1.209e+00 | 1.00 | 1.01 |
| 3 | 10 | 5.212e-01 | 1.138e+00 | 5.198e-01 | 1.126e+00 | 1.00 | 1.01 |
| 4 | 10 | 5.211e-01 | 1.114e+00 | 5.198e-01 | 1.104e+00 | 1.00 | 1.01 |
| 5 | 10 | 5.210e-01 | 1.092e+00 | 5.198e-01 | 1.084e+00 | 1.00 | 1.01 |
| 6 | 10 | 5.210e-01 | 1.073e+00 | 5.198e-01 | 1.066e+00 | 1.00 | 1.01 |
| 7 | 10 | 5.209e-01 | 1.052e+00 | 5.198e-01 | 1.046e+00 | 1.00 | 1.01 |
| 8 | 10 | 5.212e-01 | 9.798e-01 | 5.198e-01 | 9.688e-01 | 1.00 | 1.01 |

**Table 6.11:** Test Case 1-9 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 1 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 2 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 3 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 4 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 5 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 6 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 7 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 8 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 9 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |
| 10 | 10 | 7.427e-02 | 1.943e-02 | 7.425e-02 | 1.943e-02 | 1.00 | 1.00 |

**Table 6.12:** Test Case 1-10 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 1.215e-01 | 3.388e-02 | 1.215e-01 | 3.389e-02 | 1.00 | 1.00 |
| 1 | 10 | 1.215e-01 | 3.097e-02 | 1.215e-01 | 3.097e-02 | 1.00 | 1.00 |
| 2 | 10 | 1.215e-01 | 3.038e-02 | 1.215e-01 | 3.016e-02 | 1.00 | 1.01 |
| 3 | 10 | 1.215e-01 | 3.030e-02 | 1.215e-01 | 2.992e-02 | 1.00 | 1.01 |
| 4 | 10 | 1.215e-01 | 3.029e-02 | 1.215e-01 | 2.979e-02 | 1.00 | 1.02 |
| 5 | 10 | 1.215e-01 | 3.026e-02 | 1.215e-01 | 2.971e-02 | 1.00 | 1.02 |
| 6 | 10 | 1.215e-01 | 3.022e-02 | 1.215e-01 | 2.962e-02 | 1.00 | 1.02 |
| 7 | 10 | 1.215e-01 | 3.015e-02 | 1.215e-01 | 2.951e-02 | 1.00 | 1.02 |
| 8 | 10 | 1.215e-01 | 2.998e-02 | 1.215e-01 | 2.927e-02 | 1.00 | 1.02 |
| 9 | 10 | 1.215e-01 | 2.978e-02 | 1.215e-01 | 2.897e-02 | 1.00 | 1.03 |
| 10 | 10 | 1.215e-01 | 2.949e-02 | 1.215e-01 | 2.861e-02 | 1.00 | 1.03 |

**Table 6.13:** Test Case 1-11 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 1.215e-01 | 3.459e-02 | 1.215e-01 | 3.462e-02 | 1.00 | 1.00 |
| 1 | 10 | 1.215e-01 | 3.163e-02 | 1.215e-01 | 3.165e-02 | 1.00 | 1.00 |
| 2 | 10 | 1.215e-01 | 3.105e-02 | 1.215e-01 | 3.082e-02 | 1.00 | 1.01 |
| 3 | 10 | 1.215e-01 | 3.101e-02 | 1.215e-01 | 3.061e-02 | 1.00 | 1.01 |
| 4 | 10 | 1.215e-01 | 3.113e-02 | 1.215e-01 | 3.060e-02 | 1.00 | 1.02 |
| 5 | 10 | 1.215e-01 | 3.210e-02 | 1.215e-01 | 3.138e-02 | 1.00 | 1.02 |
| 6 | 10 | 1.215e-01 | 3.173e-02 | 1.215e-01 | 3.102e-02 | 1.00 | 1.02 |
| 7 | 10 | 1.215e-01 | 3.096e-02 | 1.215e-01 | 3.033e-02 | 1.00 | 1.02 |
| 8 | 10 | 1.215e-01 | 3.026e-02 | 1.215e-01 | 2.961e-02 | 1.00 | 1.02 |
| 9 | 10 | 1.215e-01 | 3.031e-02 | 1.215e-01 | 2.950e-02 | 1.00 | 1.03 |
| 10 | 10 | 1.215e-01 | 3.003e-02 | 1.215e-01 | 2.913e-02 | 1.00 | 1.03 |

**Table 6.14:** Test Case 1-12 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 0 | 10 | 1.215e-01 | 2.891e-02 | 1.215e-01 | 2.891e-02 | 1.00 | 1.00 |
| 1 | 10 | 1.215e-01 | 2.638e-02 | 1.215e-01 | 2.638e-02 | 1.00 | 1.00 |
| 2 | 10 | 1.215e-01 | 2.641e-02 | 1.215e-01 | 2.625e-02 | 1.00 | 1.01 |
| 3 | 10 | 1.215e-01 | 2.702e-02 | 1.215e-01 | 2.669e-02 | 1.00 | 1.01 |
| 4 | 10 | 1.215e-01 | 2.724e-02 | 1.215e-01 | 2.687e-02 | 1.00 | 1.01 |
| 5 | 10 | 1.215e-01 | 2.879e-02 | 1.215e-01 | 2.811e-02 | 1.00 | 1.02 |
| 6 | 10 | 1.215e-01 | 2.791e-02 | 1.215e-01 | 2.749e-02 | 1.00 | 1.02 |
| 7 | 10 | 1.215e-01 | 2.742e-02 | 1.215e-01 | 2.711e-02 | 1.00 | 1.01 |
| 8 | 10 | 1.216e-01 | 2.716e-02 | 1.215e-01 | 2.670e-02 | 1.00 | 1.02 |
| 9 | 10 | 1.215e-01 | 2.667e-02 | 1.215e-01 | 2.607e-02 | 1.00 | 1.02 |
| 10 | 10 | 1.215e-01 | 2.614e-02 | 1.215e-01 | 2.552e-02 | 1.00 | 1.02 |

**Table 6.15:** Test Case 1-13 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 6 | 6 | 2.989e-01 | 2.453e-01 | 2.989e-01 | 2.451e-01 | 1.00 | 1.00 |
| 5 | 6 | 2.989e-01 | 2.487e-01 | 2.989e-01 | 2.485e-01 | 1.00 | 1.00 |
| 4 | 6 | 2.989e-01 | 2.548e-01 | 2.989e-01 | 2.546e-01 | 1.00 | 1.00 |
| 3 | 6 | 2.989e-01 | 2.636e-01 | 2.989e-01 | 2.635e-01 | 1.00 | 1.00 |
| 2 | 6 | 2.990e-01 | 2.759e-01 | 2.989e-01 | 2.757e-01 | 1.00 | 1.00 |
| 1 | 6 | 2.989e-01 | 2.928e-01 | 2.989e-01 | 2.927e-01 | 1.00 | 1.00 |
| 0 | 6 | 2.988e-01 | 3.209e-01 | 2.989e-01 | 3.212e-01 | 1.00 | 1.00 |
| 6 | 7 | 2.989e-01 | 2.514e-01 | 2.989e-01 | 2.512e-01 | 1.00 | 1.00 |
| 5 | 7 | 2.989e-01 | 2.556e-01 | 2.989e-01 | 2.556e-01 | 1.00 | 1.00 |
| 4 | 7 | 2.989e-01 | 2.624e-01 | 2.989e-01 | 2.624e-01 | 1.00 | 1.00 |
| 3 | 7 | 2.989e-01 | 2.720e-01 | 2.989e-01 | 2.720e-01 | 1.00 | 1.00 |
| 2 | 7 | 2.990e-01 | 2.851e-01 | 2.989e-01 | 2.850e-01 | 1.00 | 1.00 |
| 1 | 7 | 2.989e-01 | 3.027e-01 | 2.989e-01 | 3.029e-01 | 1.00 | 1.00 |
| 0 | 7 | 2.988e-01 | 3.319e-01 | 2.989e-01 | 3.324e-01 | 1.00 | 1.00 |
| 6 | 8 | 2.989e-01 | 2.569e-01 | 2.989e-01 | 2.568e-01 | 1.00 | 1.00 |
| 5 | 8 | 2.989e-01 | 2.619e-01 | 2.989e-01 | 2.618e-01 | 1.00 | 1.00 |
| 4 | 8 | 2.989e-01 | 2.693e-01 | 2.989e-01 | 2.692e-01 | 1.00 | 1.00 |
| 3 | 8 | 2.989e-01 | 2.796e-01 | 2.989e-01 | 2.795e-01 | 1.00 | 1.00 |
| 2 | 8 | 2.990e-01 | 2.933e-01 | 2.989e-01 | 2.932e-01 | 1.00 | 1.00 |
| 1 | 8 | 2.989e-01 | 3.117e-01 | 2.989e-01 | 3.118e-01 | 1.00 | 1.00 |
| 0 | 8 | 2.987e-01 | 3.416e-01 | 2.989e-01 | 3.422e-01 | 1.00 | 1.00 |
| 6 | 9 | 2.989e-01 | 2.621e-01 | 2.989e-01 | 2.620e-01 | 1.00 | 1.00 |
| 5 | 9 | 2.989e-01 | 2.674e-01 | 2.989e-01 | 2.675e-01 | 1.00 | 1.00 |
| 4 | 9 | 2.989e-01 | 2.753e-01 | 2.989e-01 | 2.754e-01 | 1.00 | 1.00 |
| 3 | 9 | 2.989e-01 | 2.861e-01 | 2.989e-01 | 2.862e-01 | 1.00 | 1.00 |
| 2 | 9 | 2.990e-01 | 3.005e-01 | 2.989e-01 | 3.004e-01 | 1.00 | 1.00 |
| 1 | 9 | 2.989e-01 | 3.194e-01 | 2.989e-01 | 3.196e-01 | 1.00 | 1.00 |
| 0 | 9 | 2.987e-01 | 3.500e-01 | 2.989e-01 | 3.508e-01 | 1.00 | 1.00 |
| 6 | 10 | 2.989e-01 | 2.669e-01 | 2.989e-01 | 2.666e-01 | 1.00 | 1.00 |
| 5 | 10 | 2.989e-01 | 2.726e-01 | 2.989e-01 | 2.725e-01 | 1.00 | 1.00 |
| 4 | 10 | 2.989e-01 | 2.809e-01 | 2.989e-01 | 2.809e-01 | 1.00 | 1.00 |
| 3 | 10 | 2.989e-01 | 2.921e-01 | 2.989e-01 | 2.921e-01 | 1.00 | 1.00 |
| 2 | 10 | 2.990e-01 | 3.069e-01 | 2.989e-01 | 3.068e-01 | 1.00 | 1.00 |
| 1 | 10 | 2.989e-01 | 3.263e-01 | 2.989e-01 | 3.265e-01 | 1.00 | 1.00 |

**Table 6.16:** Test Case 1-14 Mean and Variance Results

| $x_{DX}$ [cm] | | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|---|
| min. | max. | Mean | Variance | Mean | Variance | Mean | Variance |
| 6 | 8 | 5.187e-01 | 1.038e+00 | 5.185e-01 | 1.031e+00 | 1.00 | 1.01 |
| 5 | 8 | 5.187e-01 | 1.095e+00 | 5.185e-01 | 1.089e+00 | 1.00 | 1.01 |
| 4 | 8 | 5.188e-01 | 1.153e+00 | 5.185e-01 | 1.146e+00 | 1.00 | 1.01 |
| 3 | 8 | 5.187e-01 | 1.218e+00 | 5.185e-01 | 1.210e+00 | 1.00 | 1.01 |
| 2 | 8 | 5.191e-01 | 1.293e+00 | 5.185e-01 | 1.285e+00 | 1.00 | 1.01 |
| 1 | 8 | 5.189e-01 | 1.382e+00 | 5.185e-01 | 1.375e+00 | 1.00 | 1.01 |
| 0 | 8 | 5.187e-01 | 1.515e+00 | 5.185e-01 | 1.507e+00 | 1.00 | 1.00 |
| 6 | 9 | 5.188e-01 | 1.091e+00 | 5.185e-01 | 1.084e+00 | 1.00 | 1.01 |
| 5 | 9 | 5.189e-01 | 1.154e+00 | 5.185e-01 | 1.147e+00 | 1.00 | 1.01 |
| 4 | 9 | 5.189e-01 | 1.215e+00 | 5.185e-01 | 1.209e+00 | 1.00 | 1.01 |
| 3 | 9 | 5.188e-01 | 1.285e+00 | 5.185e-01 | 1.278e+00 | 1.00 | 1.01 |
| 2 | 9 | 5.191e-01 | 1.366e+00 | 5.185e-01 | 1.357e+00 | 1.00 | 1.01 |
| 1 | 9 | 5.189e-01 | 1.460e+00 | 5.185e-01 | 1.453e+00 | 1.00 | 1.00 |
| 0 | 9 | 5.189e-01 | 1.600e+00 | 5.185e-01 | 1.592e+00 | 1.00 | 1.00 |
| 6 | 10 | 5.187e-01 | 1.124e+00 | 5.185e-01 | 1.118e+00 | 1.00 | 1.01 |
| 5 | 10 | 5.189e-01 | 1.191e+00 | 5.185e-01 | 1.184e+00 | 1.00 | 1.01 |
| 4 | 10 | 5.189e-01 | 1.256e+00 | 5.185e-01 | 1.249e+00 | 1.00 | 1.01 |
| 3 | 10 | 5.187e-01 | 1.328e+00 | 5.185e-01 | 1.322e+00 | 1.00 | 1.01 |
| 2 | 10 | 5.191e-01 | 1.413e+00 | 5.185e-01 | 1.404e+00 | 1.00 | 1.01 |
| 1 | 10 | 5.189e-01 | 1.511e+00 | 5.185e-01 | 1.504e+00 | 1.00 | 1.00 |

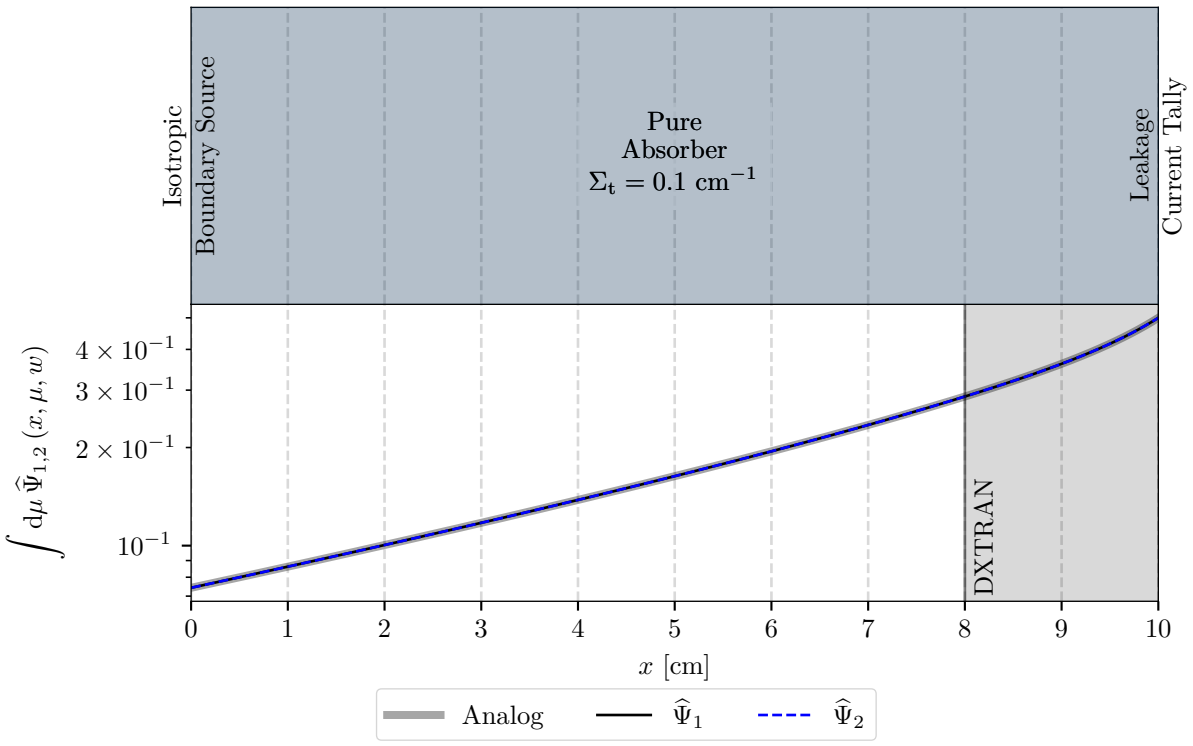



**Figure 6.14:** Test Case 1-1 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

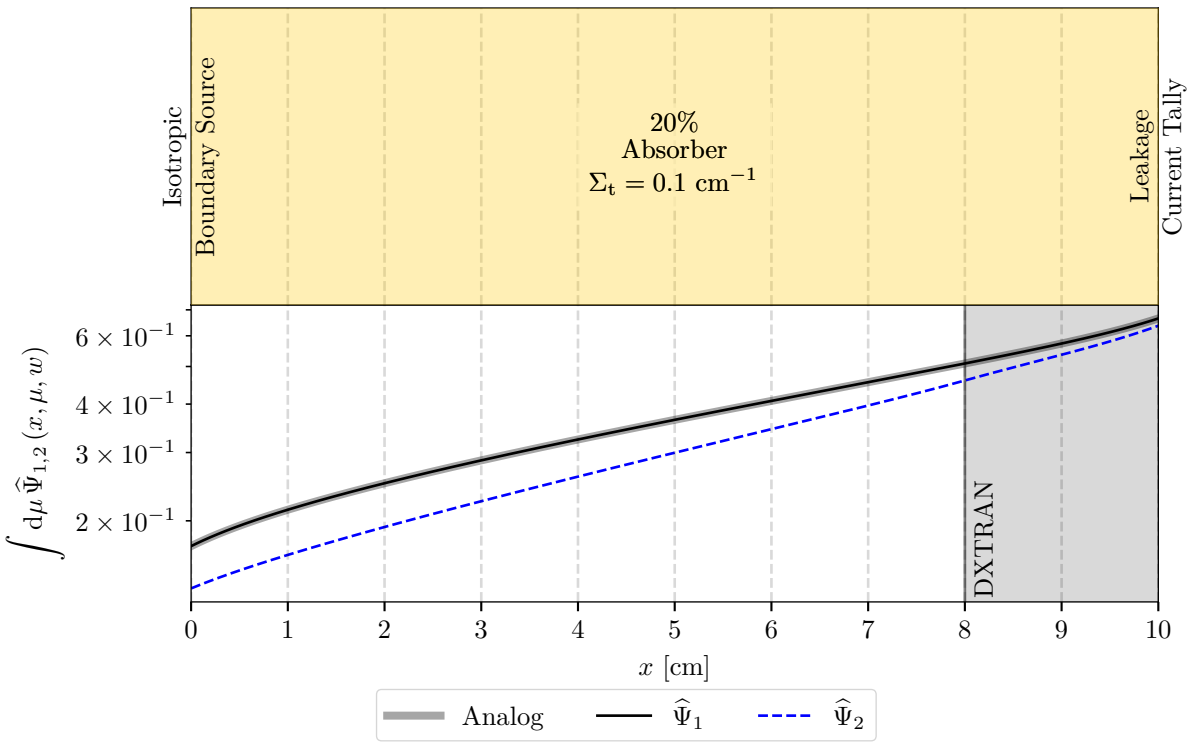



**Figure 6.15:** Test Case 1-2 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

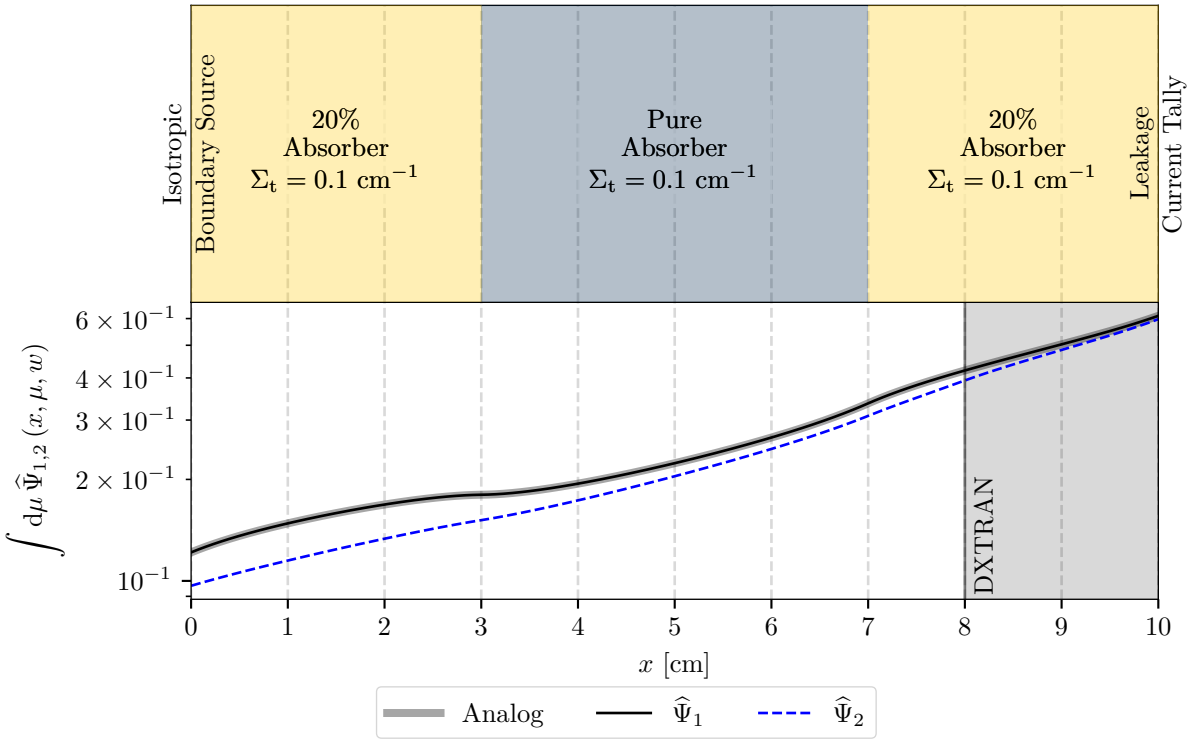



**Figure 6.16:** Test Case 1-3 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

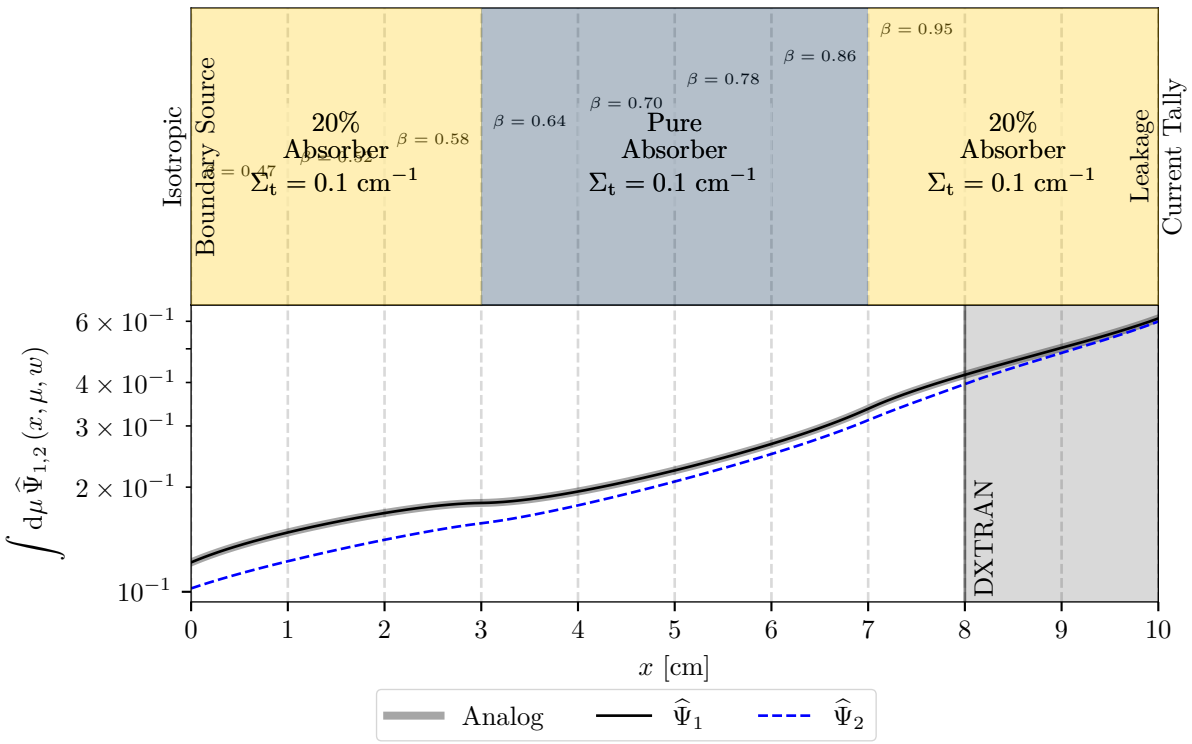



**Figure 6.17:** Test Case 1-4 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

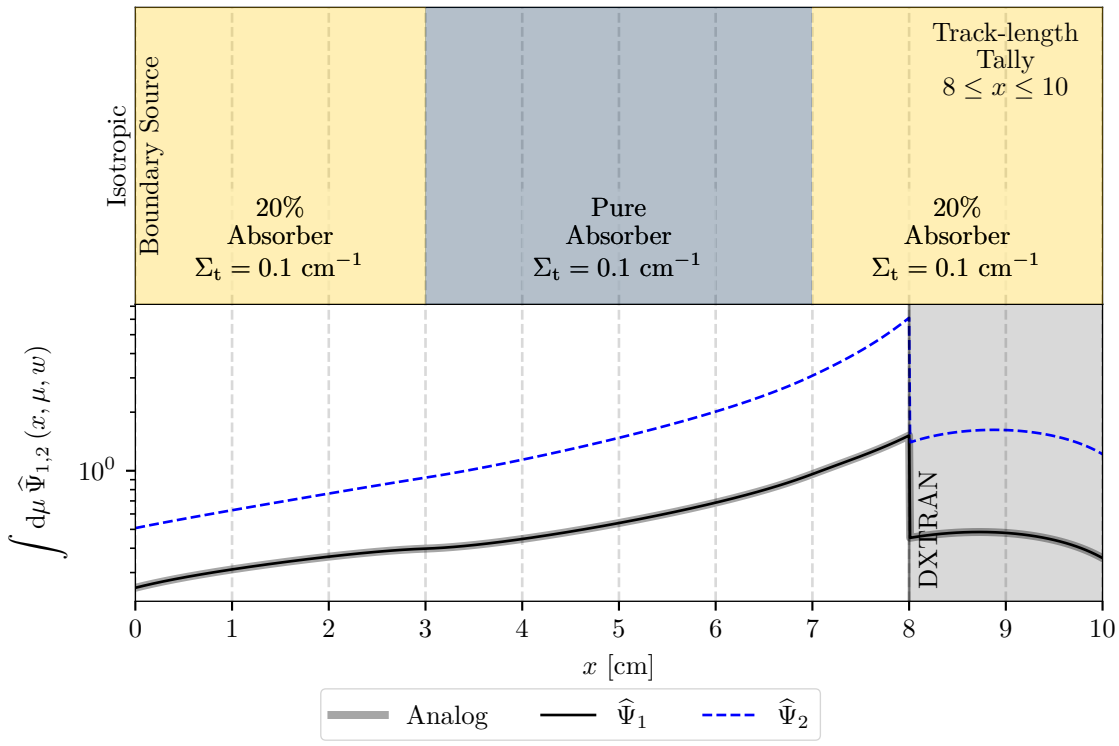



**Figure 6.18:** Test Case 1-5 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

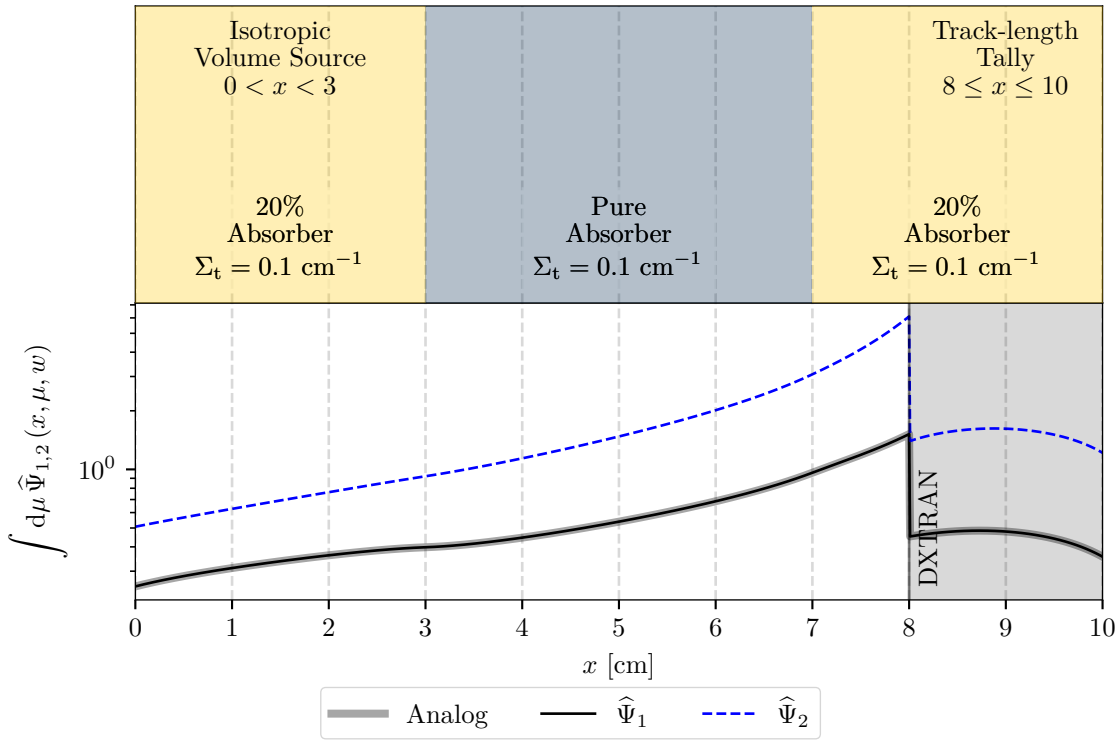



**Figure 6.19:** Test Case 1-6 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

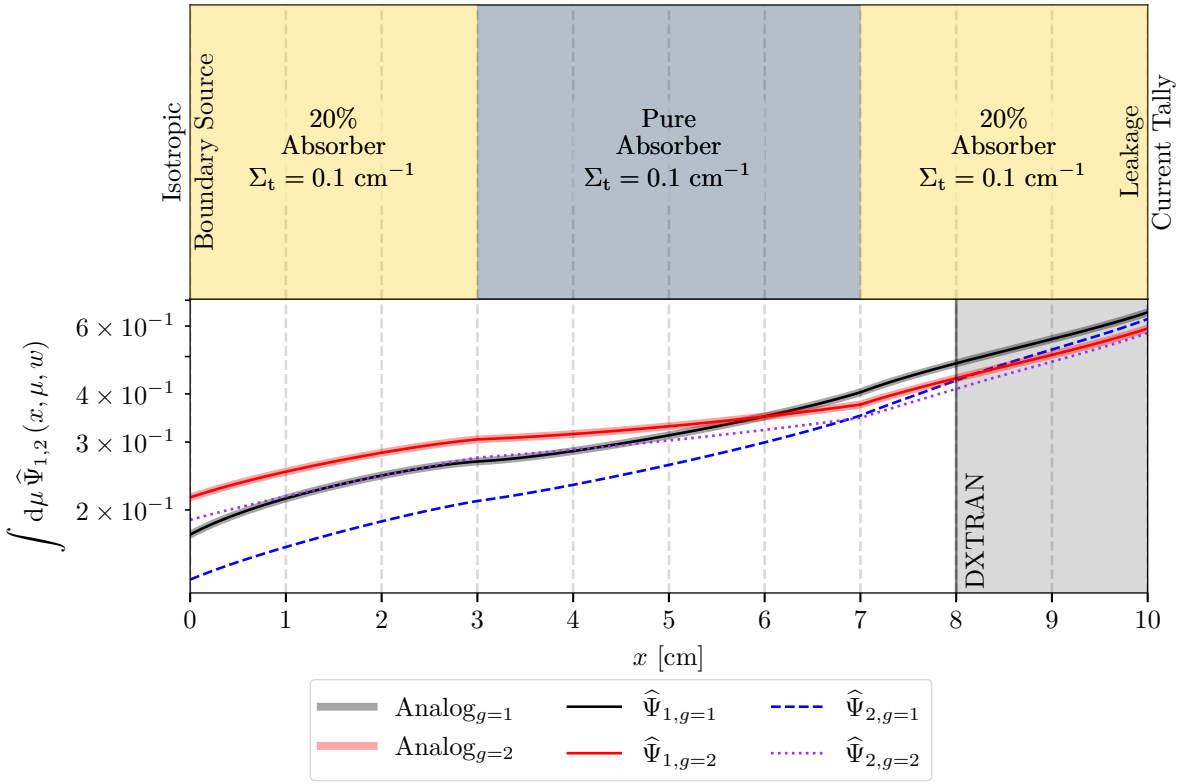



**Figure 6.20:** Test Case 1-7 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

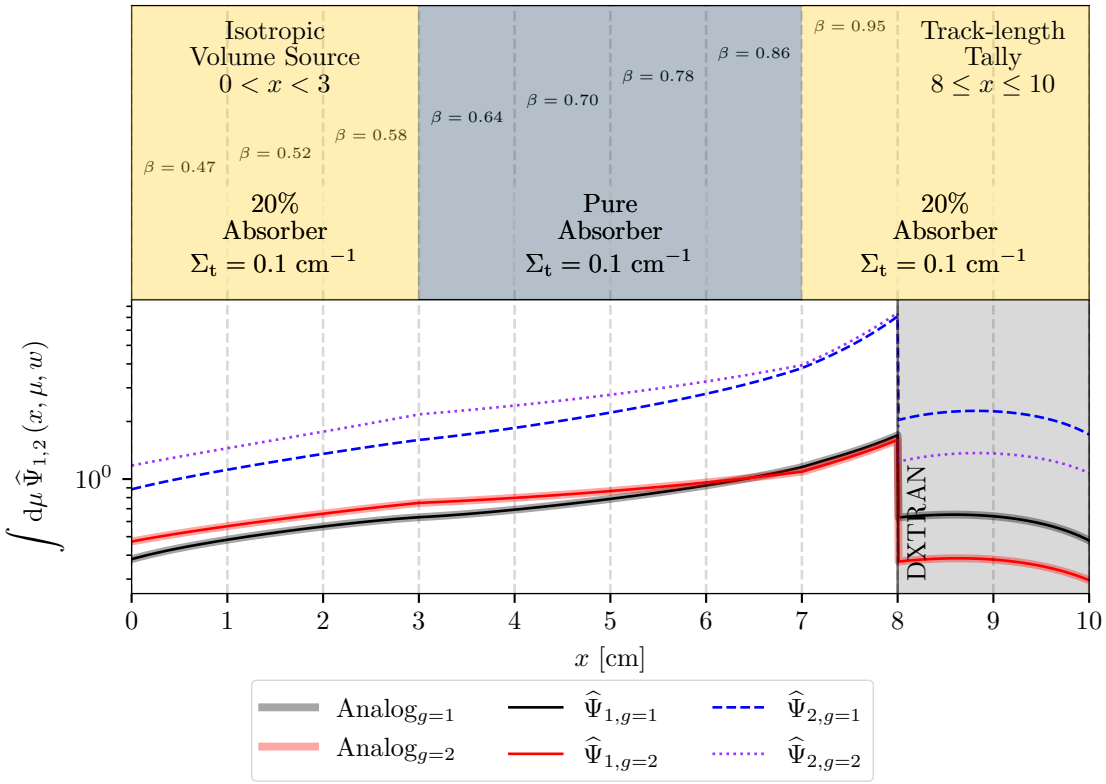



**Figure 6.21:** Test Case 1-8 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

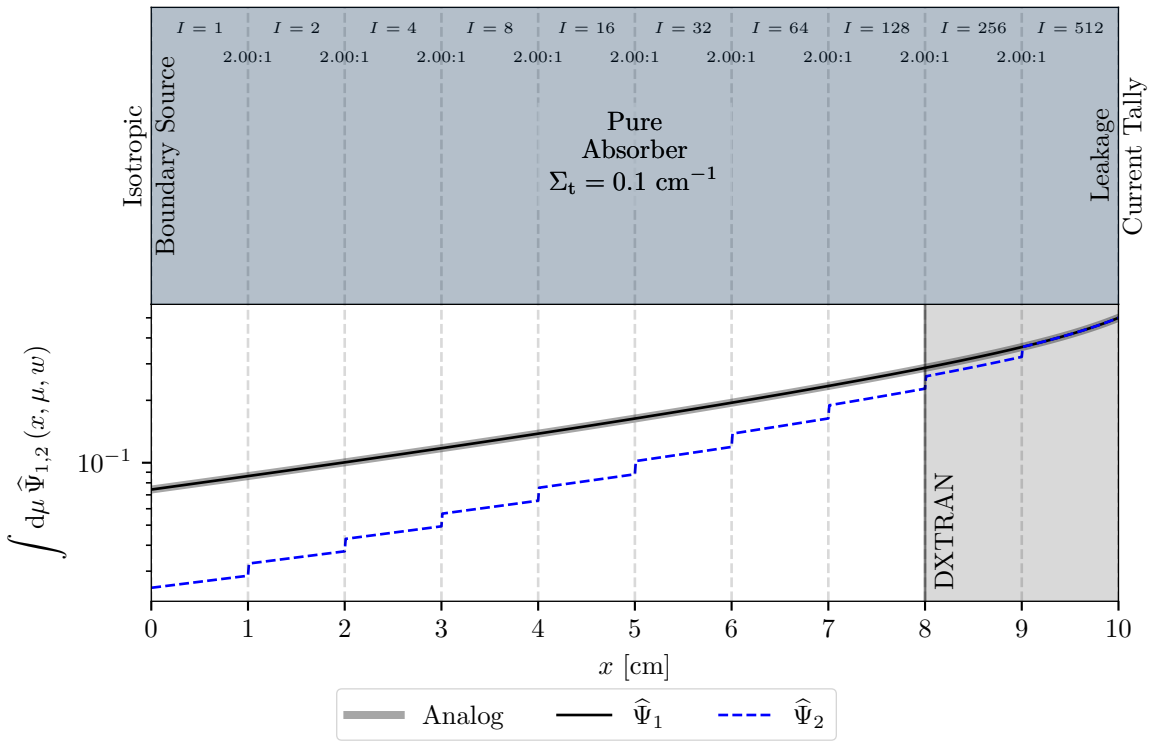



**Figure 6.22:** Test Case 1-9 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

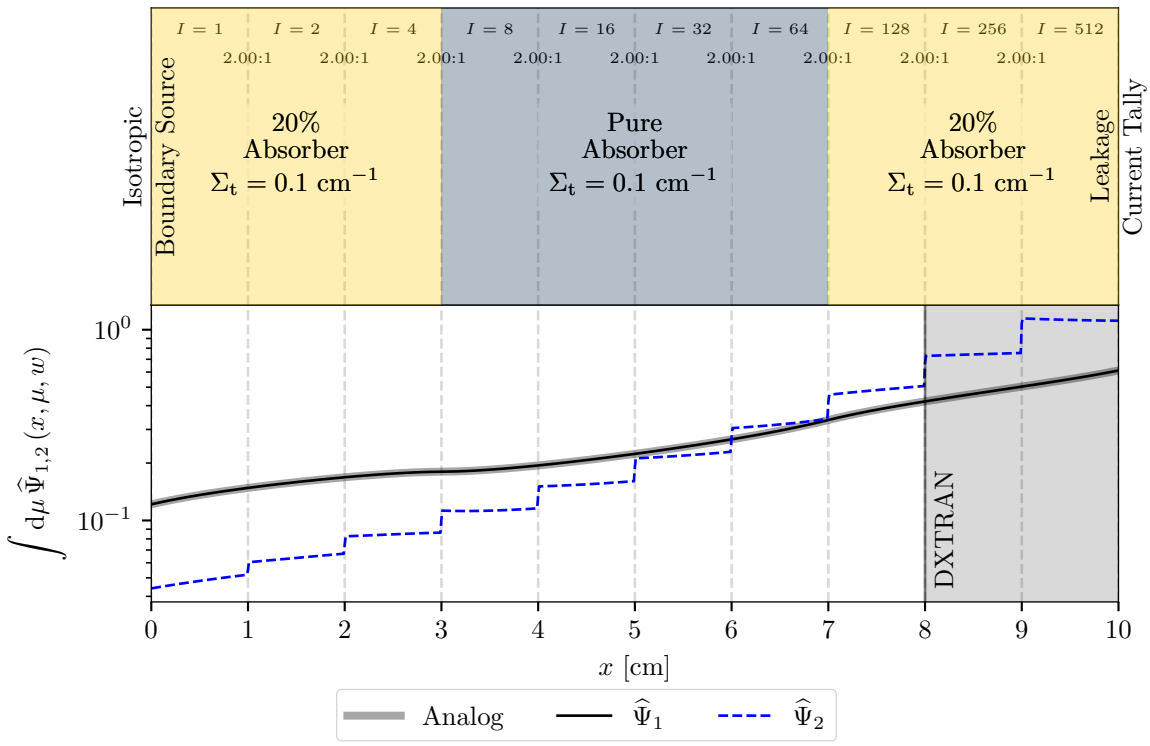



**Figure 6.23:** Test Case 1-10 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

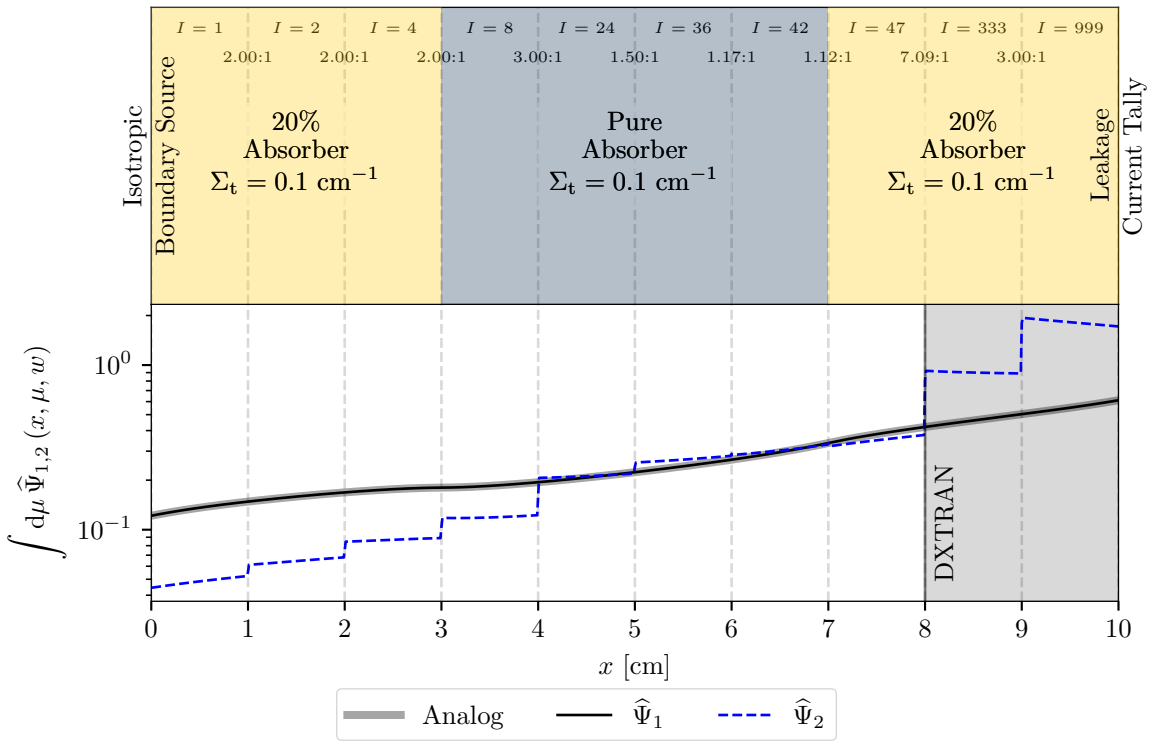



**Figure 6.24:** Test Case 1-11 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

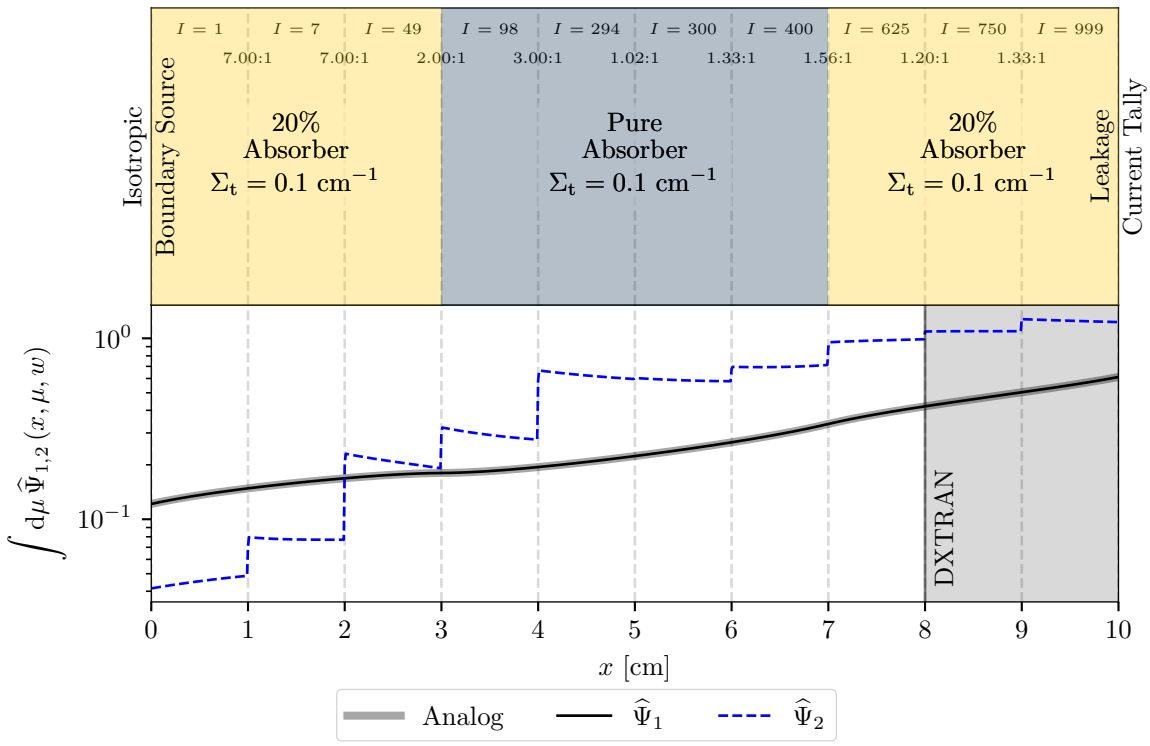



**Figure 6.25:** Test Case 1-12 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $8 < x < 10$

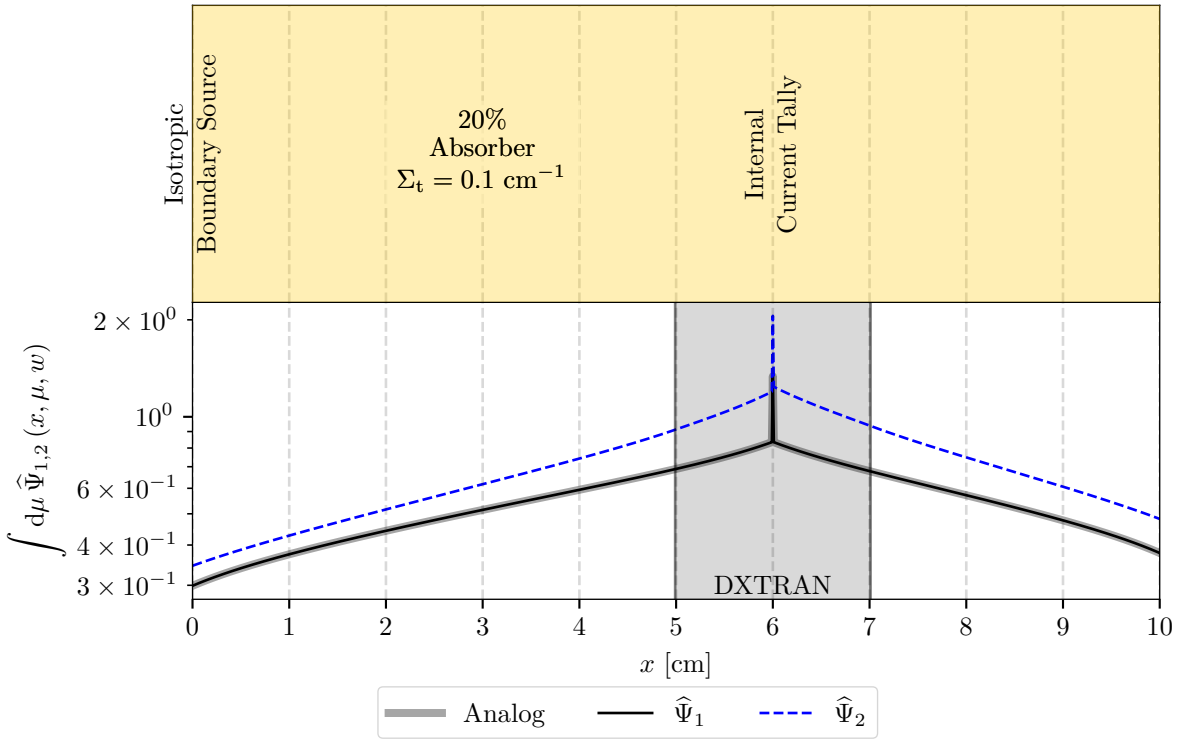



**Figure 6.26:** Test Case 1-13 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $5 < x < 7$

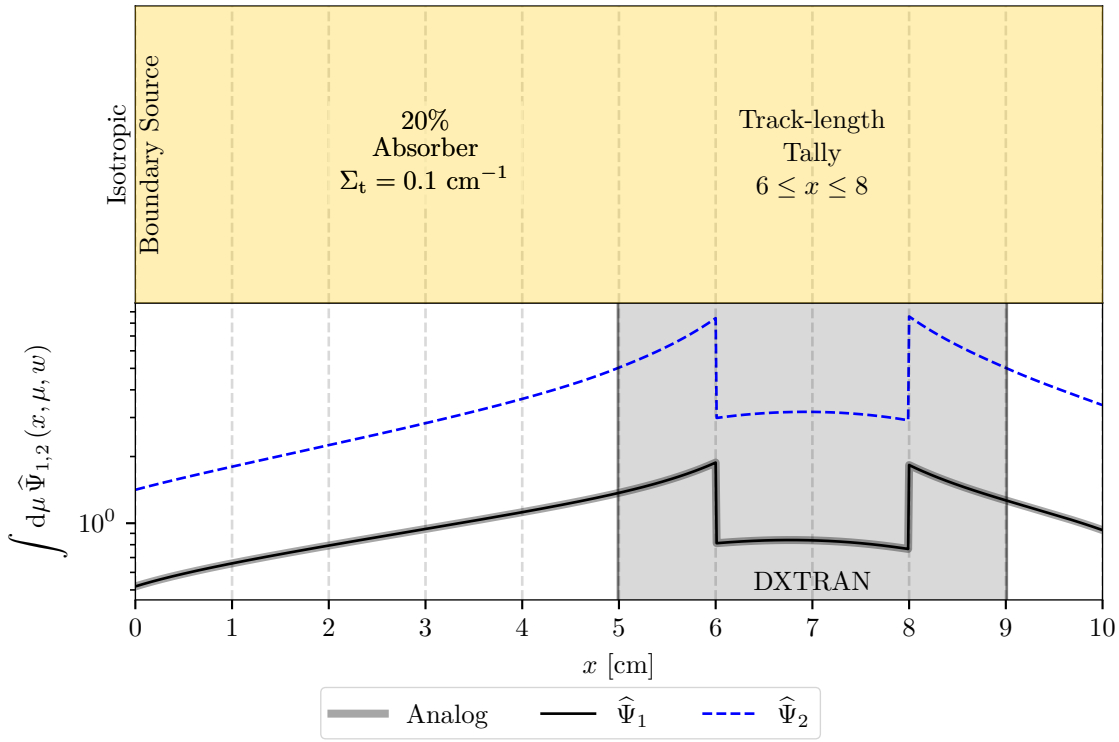



**Figure 6.27:** Test Case 1-14 $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ for a DXTRAN Region in $5 < x < 9$

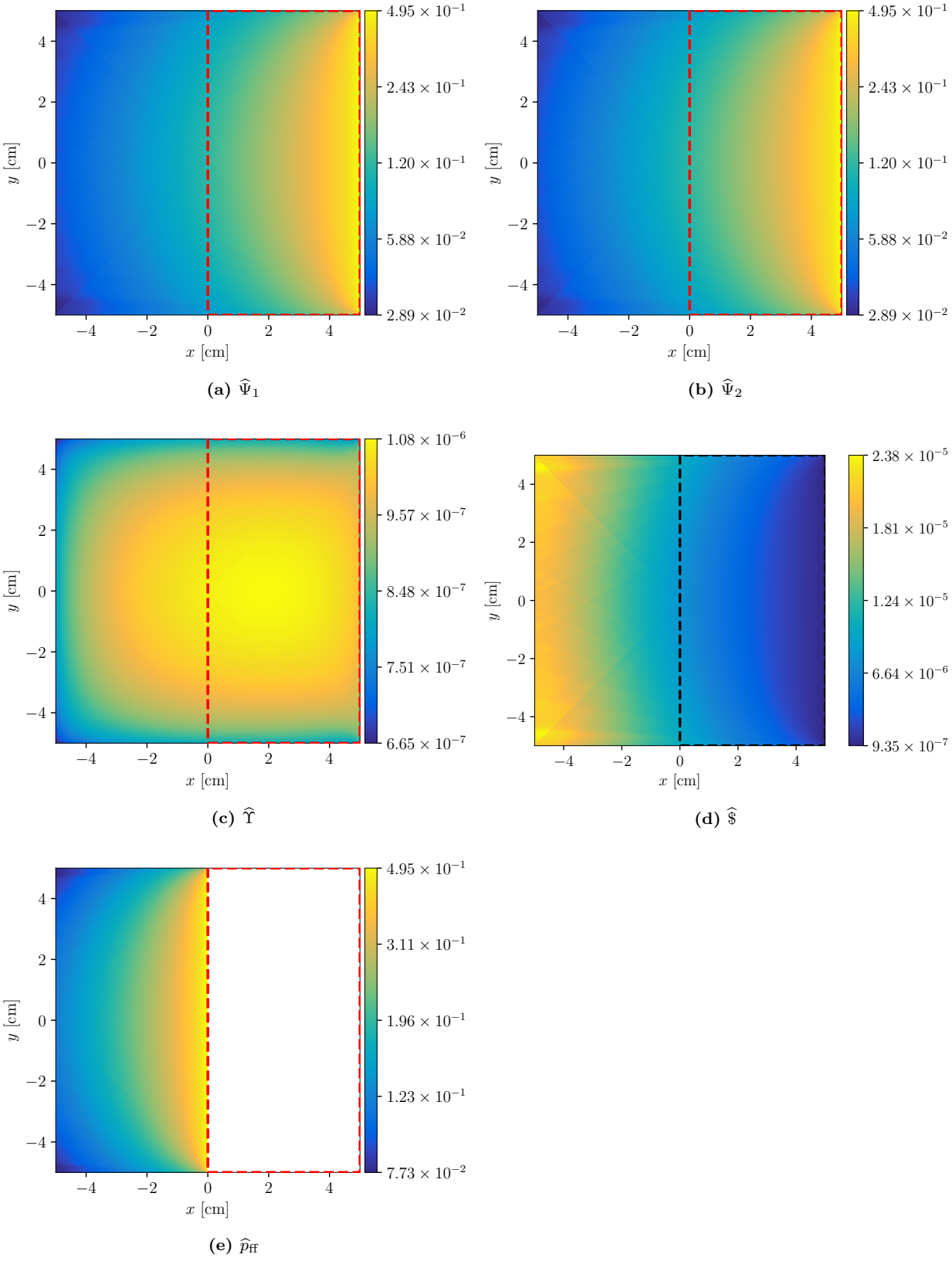
### 6.5 Two-dimensional Test Case Results

Each 2-D test case is executed using MCNP6 and COVRT. The MCNP6 sample mean ($\mu_{MC}$) and relative standard fractional uncertainty in the sample mean ($R_{\mathrm{MC}}$) values are retrieved and the Monte Carlo sample variance is calculated with Eq. (2.80). The MCNP6 sample mean and sample variance results are then compared with the COVRT-produced values by dividing the MCNP6 values by the COVRT values. The resulting mean and variance ratios for each 2-D test case is given in Table 6.17. In addition, figures showing representative scalar $\widehat{\Psi}_1(x,y)$, $\widehat{\Psi}_2(x,y)$, $\widehat{\Upsilon}(x,y)$, $\widehat{\$}(x,y)$, $p_{\mathrm{ff}}(x,y)$, and $\widehat{Q}_2(x,y)$ traverses for each test case are given in Figures 6.28–6.34.

In general, the level of agreement between the MCNP and COVRT results for the 2-D test cases is worse than in the 1-D test cases; however, it is comparable to previous work [58]. For Test Cases 2-1, 2-2, 2-4, and 2-6 the population variance predicted by COVRT agrees with the MCNP value within 3% when DXTRAN is used. However, when DXTRAN is used the population variance in Test Case 2-3 disagrees by 7%, Test Case 2-5 disagrees by 13%, and Test Case 2-7 disagrees by 11%. Note that in each of these test cases there is 1–3% disagreement in the analog cases. This remaining disagreement is attributable to angular discretization. When a finer quadrature set, and correspondingly finer spatial mesh, is used the mean and variance results improve. Unfortunately, by virtue of these test cases exhibiting a high degree of angle dependence a fine quadrature set is required to adequately represent the angle dependence. To obtain the results shown in Table 6.17 Test Cases 2-1 and 2-2 use an $S_{32}$ quadrature set with 544 angles in a hemisphere is used and the remaining 2-D test cases use an $S_{64}$ quadrature set with 2112 angles in a hemisphere. However, the level of agreement observed between the COVRT and MCNP results is sufficient to capture the behavior to enable optimization. Moreover, as described Chapter 7, a far more coarse quadrature is adequate for the optimizer to identify superior DXTRAN parameters versus unoptimized cases.

**Table 6.17:** 2-D Test Case Mean and Variance Results Comparison

| Test Case | MCNP6 | | COVRT | | MCNP6/COVRT | |
|---|---|---|---|---|---|---|
| | Mean | Variance | Mean | Variance | Mean | Variance |
| 2-1 | 3.554e-02 | 3.428e-02 | 3.540e-02 | 3.415e-02 | 1.00 | 1.00 |
| | 3.554e-02 | 3.428e-02 | 3.540e-02 | 3.415e-02 | 1.00 | 1.00 |
| 2-2 | 6.331e-02 | 5.930e-02 | 6.315e-02 | 5.916e-02 | 1.00 | 1.00 |
| | 6.330e-02 | 4.689e-02 | 6.315e-02 | 4.622e-02 | 1.00 | 1.01 |
| 2-3 | 8.273e-04 | 5.815e-04 | 8.202e-04 | 5.725e-04 | 1.01 | 1.02 |
| | 8.201e-04 | 1.319e-04 | 8.202e-04 | 1.238e-04 | 1.00 | 1.07 |
| 2-4 | 5.418e-03 | 5.388e-03 | 5.583e-03 | 5.552e-03 | 0.97 | 0.97 |
| | 5.450e-03 | 1.062e-03 | 5.583e-03 | 1.052e-03 | 0.98 | 1.01 |
| 2-5 | 1.392e-03 | 1.390e-03 | 1.439e-03 | 1.437e-03 | 0.97 | 0.97 |
| | 1.379e-03 | 2.952e-04 | 1.439e-03 | 3.387e-04 | 0.96 | 0.87 |
| 2-6 | 5.418e-03 | 5.388e-03 | 5.583e-03 | 5.552e-03 | 0.97 | 0.97 |
| | 5.413e-03 | 3.262e-03 | 5.583e-03 | 3.377e-03 | 0.97 | 0.97 |
| 2-7 | 1.856e-03 | 1.852e-03 | 1.867e-03 | 1.864e-03 | 0.99 | 0.99 |
| | 1.868e-03 | 6.415e-04 | 1.867e-03 | 5.778e-04 | 1.00 | 1.11 |




**(a)** $\widehat{\Psi}_1$

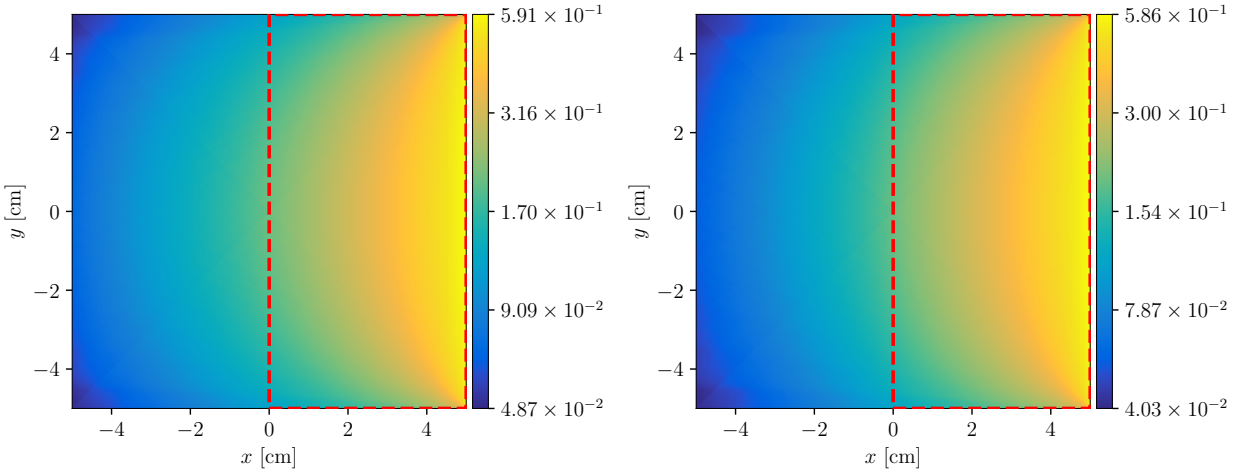
**(b)** $\widehat{\Psi}_2$

**(c)** $\widehat{\Upsilon}$

**(d)** $\widehat{\$}$

**(e)** $\widehat{p}_{\text{ff}}$

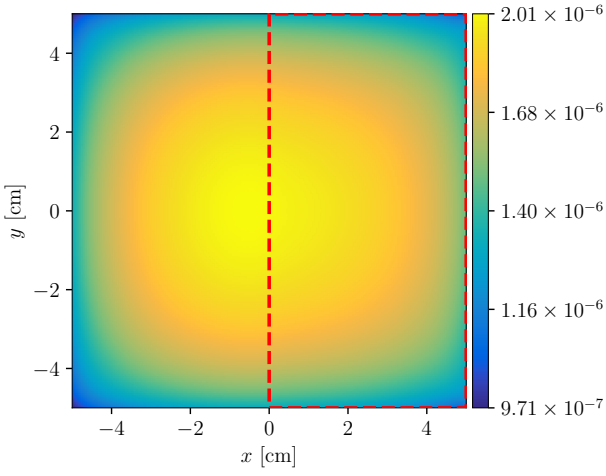
**Figure 6.28:** Test Case 2-1 Results



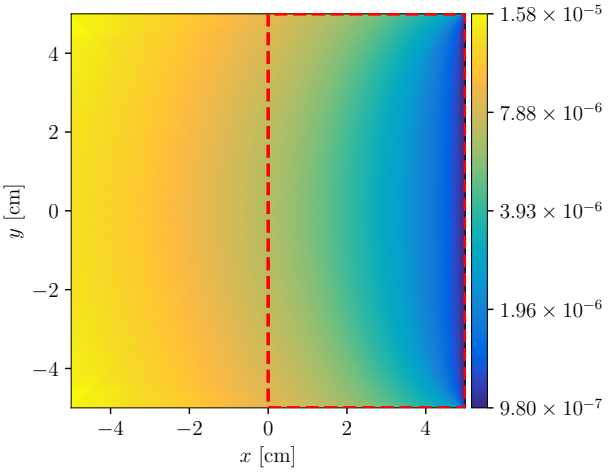

**(a)** $\widehat{\Psi}_1$



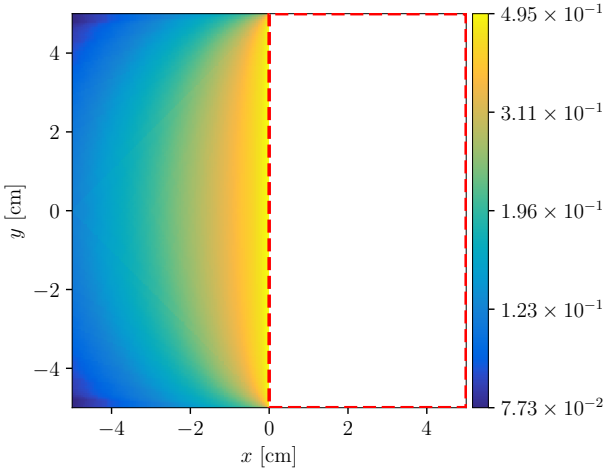

**(b)** $\widehat{\Psi}_2$



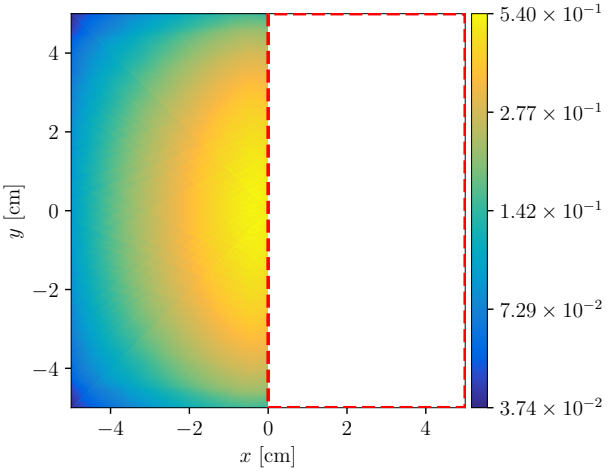

**(c)** $\widehat{\Upsilon}$




**(d)** $\widehat{\$}$

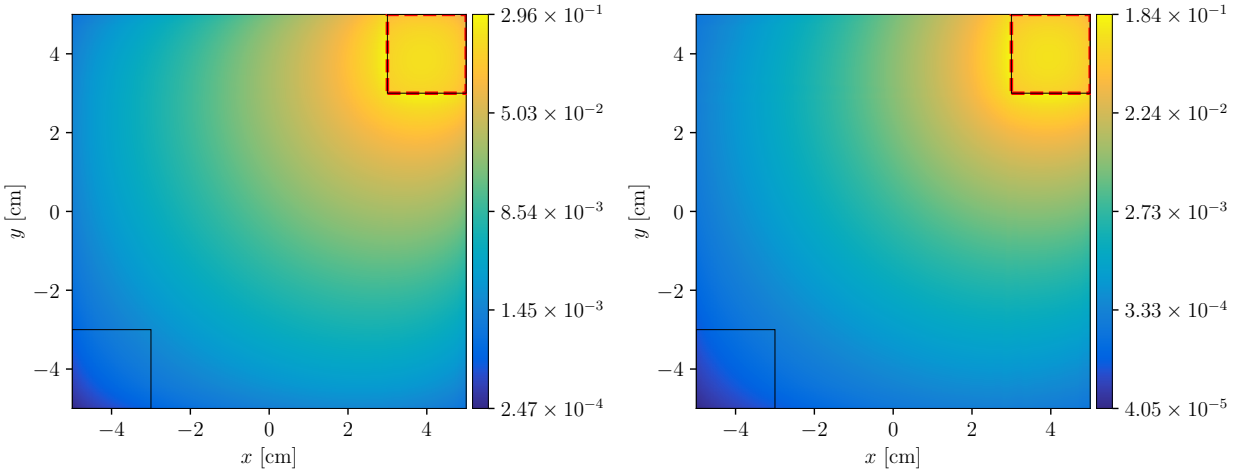



**(e)** $\widehat{p}_{\text{ff}}$




**(f)** $\widehat{Q}_2$

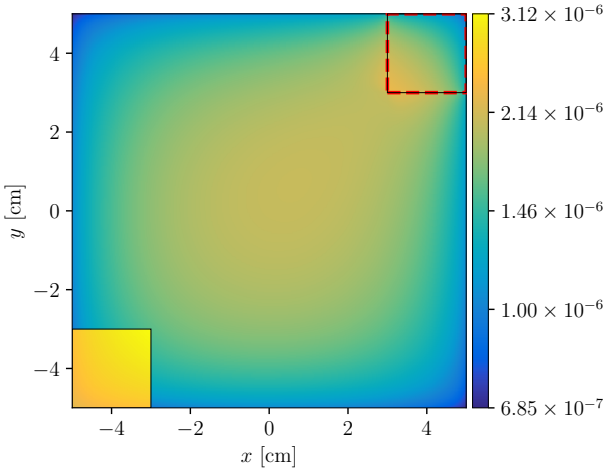
**Figure 6.29:** Test Case 2-2 Results



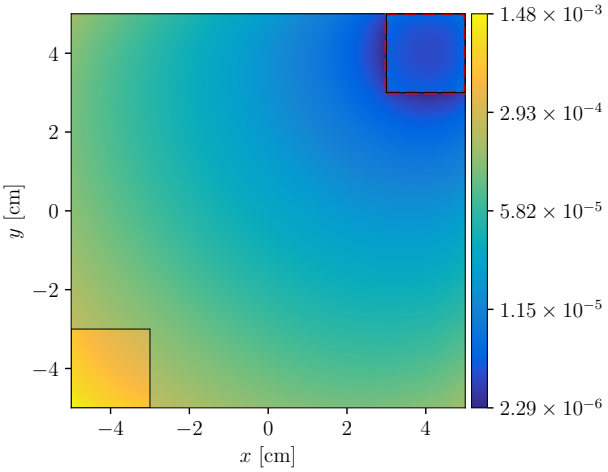

**(a)** $\widehat{\Psi}_1$



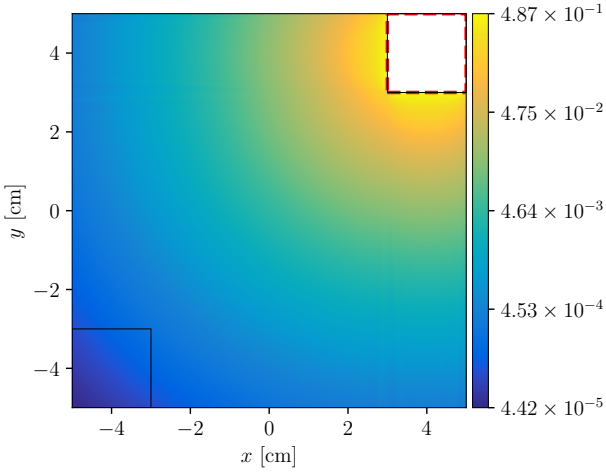

**(b)** $\widehat{\Psi}_2$



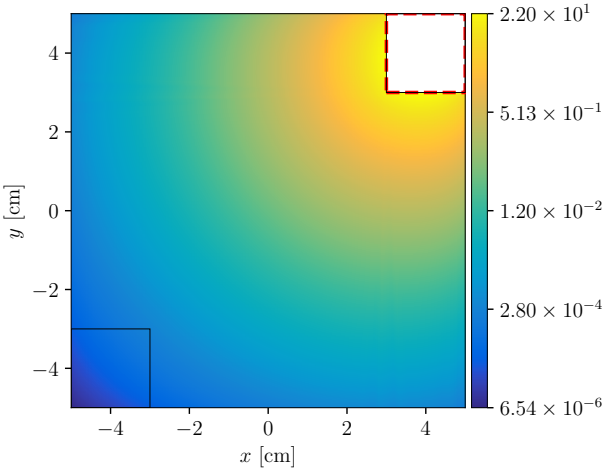

**(c)** $\widehat{\Upsilon}$




**(d)** $\widehat{\$}$

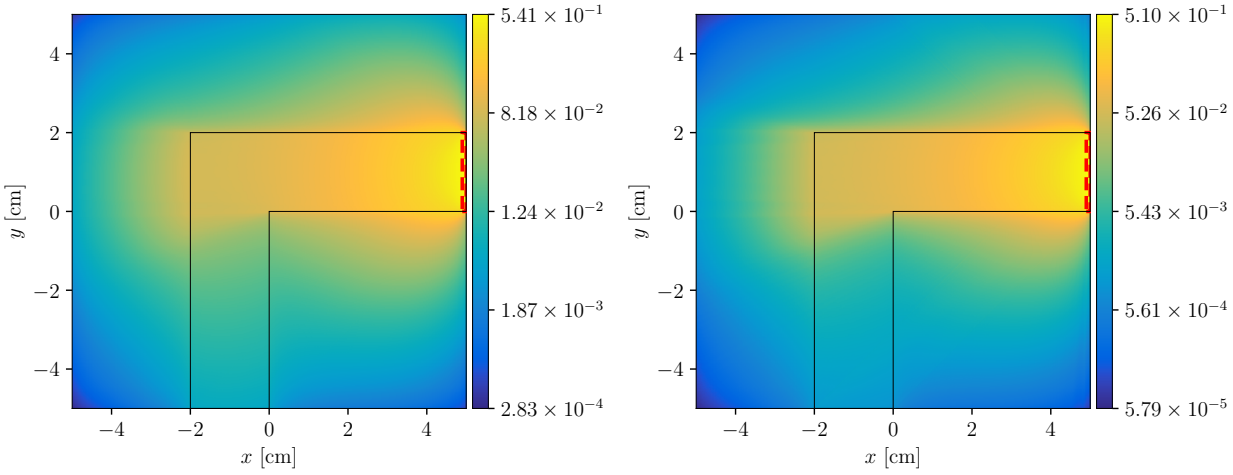



**(e)** $\widehat{p}_{\text{ff}}$




**(f)** $\widehat{Q}_2$

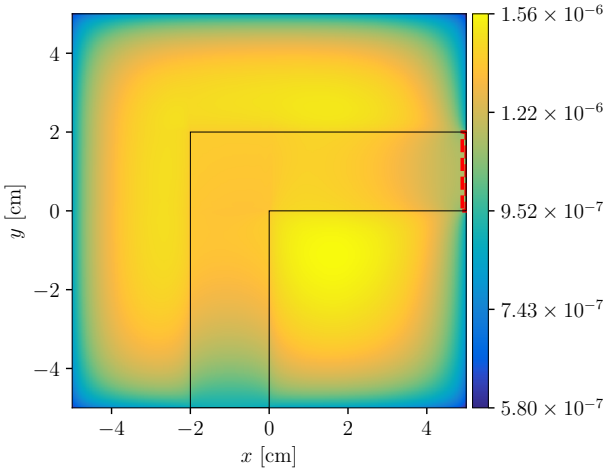
**Figure 6.30:** Test Case 2-3 Results



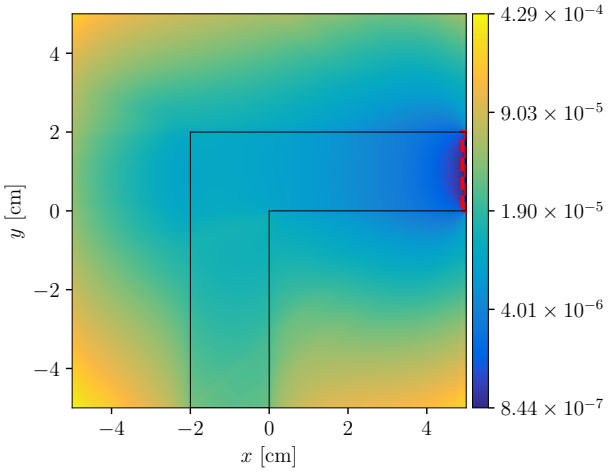

**(a)** $\widehat{\Psi}_1$



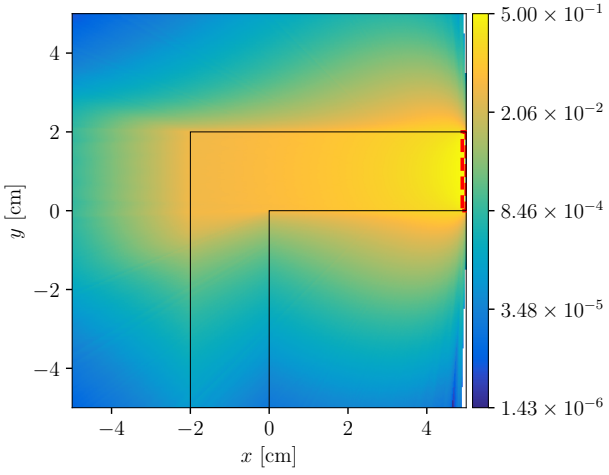

**(b)** $\widehat{\Psi}_2$



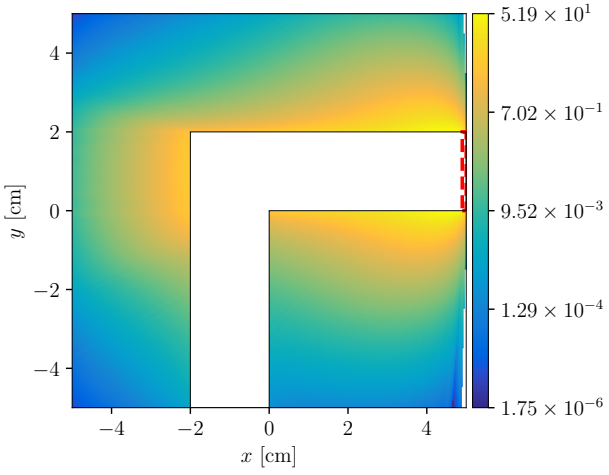

**(c)** $\widehat{\Upsilon}$




**(d)** $\widehat{\$}$

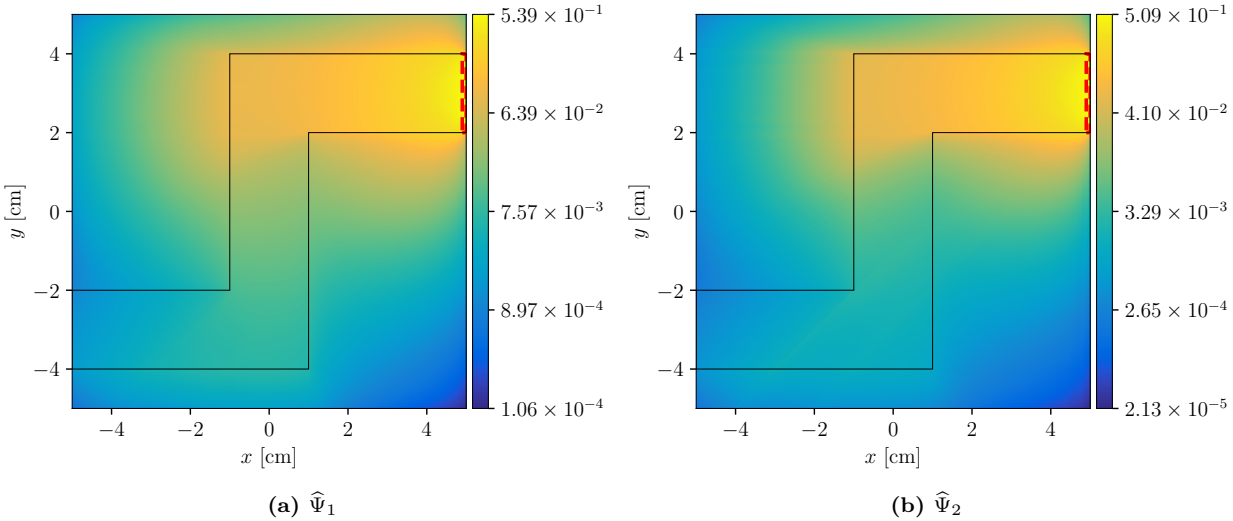



**(e)** $\widehat{p}_{\text{ff}}$




**(f)** $\widehat{Q}_2$

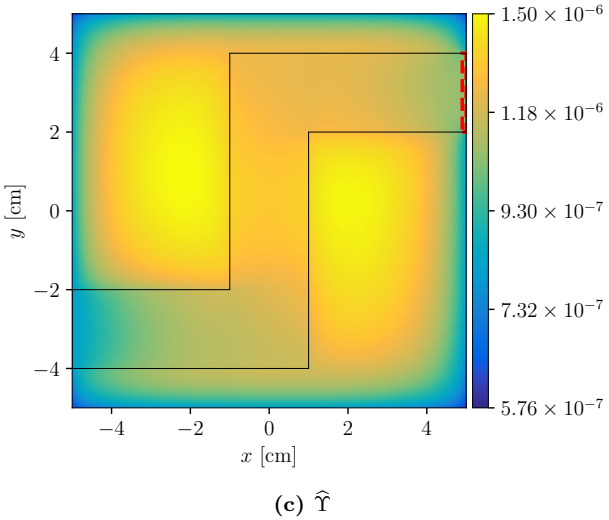
**Figure 6.31:** Test Case 2-4 Results



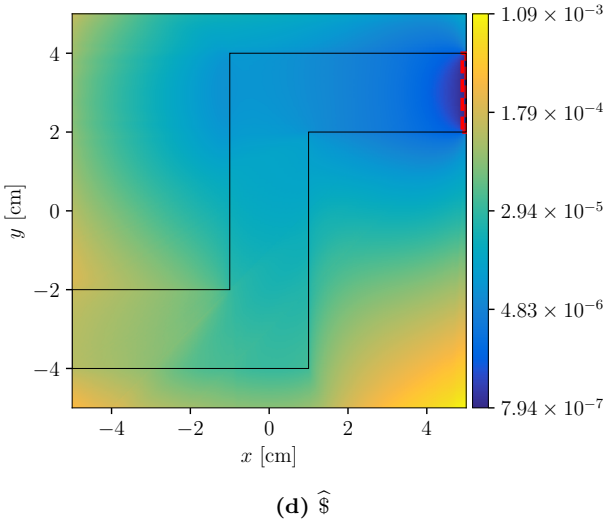

**(a)** $\widehat{\Psi}_1$



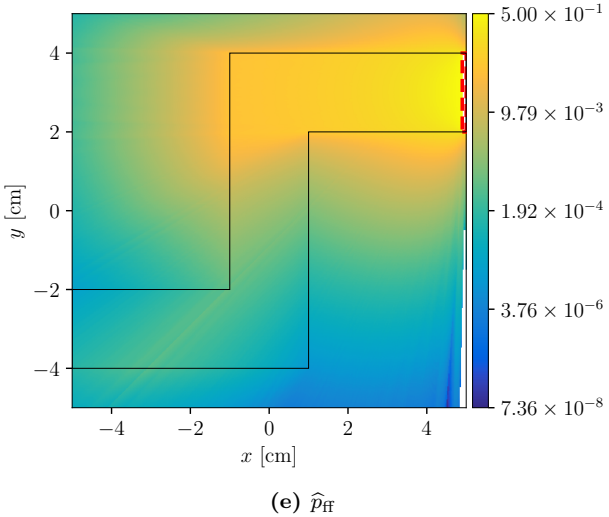

**(b)** $\widehat{\Psi}_2$



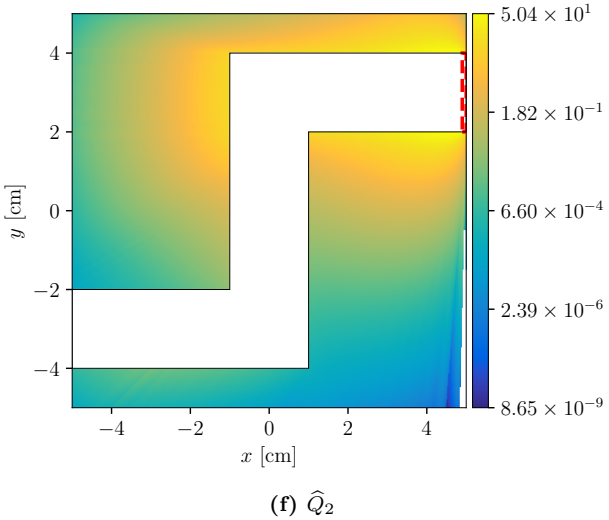

**(c)** $\widehat{\Upsilon}$




**(d)** $\widehat{\$}$

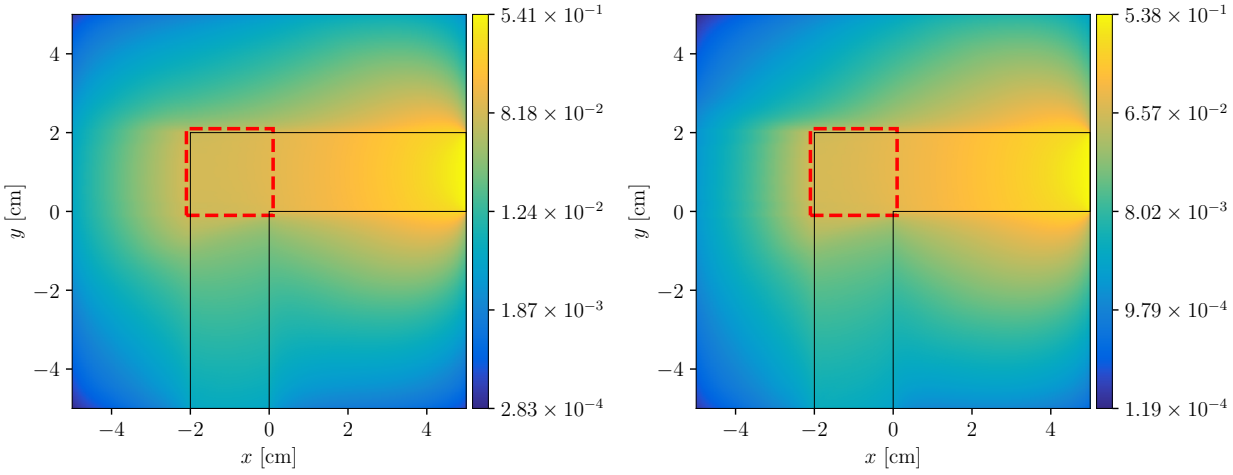



**(e)** $\widehat{p}_{\text{ff}}$




**(f)** $\widehat{Q}_2$

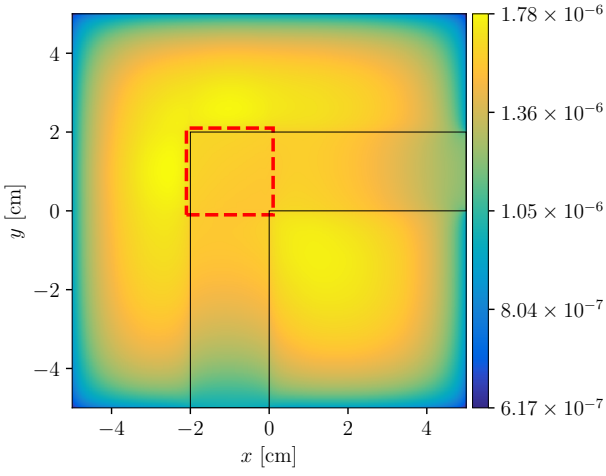
**Figure 6.32:** Test Case 2-5 Results



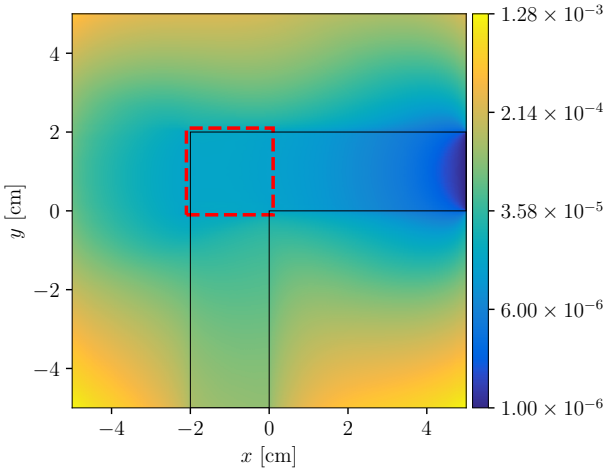

**(a)** $\widehat{\Psi}_1$



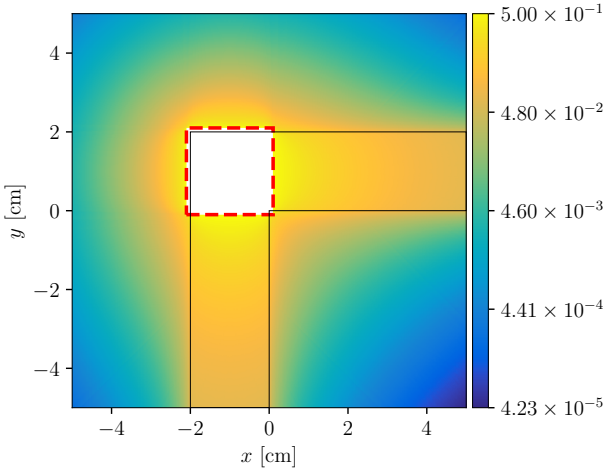

**(b)** $\widehat{\Psi}_2$



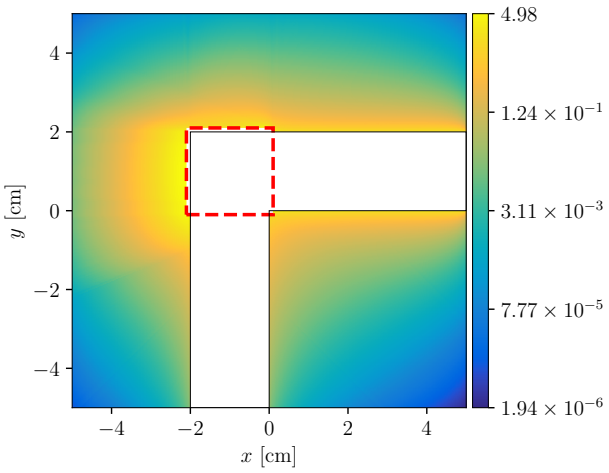

**(c)** $\widehat{\Upsilon}$




**(d)** $\widehat{\$}$

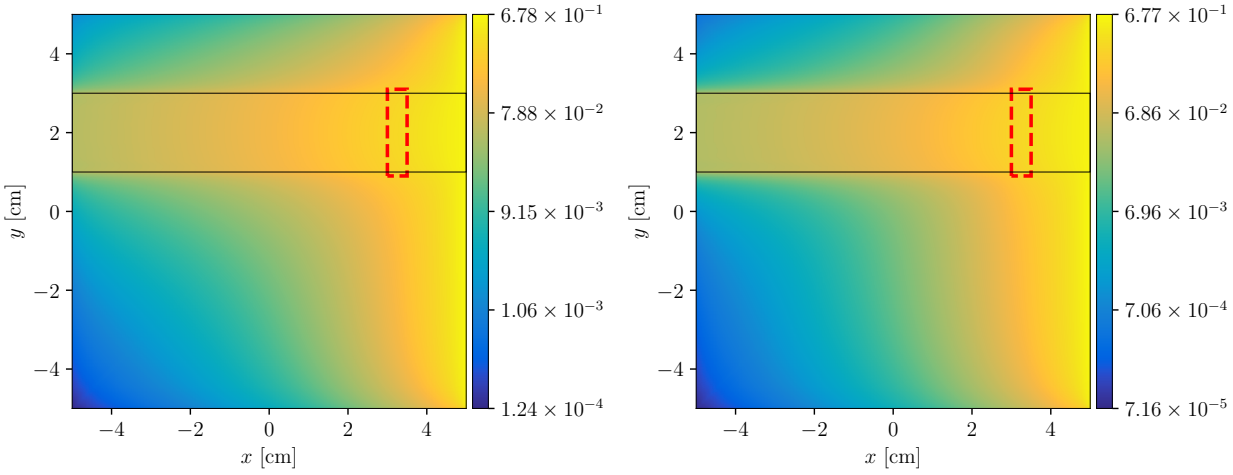



**(e)** $\widehat{p}_{\text{ff}}$




**(f)** $\widehat{Q}_2$

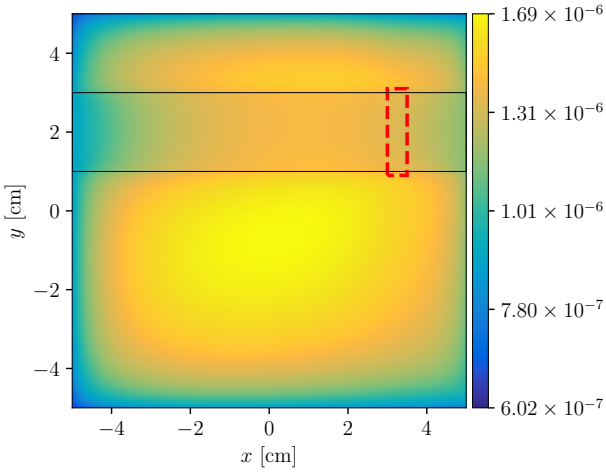
**Figure 6.33:** Test Case 2-6 Results



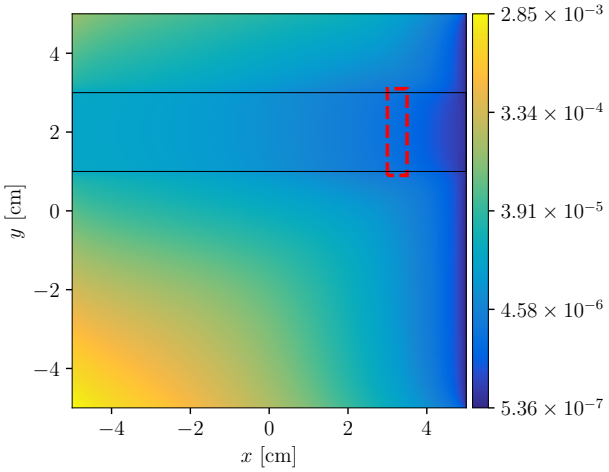

**(a)** $\widehat{\Psi}_1$



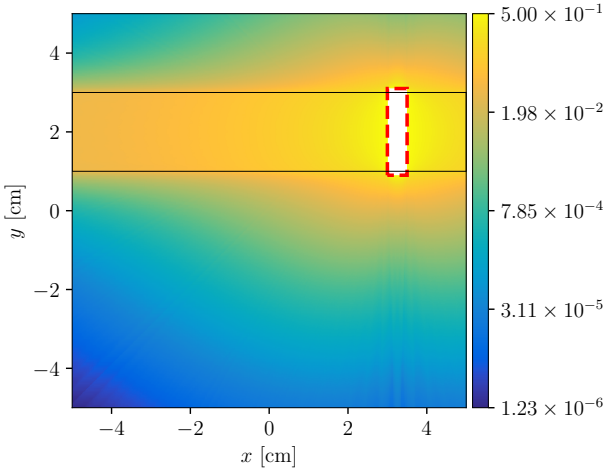

**(b)** $\widehat{\Psi}_2$



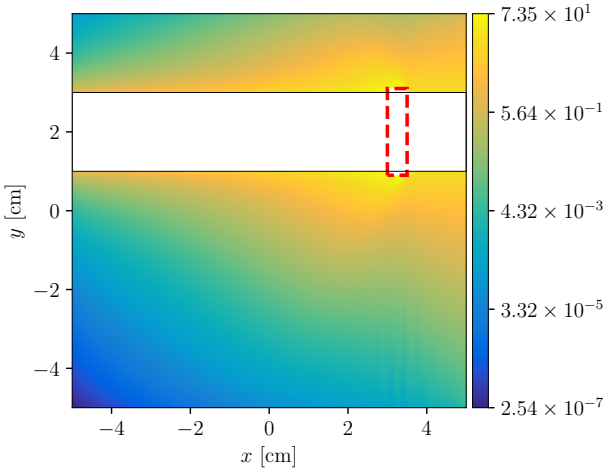

**(c)** $\widehat{\Upsilon}$




**(d)** $\widehat{\$}$




**(e)** $\widehat{p}_{\text{ff}}$




**(f)** $\widehat{Q}_2$

**Figure 6.34:** Test Case 2-7 Results

### 6.6 Summary

For the 14 1-D test cases examined herein the MCNP and COVRT mean and variance values agree within 1.4% if DXTRAN is used alone and within 3.4% if DXTRAN is used in concert with importance splitting/rouletting. For the seven 2-D test cases, the level of agreement is worse (generally within 10% but as much as 13%) but is comparable to prior work. Because of the good level of agreement in 1-D and improving degree of agreement in 2-D as the angular quadrature is refined, it is concluded that the deterministic solution of the HSMEs for DXTRAN is likely implemented correctly.

# Chapter 7

# Monte Carlo Figure of Merit Optimization

In Chapter 6, the HSMEs derived in Chapter 3 and implemented in Chapter 5 are verified. The FTE and associated computational time kernels are derived and tested in Chapter 4. With the solutions to the HSMEs and FTE, the computational cost of a Monte Carlo can be predicted [Eq. (4.32)]. The computational cost varies as a function of the variance-reduction parameters used in the calculation. As such, the variance-reduction parameters can be optimized to produce the minimum computational cost as described in Chapter 4.

A subset of the 1-D and 2-D test cases described in Chapter 6 are used to perform optimization calculations to identify DXTRAN parameters that are expected to lead to improved (larger) Monte Carlo FOM. The two sets of DXTRAN parameters optimized are (1) the Monte Carlo cell-wise rouletting parameters, $\beta_c$, and (2) the DXTRAN region size and position defined by its spatial extents.

This chapter begins with a description of the DXTRAN parameter optimization process. The chapter then continues by describing the behavior of six 1-D test cases with optimization applied. Next, the optimization behavior of six 2-D test cases is described. The chapter concludes with a summary of overall optimization results from the calculations and recommendations for how DXTRAN variance reduction parameters should be specified based on the optimization results.

## 7.1 DXTRAN Optimization Overview

For the 1-D calculations, the DXTRAN left-hand boundary is denoted as $x_{\min}$ and the right-hand boundary is $x_{\max}$. In 2-D calculations, the left- and right-hand boundaries are denoted the same and the lower and upper boundaries are $y_{\min}$ and $y_{\max}$, respectively.

The optimization calculation workflow is:

1. Create an MCNP input file and corresponding COVRT input file (no different than in Chapter 6),
2. Execute the MCNP calculations to establish a FOM corresponding to the initially guessed, non-optimized, DXTRAN parameters.
3. Execute PyCOVRT to use the COVRT input file with the Python implementation of NOMAD to predict DXTRAN parameters leading to a minimum computational cost (maximum Monte Carlo FOM),

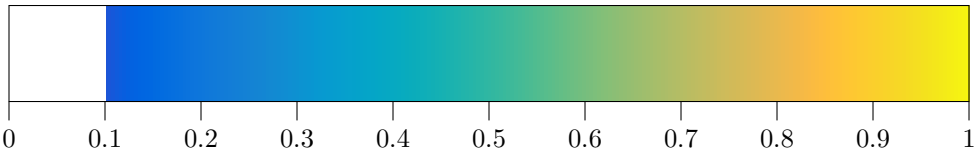



**Figure 7.1:** Color Bar for Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$ That Gives the Lowest Computational Cost

4. Incorporate the PyCOVRT-determined DXTRAN parameters into the MCNP input files and re-run to determine the change in FOM.

To attempt to make a fair comparison between pre- and post-optimization MCNP FOMs, 25 identical MCNP calculations are made for both cases. The resulting FOMs are then averaged. This attempts to mitigate the effect of the variation in MCNP behavior observed in Chapter 4. Moreover, all MCNP calculations are made on a Los Alamos National Laboratory dedicated-use computing node through an allocation on a supercomputer. Each node has two Intel Xeon CPUs (model E5-2695 v4) operating at 2.10 GHz providing 36 processing cores and 128 GB of RAM. All COVRT and MCNP calculations used 36 OpenMP threads. Only one calculation was run at a time to avoid inducing timing fluctuations like those described in Chapter 4.

For the 1-D calculations, an $S_{32}$ quadrature set is used because the calculations are quick running, so running many optimization iterations does not take unduly long. However, for the 2-D calculations an $S_8$ quadrature set is used. This level of angular refinement is chosen by balancing PyCOVRT calculation time and detail in the result. Using a coarse quadrature will likely not result in mean, variance, and future time values that match the MCNP calculations. However, the relative change in computational cost behavior arising from varying DXTRAN parameters should adequately represent the relative change in MCNP calculation behavior.

When optimization is performed, an initial guess is provided for the optimization parameters. In all cases, the initial guess for all $\beta_c$ values is one. The DXTRAN initial size and position is described with the results for each case. In these cases, some "good" guesses are made as well as "bad" guesses. Good guesses are those where the DXTRAN region closely encompasses the tally region and bad guesses are those where it does not. Because of the behavior of the optimization algorithms observed in Chapter 4, varying the "quality" of the initial guess is important to assessing the overall process.

For each of the optimization calculations for each test case, three subfigures are shown that give the optimizer iteration-by-iteration value:

1. Monte Carlo cell-wise DXTRAN rouletting parameter $\beta_c$. To display the evolution of the $\beta_c$ values, the technique of small multiples described by Tufte [124] is used where each Monte Carlo spatial cell is represented with the fluctuating $\beta_c$ value shown within as a line plot. The spatial cell is colored according to the $\beta_c$ value corresponding to the optimization iteration that produced the minimum computational cost. In all cases, the values are shaded according to the color bar shown in Fig. 7.1. The $\beta_c$ values are allowed to vary between 0.1 and 1.0, which is why the color bar is truncated below 0.1. This lower bound is used because of the large variance introduced by extremely low rouletting

parameters as demonstrated in Chapter 4. The optimal $\beta_c$ values are not tabulated but any trends observed are described along with the results of each test case.

2. DXTRAN boundary values. The DXTRAN boundary values for each iteration are given as a line plot. A dotted line is also provided that indicates the iteration that corresponds to the lowest computational cost. The values corresponding to the optimization iteration that produced the minimum computational cost are not shown in the figures but are instead given with the results of each test case.

3. Computational cost function. The computational cost function by optimization iteration is given as a line plot. A dotted line is also provided that indicates the iteration that corresponds to the lowest computational cost. Finally, an inset axis shows the iteration corresponding to the lowest computational cost, the function's value, and neighboring iteration points.

Because of the number of figures for each case, all figures are collected following the last 1-D and 2-D test case discussions. Note that all computational cost values are expressed in minutes consistent with MCNP output [1, page 2-117] and Table 4.1.

### 7.1.1 Optimization of Test Case 1-2

For this case 2990 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (7.99, 10.00)$, the computational cost is calculated as $1.29 \times 10^{-5}$. The minimum computational cost was found at iteration 1305 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (8.94, 9.98)$. The optimized computational cost is calculated as $1.24 \times 10^{-5}$. This change is approximately 4% better than in the unoptimized case. Because of the amont of noise in the computational cost observed while profiling the MCNP code, this small of a change is likely to be insignificant in the eventual MCNP calculations.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $2.54 \times 10^5 \pm 3.48 \times 10^3$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $2.50 \times 10^5 \pm 3.11 \times 10^3$. As such, optimization provided a 2% decrease in FOM.

As expected, the 4% improvement predicted in the COVRT cases was not realized in the MCNP calculations (exhibiting a 2% decrease on average). However, several of the optimization behaviors are noteworthy. First, the left-most spatial cell (encompassing the forward source) has a reduced $\beta_c$. This appears to be an attempt to reduce the time contribution to the cost from this cell (because this cell contributes additional time relative to others because of the time taken to generate source particles). Presumably this effect is exacerbated if DXTRAN is also played during source emission. Also, the DXTRAN region encompasses the right-most cell. Making $x_{\min} \approx 9$ reduces the non-DXTRAN region and accordingly the amount of time taken to process DXTRAN. For this $\beta_c$ is not one within the DXTRAN region, but this reduction is meaningless because DXTRAN is not played here. Finally, one should recognize that (a) the initial guess was quite close to the optimal computational cost and (b) many iterations were spent by the optimizer casting about but not finding an improved computational cost. About half the iterations are spent after the minimum is found with only minor changes to the DXTRAN parameters. Modifying the coarseness of permissible search values in future work is expected to reduce this behavior.

### 7.1.2 Optimization of Test Case 1-4

For this case 1857 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (7.99, 10.00)$, the computational cost is calculated as $1.80 \times 10^{-5}$. The minimum computational cost was found at iteration 863 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (9.99, 10.00)$. The optimized computational cost is calculated as $1.60 \times 10^{-5}$. This change is approximately 11% better than in the unoptimized case.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.45 \times 10^5 \pm 1.07 \times 10^3$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.58 \times 10^5 \pm 1.01 \times 10^3$. As such, optimization provided a 9% increase in FOM.

The predicted 11% improvement in computational cost is realized with a 9% improvement in FOM. Unlike in the previous case the optimizer moved the DXTRAN region to closely encompass the tally region. For this calculation, $\beta_4$, $\beta_6$, and $\beta_7$ have optimizer assigned values that are not one. Because these cells are purely absorbing they have no ability to initiate DXTRAN. As such, logic should be introduced to disallow the optimizer from modifying optimization parameters in cells that are purely absorbing or within the DXTRAN region as described previously in Section 7.1.1. Like the previous case, about half the optimization iterations are spent after the minimum is found with only very minor changes to the DXTRAN parameters.

### 7.1.3 Optimization of Test Case 1-5

For this case 913 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (7.99, 10.00)$, the computational cost is calculated as $2.24 \times 10^{-5}$. The minimum computational cost was found at iteration 302 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (8.02, 8.88)$. The optimized computational cost is calculated as $2.21 \times 10^{-5}$. This change is approximately 1% better than in the unoptimized case. Because of the amont of noise in the computational cost observed while profiling the MCNP code, this small of a change is likely to be insignificant in the eventual MCNP calculations.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.23 \times 10^5 \pm 1.29 \times 10^3$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.42 \times 10^5 \pm 1.19 \times 10^3$. As such, optimization provided a 15% increase in FOM.

In this case COVRT predicted a 1% improvement in computational cost but a 15% improvement in FOM is realized. This is unexpected. Moreover, the optimizer found that the optimal DXTRAN position is only over the left-half of the track-length tally region. Historically, common guidance has been to avoid having a single tally exist both inside and outside a DXTRAN region. The logic underpinning this argument is that there can be large weight fluctuation between those tallies immediately outside the DXTRAN region versus those inside. In this case this does not seem to be a detriment because the material is relatively optically thin so a particle is unlikely to scatter in the region not covered by the DXTRAN region (to contribute additional track length) before leaking out of the problem. This behavior is not an abnormality; it is also observed in Section 7.1.8 for Test Case 2-3, which also incorporates a track-length tally.

### 7.1.4 Optimization of Test Case 1-6

For this case 1385 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (7.99, 10.00)$, the computational cost is calculated as $1.86 \times 10^{-5}$. The minimum computational cost was found at iteration 834 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (8.00, 8.87)$. The optimized computational cost is calculated as $1.84 \times 10^{-5}$. This change is approximately 1% better than in the unoptimized case. Because of the amont of noise in the computational cost observed while profiling the MCNP code, this small of a change is likely to be insignificant in the eventual MCNP calculations.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.63 \times 10^5 \pm 9.54 \times 10^2$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.91 \times 10^5 \pm 1.85 \times 10^3$. As such, optimization provided a 17% increase in FOM.

In this case COVRT predicted a 1% improvement in computational cost but a 17% improvement in FOM is realized. The behavior of the optimizer and the subsequent performance is similar to Test Case 1-5 as described previously. This makes sense because only the forward source changed between the two cases. This suggests that DXTRAN specification is more sensitive to the tally conditions than the source conditions. Further, to the left of the DXTRAN region the $\beta_c$ values are depressed to reduce DXTRAN contributions. In this case, the reduction in time satisfactorily offsets the associated increase in variance.

### 7.1.5 Optimization of Test Case 1-13

For this case 795 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (4.99, 7.01)$, the computational cost is calculated as $1.04 \times 10^{-5}$. The minimum computational cost was found at iteration 226 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (5.94, 5.96)$. The optimized computational cost is calculated as $9.29 \times 10^{-6}$. This change is approximately 11% better than in the unoptimized case.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $3.25 \times 10^5 \pm 4.33 \times 10^3$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $2.40 \times 10^5 \pm 1.31 \times 10^4$. As such, optimization provided a 26% decrease in FOM.

In this case COVRT predicted a 11% improvement in computational cost but a 26% reduction in FOM is realized. This case represents the only 1-D calculation where the optimizer-generated DXTRAN parameters caused a reduction in FOM. Initially, the DXTRAN fully encompassed the internal current tally but the optimization dictated that the DXTRAN be made small and slightly to the left of the tally surface at $x = 6$. This behavior is not altogether unreasonable (it is similar to Sections 7.1.3 and 7.1.8); but this may have been a case of false optimizer convergence to a minimum value.

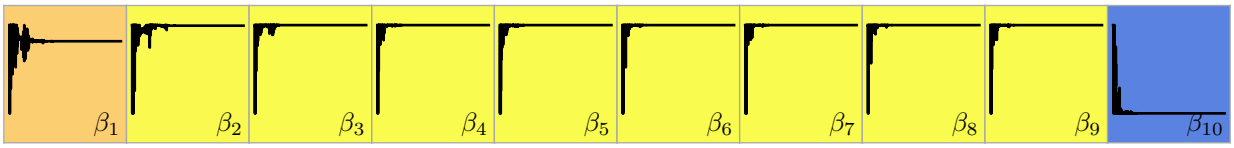
### 7.1.6 Optimization of Test Case 1-14

For this case 743 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (4.99, 9.01)$, the computational cost is calculated

as $1.60 \times 10^{-5}$. The minimum computational cost was found at iteration 183 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (6.02, 8.00)$. The optimized computational cost is calculated as $1.35 \times 10^{-5}$. This change is approximately 15% better than in the unoptimized case.

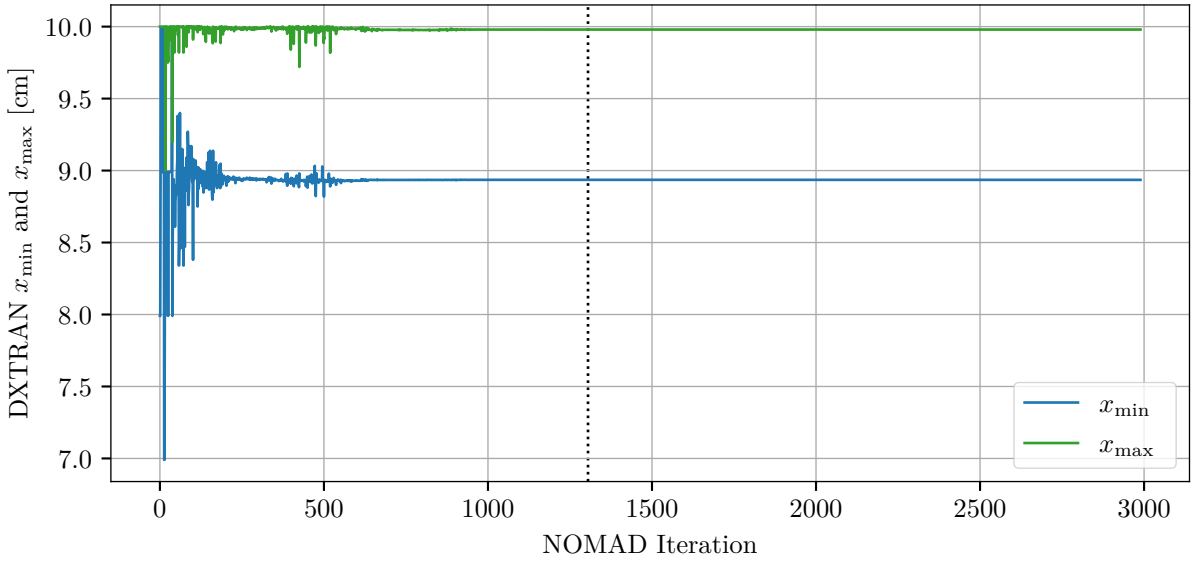
As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $2.11 \times 10^5 \pm 2.96 \times 10^3$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $2.84 \times 10^5 \pm 2.32 \times 10^3$. As such, optimization provided a 35% increase in FOM.

In this case COVRT predicted a 15% improvement in computational cost but a 35% improvement in FOM is realized. Unlike in Test Case 1-13 with the internal current tally in Section 7.1.5, this case encompassed the track-length tally with the DXTRAN region. However, similar to Test Case 1-13, the $\beta_c$ rouletting values are kept at one globally.

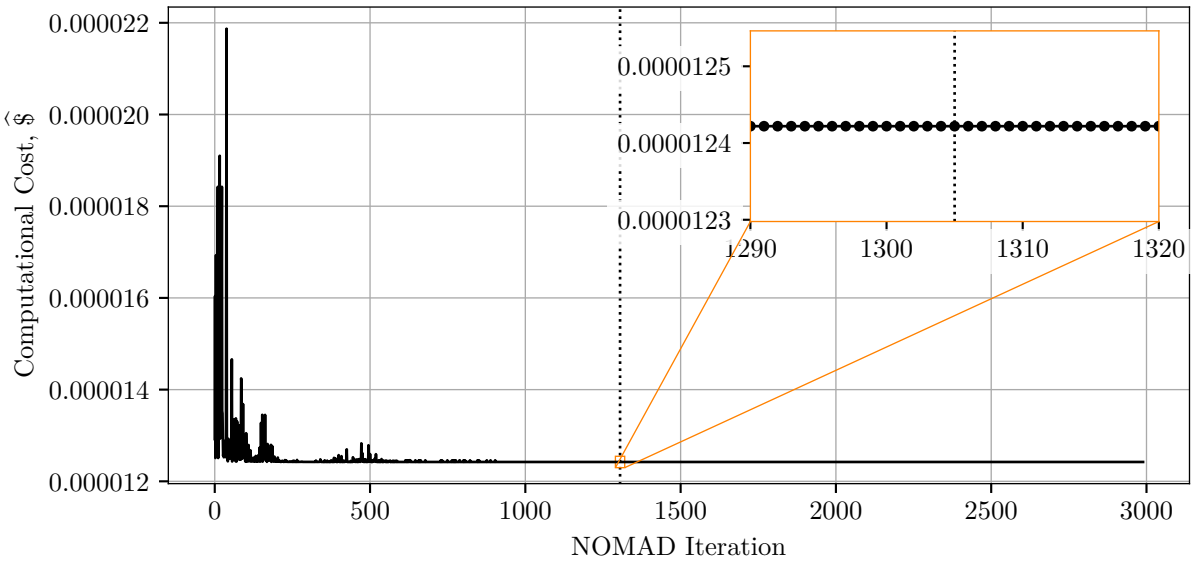



Monte Carlo Cells along $x$ Axis

**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$

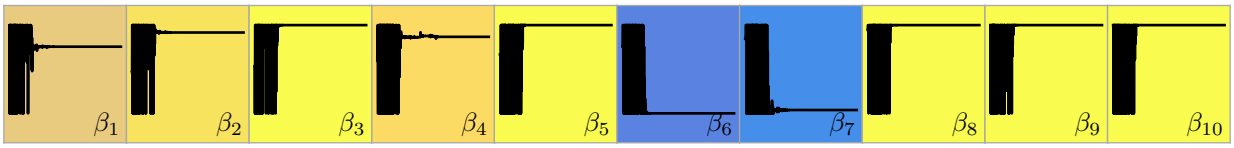



**(b)** DXTRAN Boundary

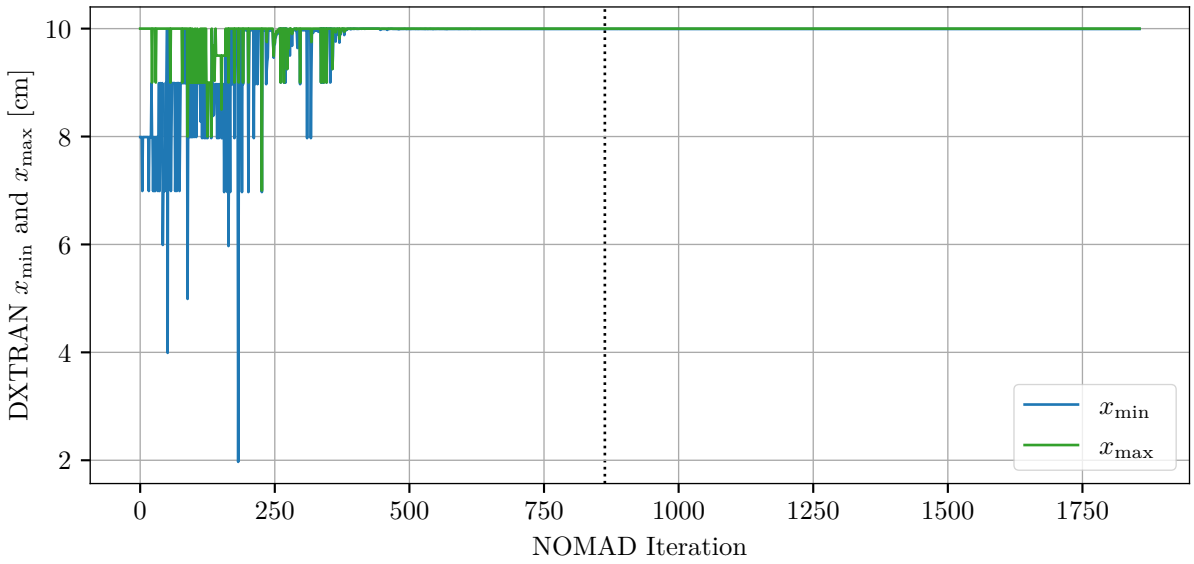



**(c)** Computational Cost Function

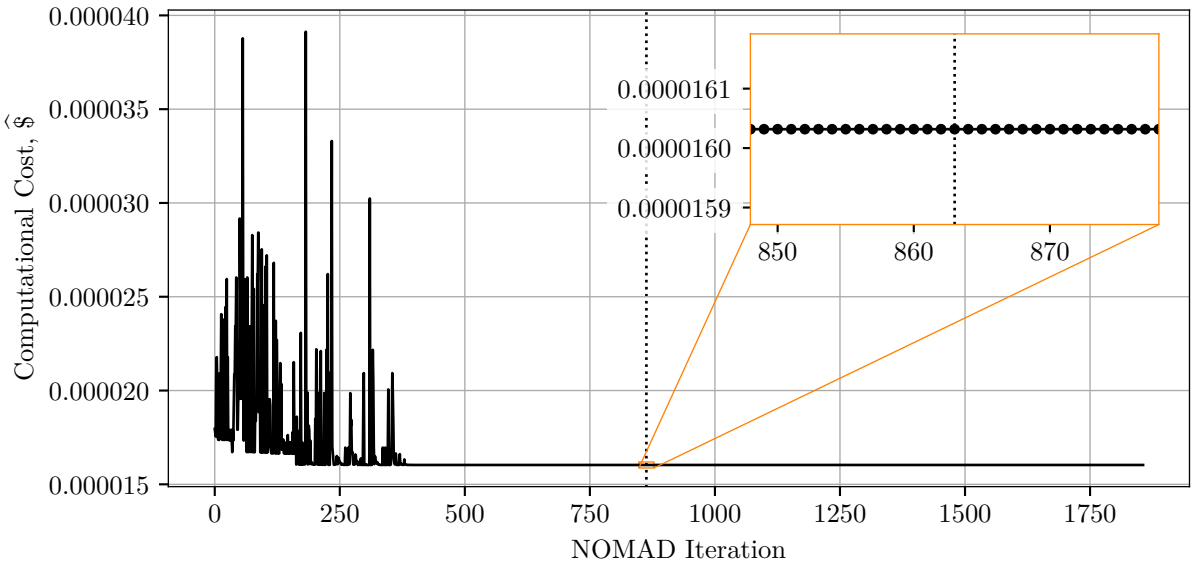
**Figure 7.2:** Test Case 1-2 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$

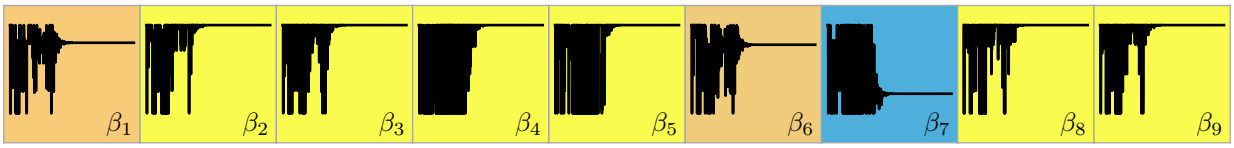



**(b)** DXTRAN Boundary

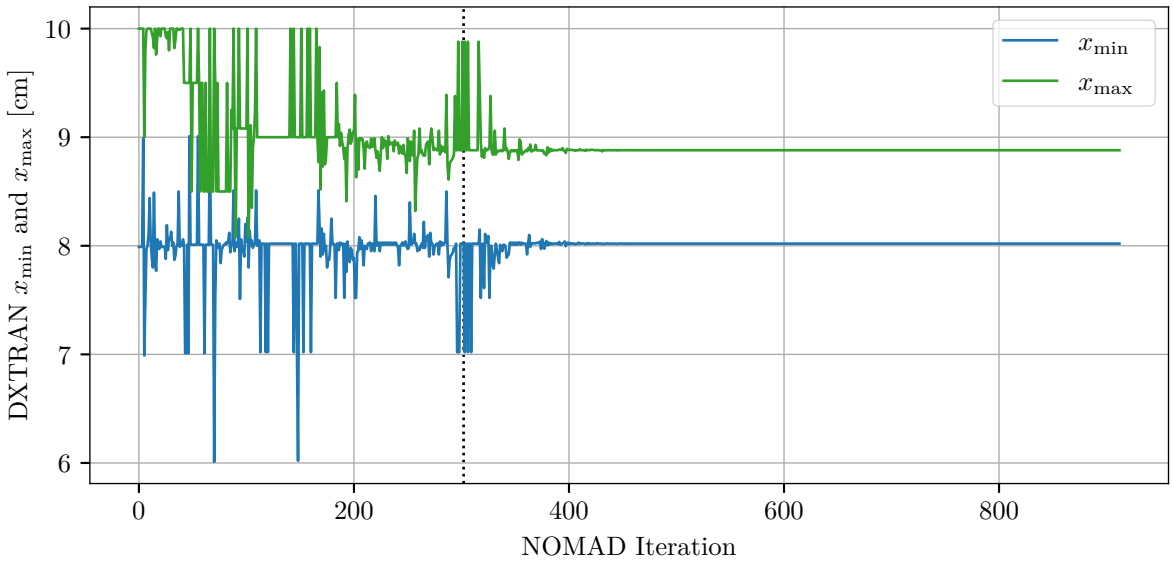



**(c)** Computational Cost Function

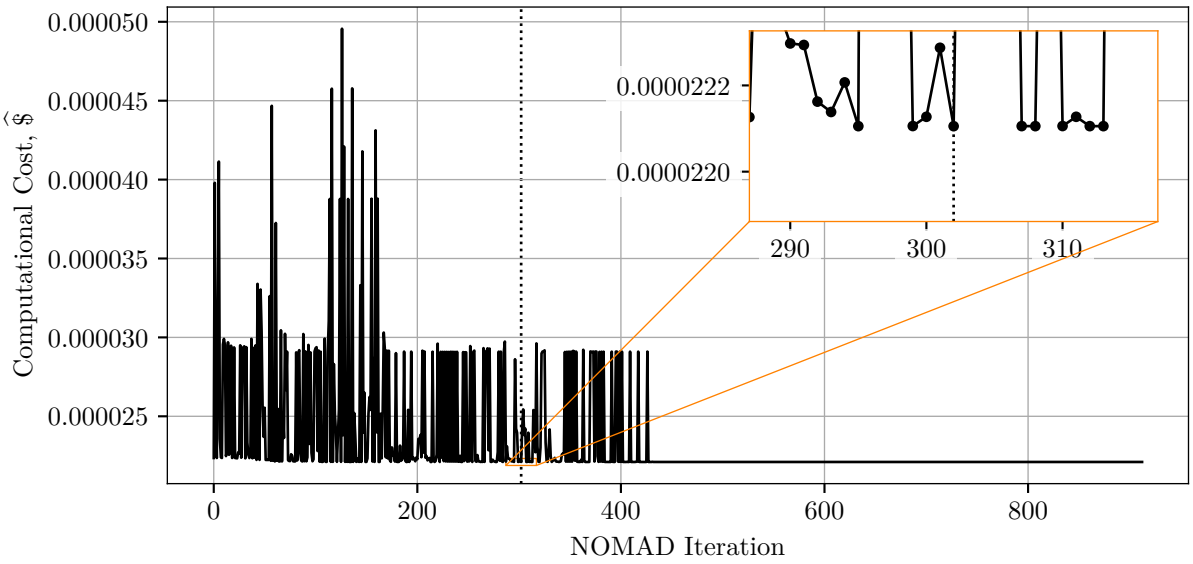
**Figure 7.3:** Test Case 1-4 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$

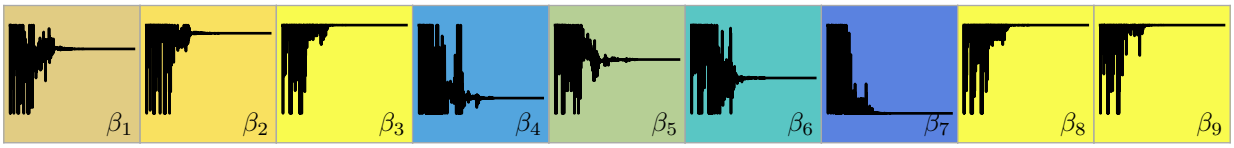



**(b)** DXTRAN Boundary

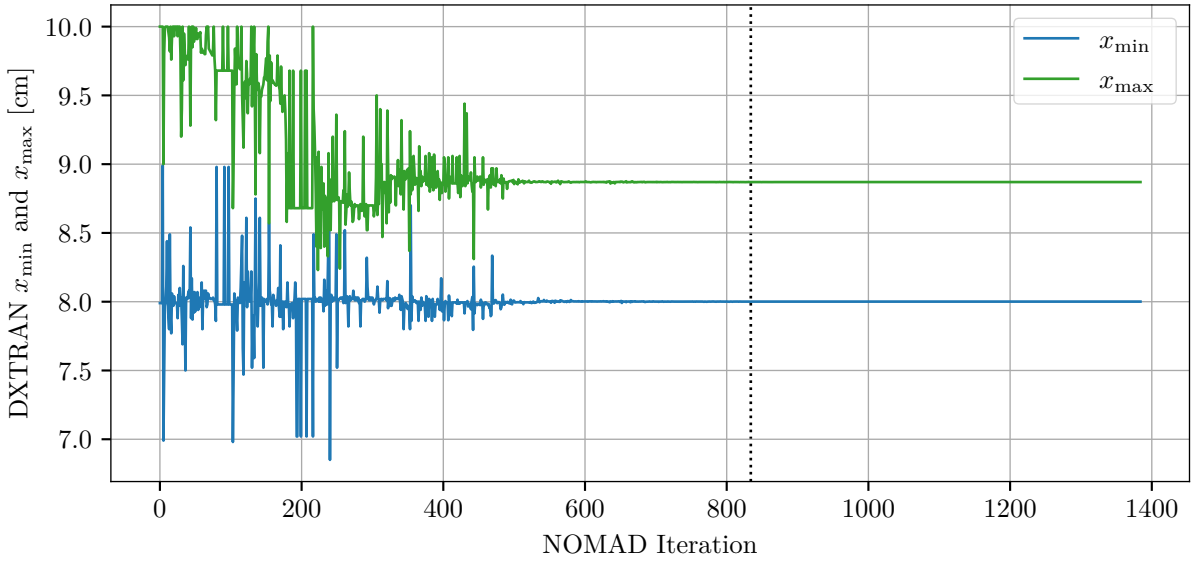



**(c)** Computational Cost Function

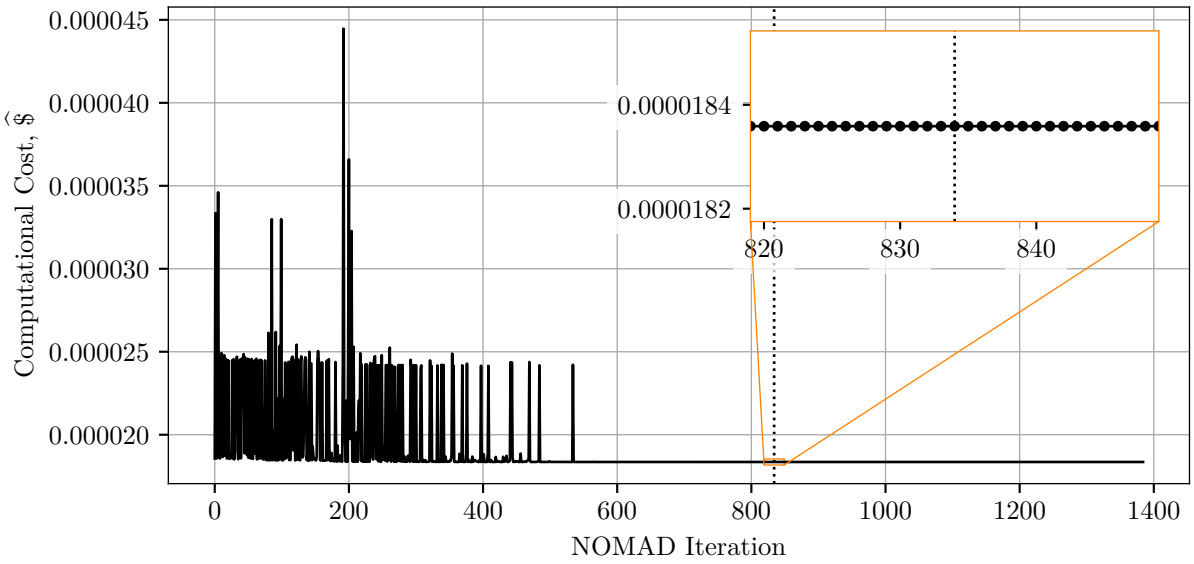
**Figure 7.4:** Test Case 1-5 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$

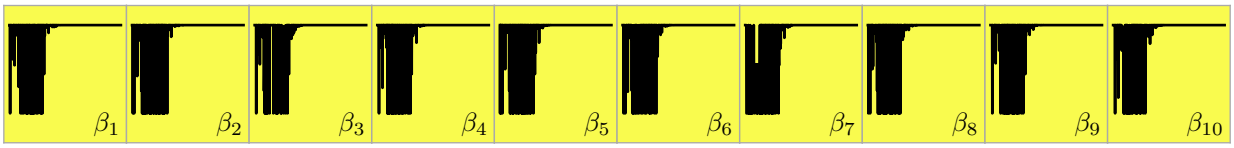



**(b)** DXTRAN Boundary

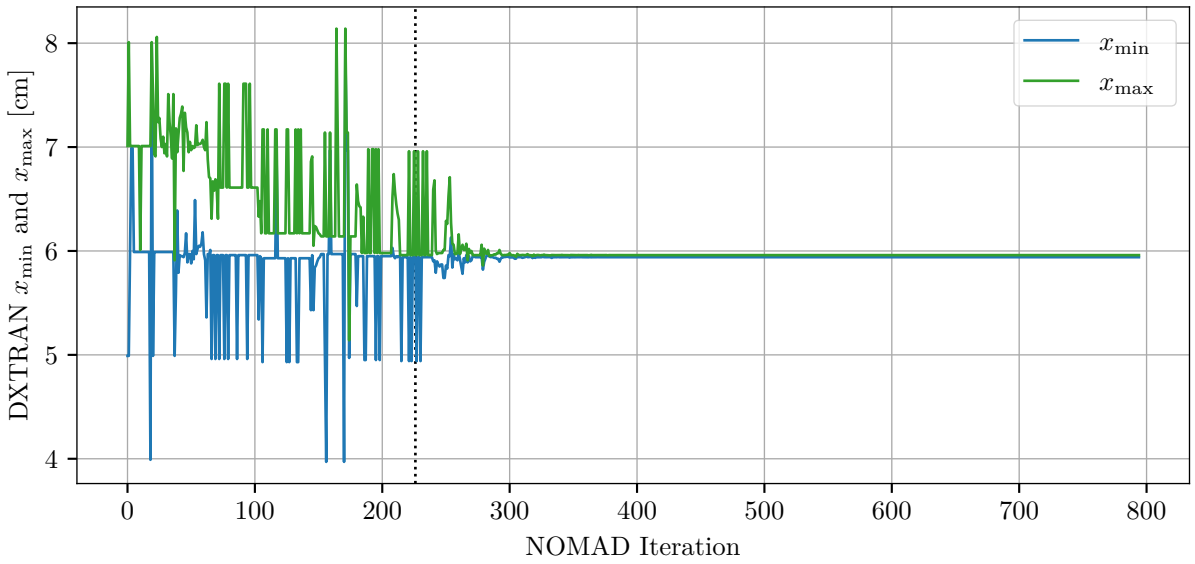



**(c)** Computational Cost Function

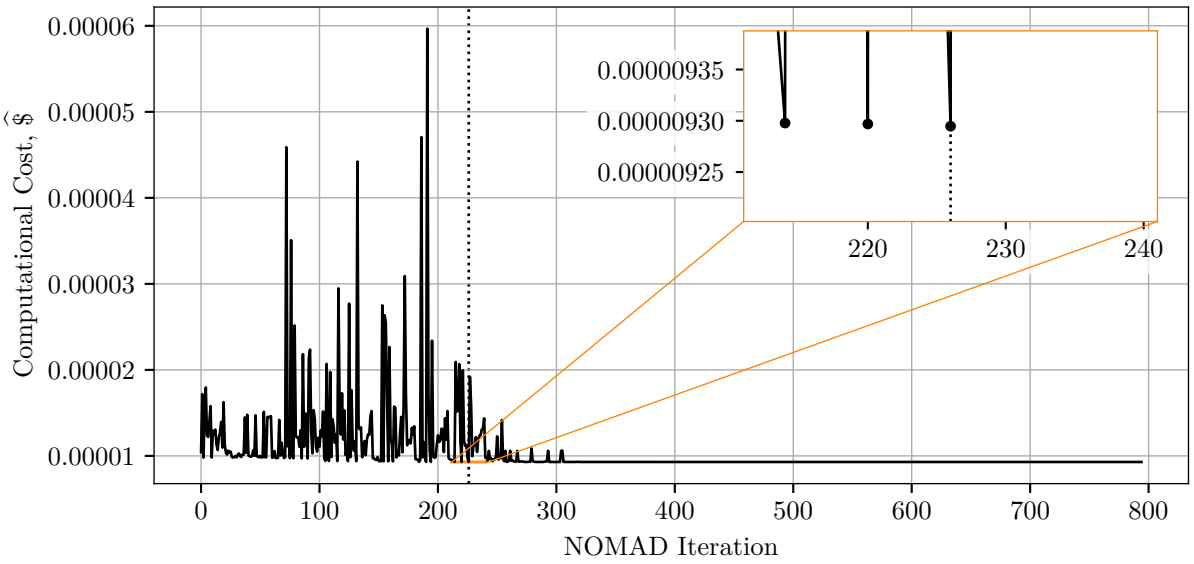
**Figure 7.5:** Test Case 1-6 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$

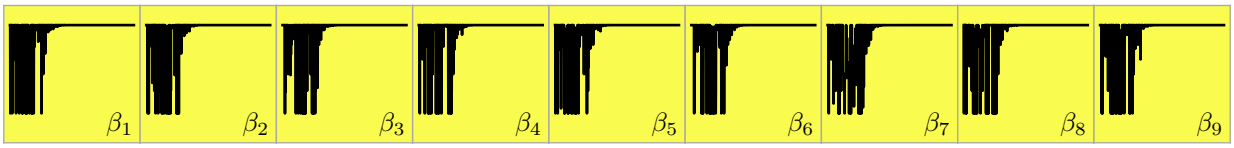



**(b)** DXTRAN Boundary

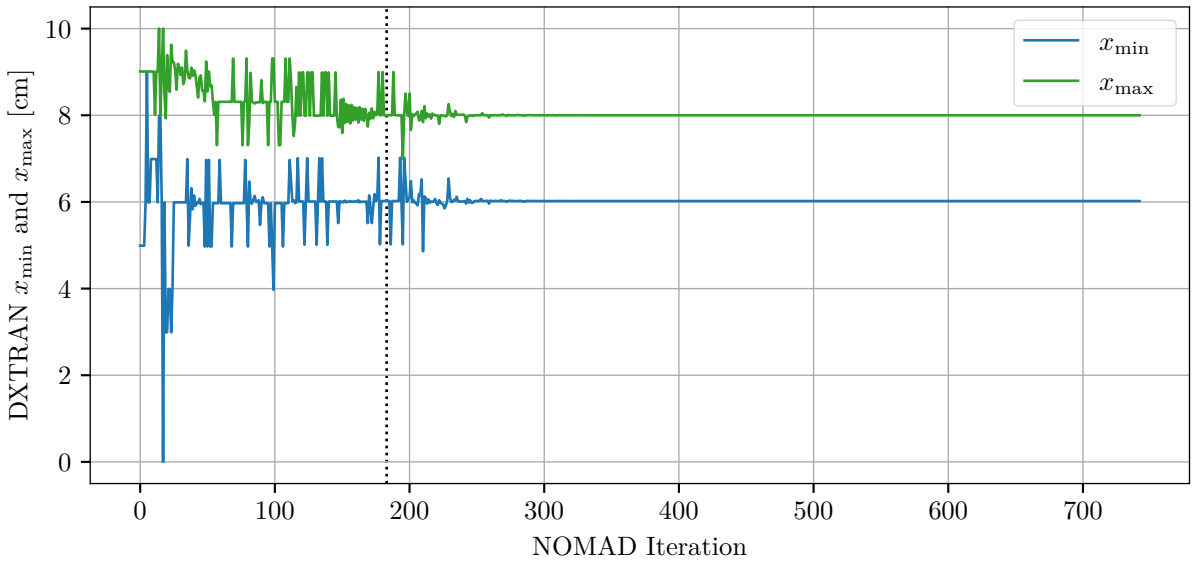



**(c)** Computational Cost Function

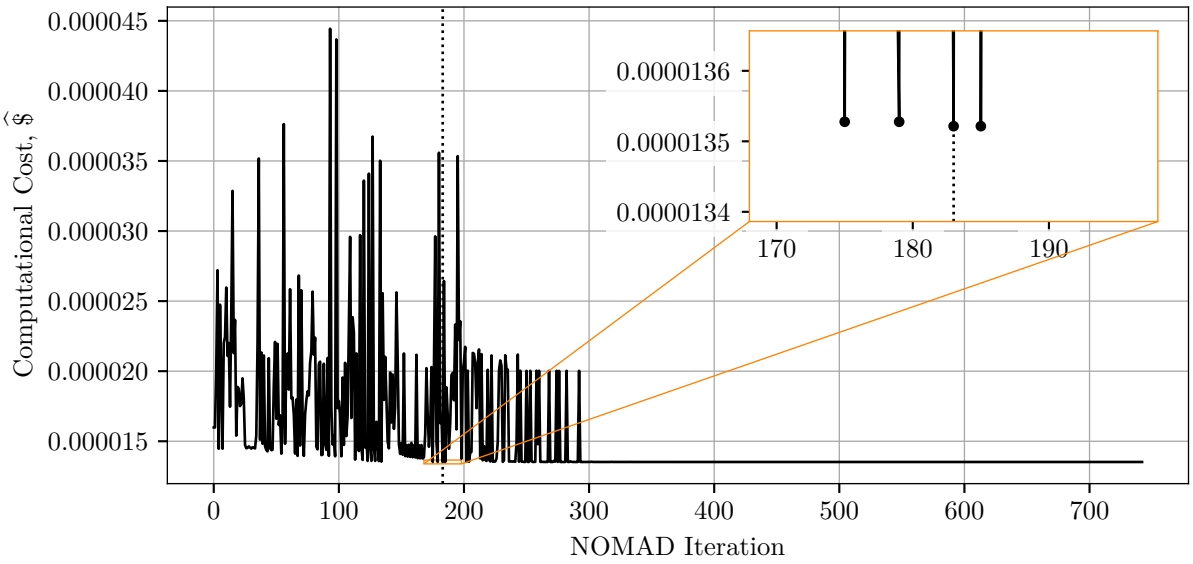
**Figure 7.6:** Test Case 1-13 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$




**(b)** DXTRAN Boundary




**(c)** Computational Cost Function

**Figure 7.7:** Test Case 1-14 Optimization Evolution

### 7.1.7 Optimization of Test Case 2-2

For this case 9321 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (0.00, 5.00)$ and $(y_{\min}, y_{\max}) = (-4.99, 5.00)$ the computational cost is calculated as $3.29 \times 10^{-5}$. The minimum computational cost was found at iteration 9235 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (4.88, 5.00)$ and $(y_{\min}, y_{\max}) = (-4.64, 5.00)$. The optimized computational cost is calculated as $2.67 \times 10^{-5}$. This change is approximately 19% better than in the unoptimized case.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $7.33 \times 10^4 \pm 7.38 \times 10^2$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $8.48 \times 10^4 \pm 6.49 \times 10^2$. As such, optimization provided a 16% increase in FOM.

In this case COVRT predicted a 19% improvement in computational cost but a 16% improvement in FOM is realized. Figure 7.8a demonstrates a reasonable change in $\beta_c$ values where those near the forward source and far from the tally are reduced. The reduction follows a general pattern where cells toward the center of the problem are not reduced as much as those near the periphery though it is not perfectly symmetric. This indicates that those particles initiating DXTRAN are more important and likely to contribute to the tally than those near the periphery. Furthermore, the DXTRAN region was moved by the optimizer from covering the right-half of the problem to being essentially just encompassing the tally region. The final position is not perfectly symmetric or encompassing of the tally region. However, it is close enough to be logical based on the 1-D results and indicates that additional optimizer controls may be needed to ensure symmetric behavior for an otherwise symmetric problem.

### 7.1.8 Optimization of Test Case 2-3

For this case 19173 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (3.00, 5.00)$ and $(y_{\min}, y_{\max}) = (3.00, 5.00)$ the computational cost is calculated as $5.55 \times 10^{-4}$. The minimum computational cost was found at iteration 15683 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (2.13, 4.67)$ and $(y_{\min}, y_{\max}) = (2.35, 4.60)$. The optimized computational cost is calculated as $4.78 \times 10^{-4}$. This change is approximately 14% better than in the unoptimized case.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.50 \times 10^3 \pm 8.52 \times 10^1$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $4.86 \times 10^3 \pm 1.36 \times 10^2$. As such, optimization provided a 225% increase in FOM.

In this case COVRT predicted a 14% improvement in computational cost but a 225% improvement in FOM is realized. In this case, the FOM improvement is far greater than the predicted change in computational cost. This is consistent with Test Case 1-5 in Section 7.1.3. Moreover, the final DXTRAN position in this test case is roughly 2×2 cm; however, it is offset down and to the left of the $2 \times 2$-cm track-length tally region. This type of behavior is also consistent with Test Case 1-5. Because of this consistent behavior and the behavior of Test Case 1-14 where an internal track-length tally was closely encompassed by the DXTRAN region, an

additional 2-D test case that would be good to examine in the future might feature an internal track-length tally. That said, because of the consistent under-prediction of improvement when using track-length tallies, more effort to better characterize the future time contribution may be required. The shape and magnitudes of the $\beta_c$ values are also noteworthy. In particular, the $\beta_c$ values are reduced moving radially outward from the tally region in a roughly symmetric pattern. Additional study on the fall off of the optimized $\beta_c$ values versus optical distance from the tally region represents potentially insightful future work.

#### 7.1.9 Optimization of Test Case 2-4

For this case 3500 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (4.50, 5.00)$ and $(y_{\min}, y_{\max}) = (0.00, 2.00)$ the computational cost is calculated as $1.24 \times 10^{-4}$. The minimum computational cost was found at iteration 83 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (4.49, 5.00)$ and $(y_{\min}, y_{\max}) = (0.00, 2.21)$. The optimized computational cost is calculated as $1.23 \times 10^{-4}$. This change is not significant and suggest that this case does not benefit from DXTRAN. Because of the amont of noise in the computational cost observed while profiling the MCNP code, this small of a change is likely to be insignificant in the eventual MCNP calculations.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.62 \times 10^4 \pm 1.07 \times 10^3$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $1.59 \times 10^4 \pm 1.01 \times 10^3$. As such, optimization provided a 2% decrease in FOM.

In this case COVRT predicted less than a 1% improvement in computational cost but a 2% reduction in FOM is realized. For this test case with a 90° duct and for Test Case 2-5 in Section 7.1.10 with a dogleg duct, the optimizer's behavior is nearly identical. In both of these cases, the optimal $\beta_c$ values are kept at one globally and the DXTRAN region is not substantially moved from the initially guessed configuration where it closely encompasses the tally region.

#### 7.1.10 Optimization of Test Case 2-5

For this case 3433 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (4.50, 5.00)$ and $(y_{\min}, y_{\max}) = (2.00, 4.00)$ the computational cost is calculated as $4.76 \times 10^{-4}$. The minimum computational cost was found at iteration 4 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (4.49, 5.00)$ and $(y_{\min}, y_{\max}) = (2.00, 4.01)$. The optimized computational cost is calculated as $4.72 \times 10^{-4}$. This change is not significant and suggest that this case does not benefit from DXTRAN. Because of the amont of noise in the computational cost observed while profiling the MCNP code, this small of a change is likely to be insignificant in the eventual MCNP calculations.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $4.17 \times 10^3 \pm 2.68 \times 10^2$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $4.11 \times 10^3 \pm 2.82 \times 10^2$. As such, optimization provided a 2% decrease in FOM.

In this case COVRT predicted less than a 1% improvement in computational cost but a 2% reduction in

FOM is realized. This behavior is very similar to Test Case 2-4 in Section 7.1.9, so the comments made there also apply to this test case.

### 7.1.11 Optimization of Test Case 2-6

For this case 12218 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (-2.10, 0.10)$ and $(y_{\min}, y_{\max}) = (-0.10, 2.10)$ the computational cost is calculated as $2.70 \times 10^{-4}$. The minimum computational cost was found at iteration 8326 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (2.90, 4.99)$ and $(y_{\min}, y_{\max}) = (-0.05, 2.13)$. The optimized computational cost is calculated as $9.86 \times 10^{-5}$. This change is approximately 64% better than in the unoptimized case.

As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $2.32 \times 10^3 \pm 1.23 \times 10^1$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $2.45 \times 10^4 \pm 2.38 \times 10^2$. As such, optimization provided a 956% increase in FOM.

In this case COVRT predicted a 64% improvement in computational cost but a 956% improvement in FOM is realized. This case represents the largest predicted improvement as a result of optimization and also the largest discrepancy between the predicted and realized improvement as a result of optimization. This case examines the behavior of the optimizer when the initially guessed DXTRAN region size and position is used strictly as a shield for particles traversing the duct. The optimizer moves the DXTRAN region, kept at roughly $2 \times 2$ cm, to the end of the duct to both draw particles toward the tally region and to shield it. In addition, the $\beta_c$ values for this case are varied across the permissible range with cells generally optically far from the tally specified to minimize $\beta_c$.

Because this case is otherwise identical to Test Case 2-4 in Section 7.1.9, it is surprising to observe a different optimized result. This case has a higher Monte Carlo FOM than Test Case 2-4 suggesting that this DXTRAN and $\beta_c$ configuration is better in a FOM sense than closely encompassing the tally region. As such, this case demonstrates that initially suboptimal guessed parameters that are still of good quality can confound the MADS optimization algorithm. This result also indicates that additional future work to identify a systematic method to produce guessed optimization parameters (or a more robust optimization method) would be valuable.

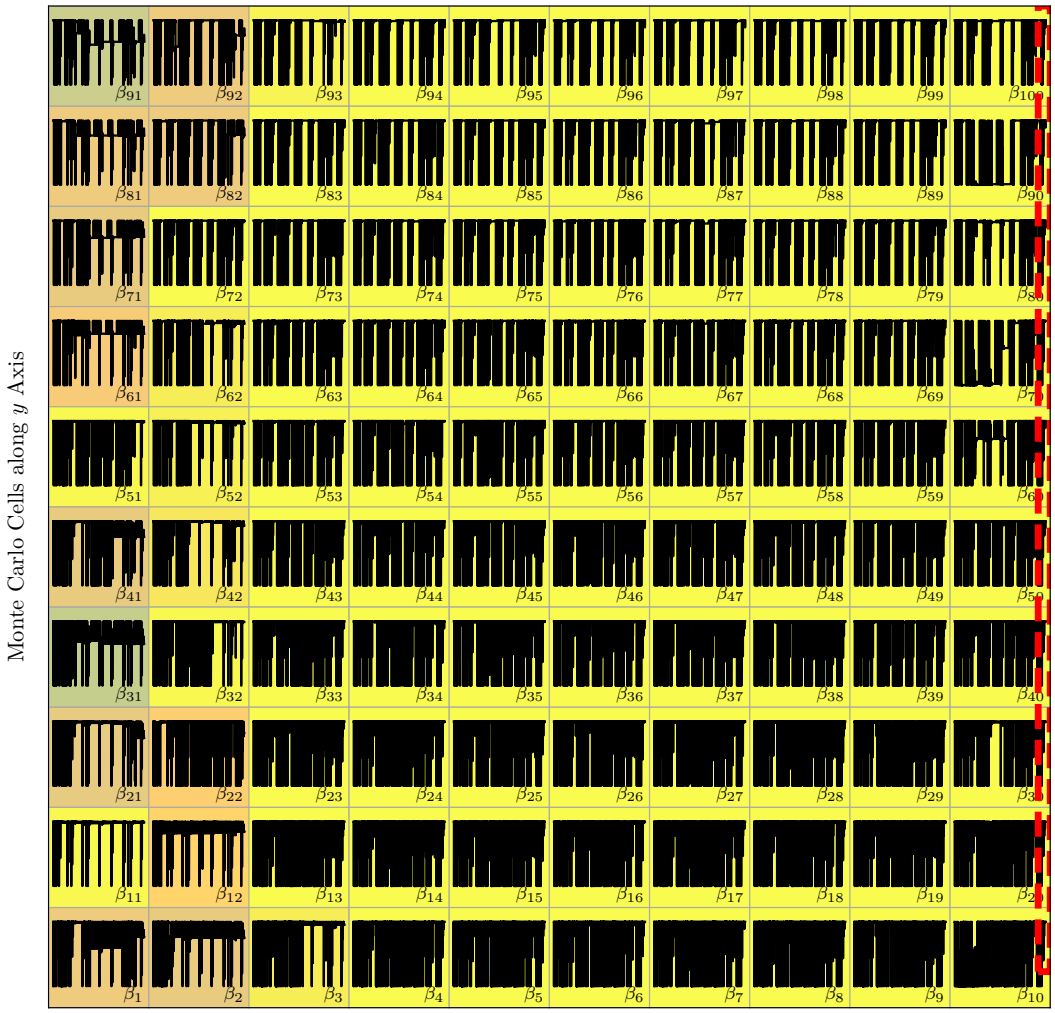
### 7.1.12 Optimization of Test Case 2-7

For this case 23005 optimization iterations were performed. For the initial guess with $\beta_c$ set to unity globally and with the DXTRAN region defined with $(x_{\min}, x_{\max}) = (3.00, 3.50)$ and $(y_{\min}, y_{\max}) = (0.90, 3.10)$ the computational cost is calculated as $3.02 \times 10^{-4}$. The minimum computational cost was found at iteration 22953 (neglecting failed attempts). For this iteration, the DXTRAN region is defined with $(x_{\min}, x_{\max}) = (-0.41, 4.96)$ and $(y_{\min}, y_{\max}) = (0.13, 3.04)$. The optimized computational cost is calculated as $2.14 \times 10^{-4}$. This change is approximately 29% better than in the unoptimized case.

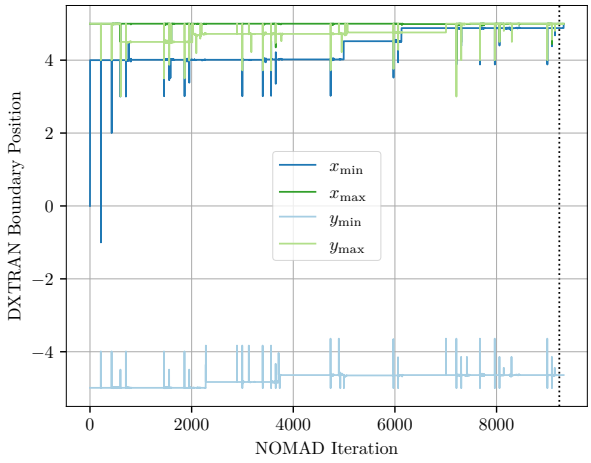
As a result of running 25 MCNP calculations with the initial DXTRAN parameters, the FOM obtained is $4.56 \times 10^3 \pm 1.65 \times 10^2$. As a result of running 25 MCNP calculations with the initial DXTRAN parameters,

the FOM obtained is $7.09 \times 10^3 \pm 8.05 \times 10^1$. As such, optimization provided a 56% increase in FOM.

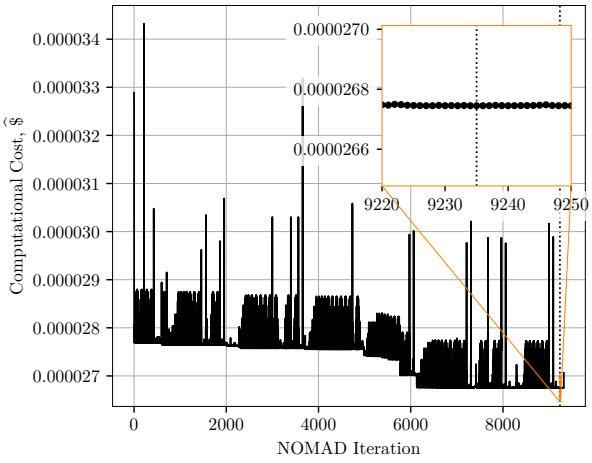
In this case COVRT predicted a 29% improvement in computational cost but a 56% improvement in FOM is realized. The optimized DXTRAN size is much larger than the guessed values. In particular, the DXTRAN region occupies the right-half of the duct and penetrates substantially into the shield. However, this extension in the duct and into the shield is in the direction of the source. The optimization appears to be using DXTRAN to draw particles toward the tally both through the duct and the shield. In addition, the $\beta_c$ values are reduced for regions optically far from the tally; however, a smooth gradient is not observed (within the coarseness of the spatial mesh that $\beta_c$ can be defined on). Along with the comments in Section 7.1.8, it would be valuable to refine the spatial mesh that $\beta_c$ can be defined on to better assess the effect of varying $\beta_c$ values.




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$ and Predicted Optimal DXTRAN Size/Position

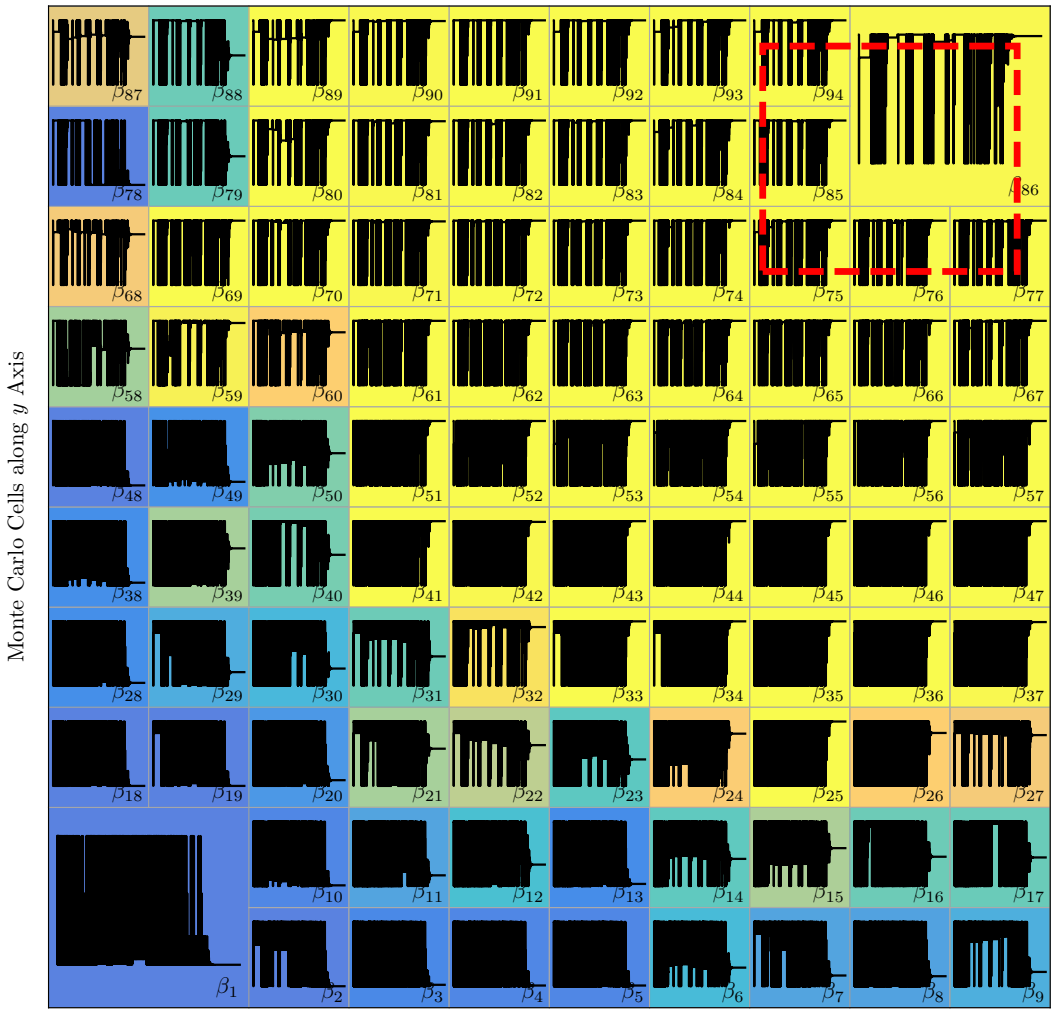



**(b)** DXTRAN Boundary

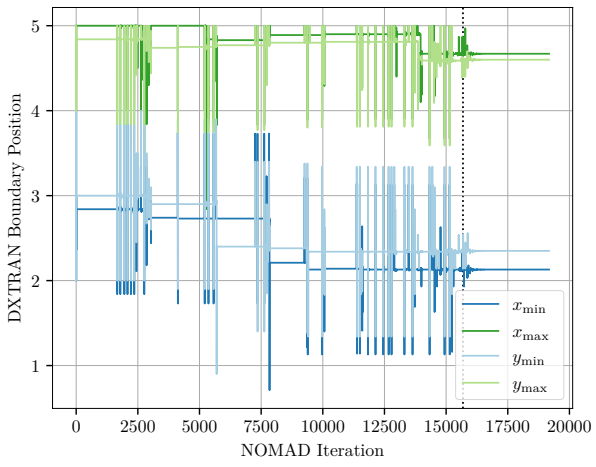



**(c)** Computational Cost Function

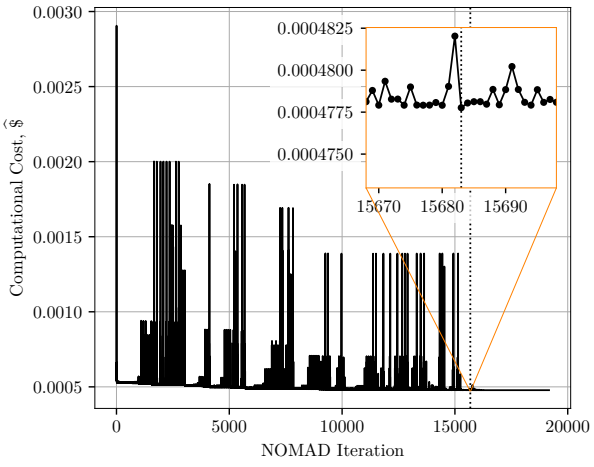
**Figure 7.8:** Test Case 2-2 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$ and Predicted Optimal DXTRAN Size/Position

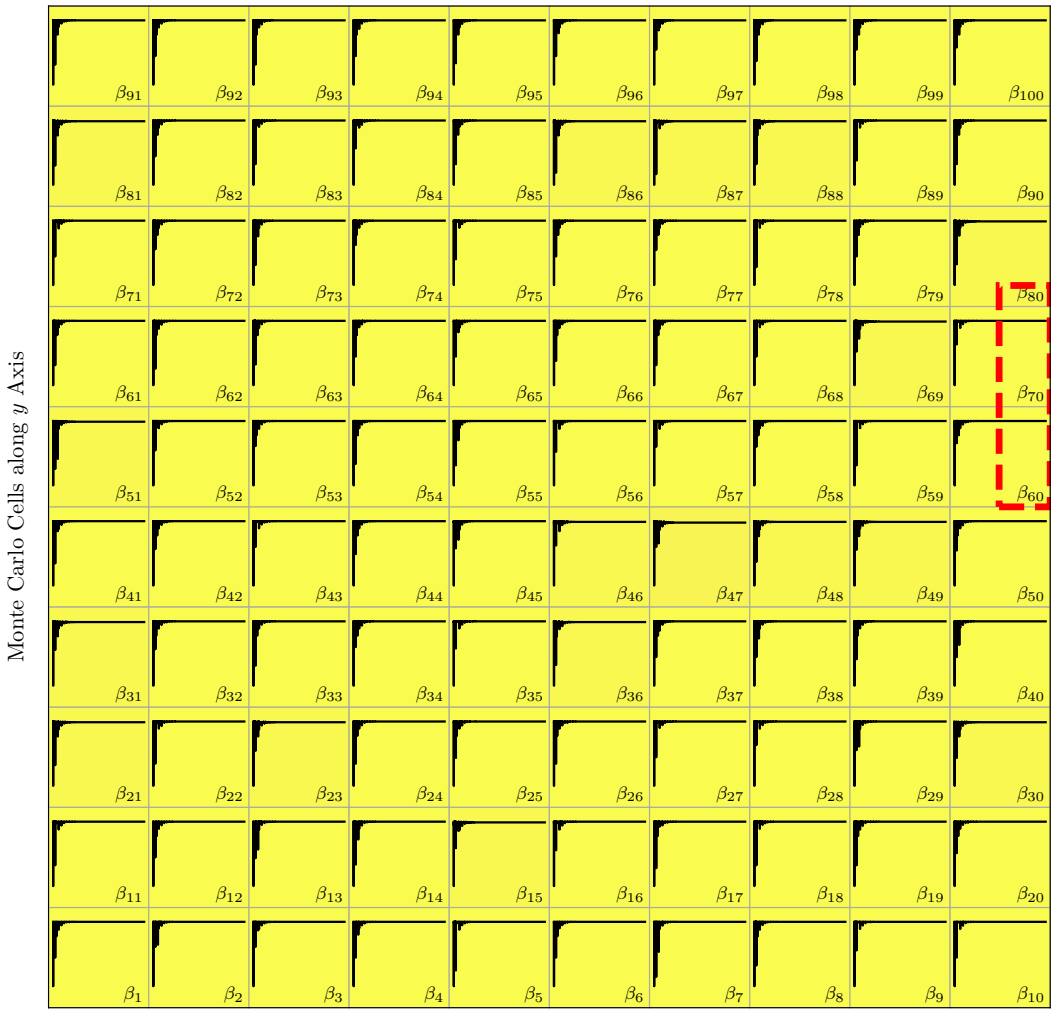



**(b)** DXTRAN Boundary

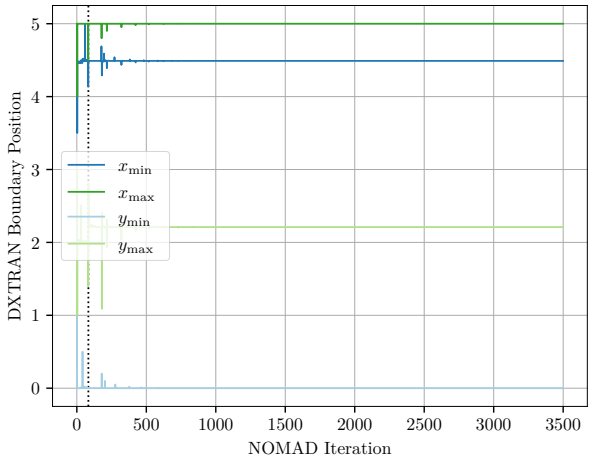



**(c)** Computational Cost Function

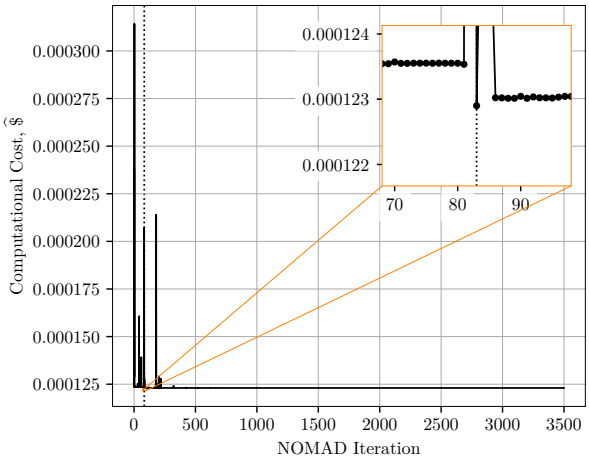
**Figure 7.9:** Test Case 2-3 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$ and Predicted Optimal DXTRAN Size/Position

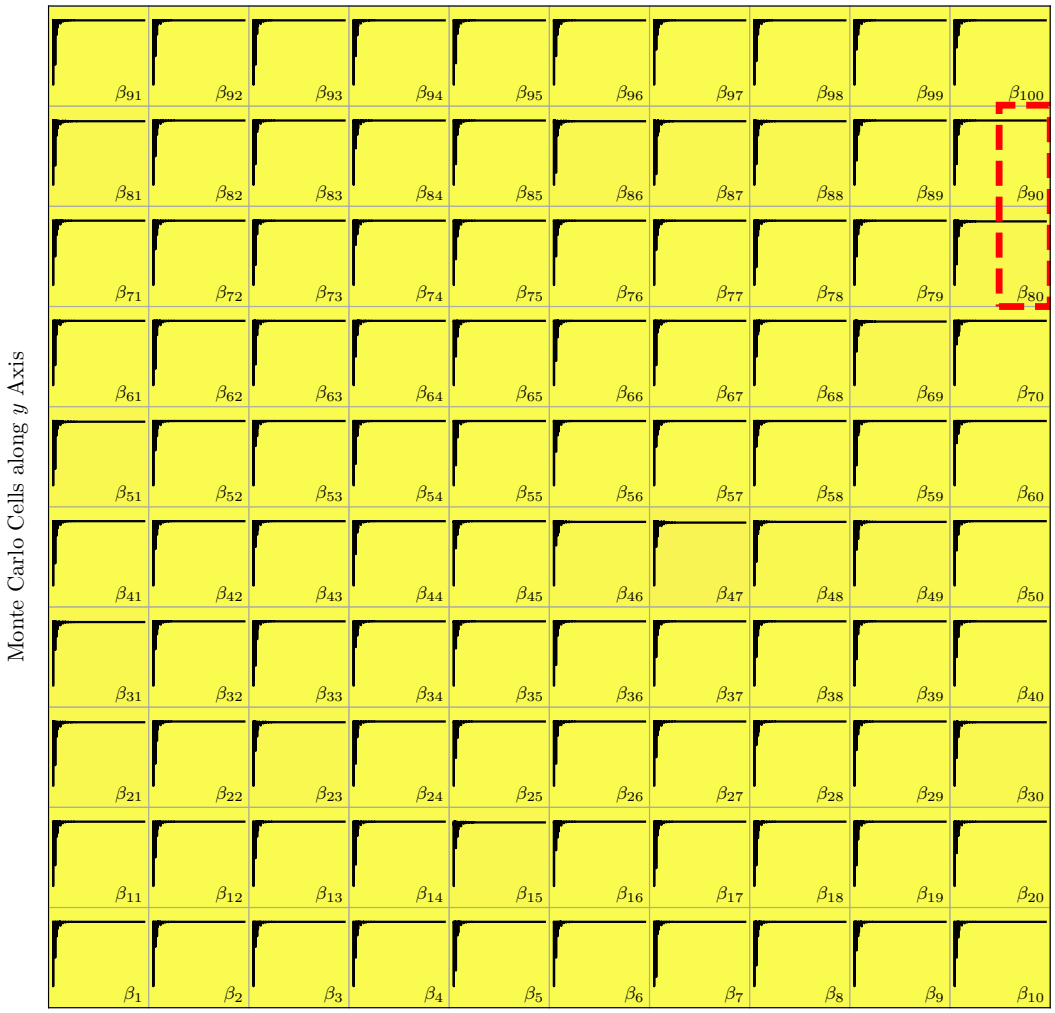



**(b)** DXTRAN Boundary

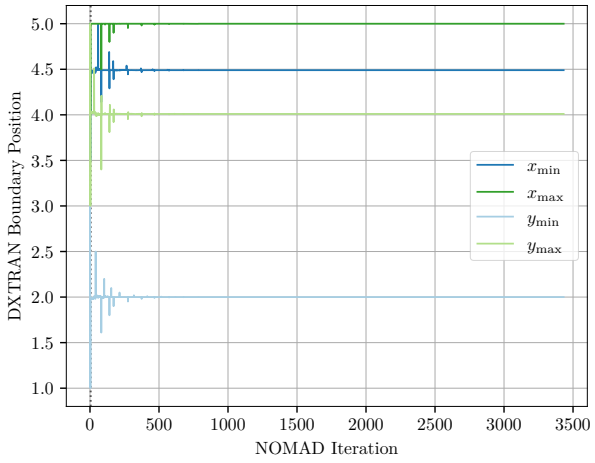



**(c)** Computational Cost Function

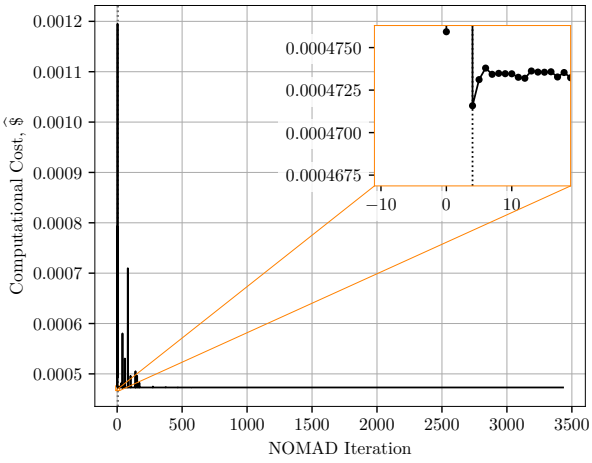
**Figure 7.10:** Test Case 2-4 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$ and Predicted Optimal DXTRAN Size/Position

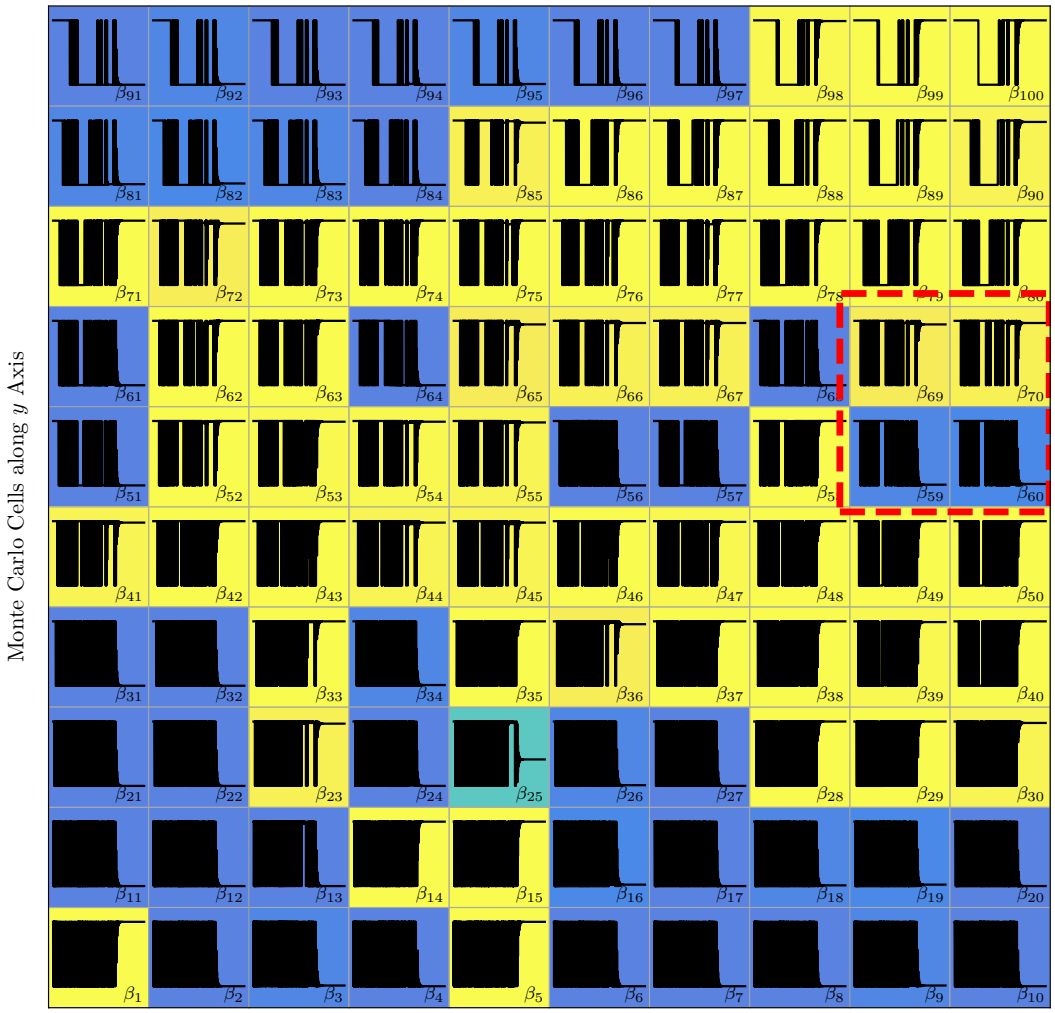



**(b)** DXTRAN Boundary

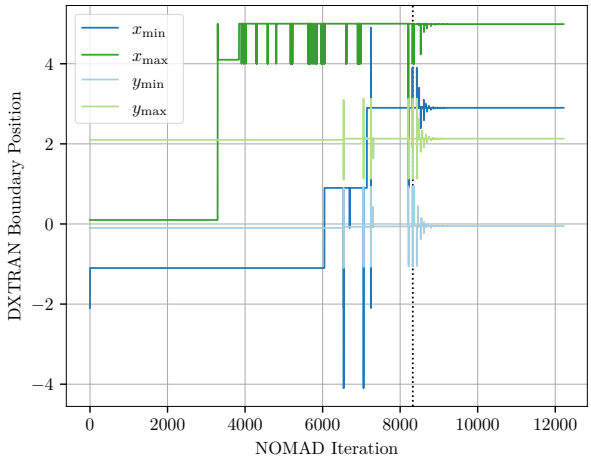



**(c)** Computational Cost Function

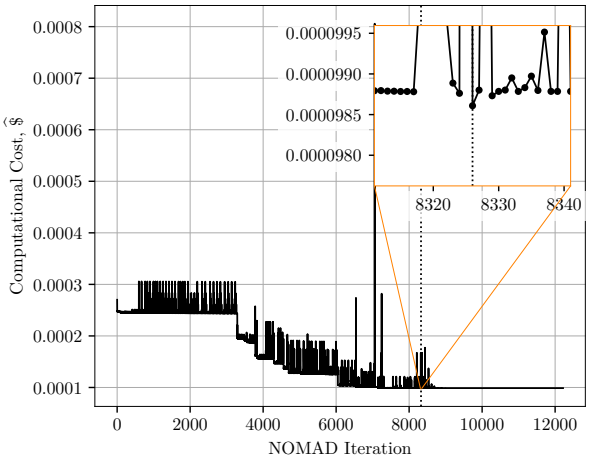
**Figure 7.11:** Test Case 2-5 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$ and Predicted Optimal DXTRAN Size/Position

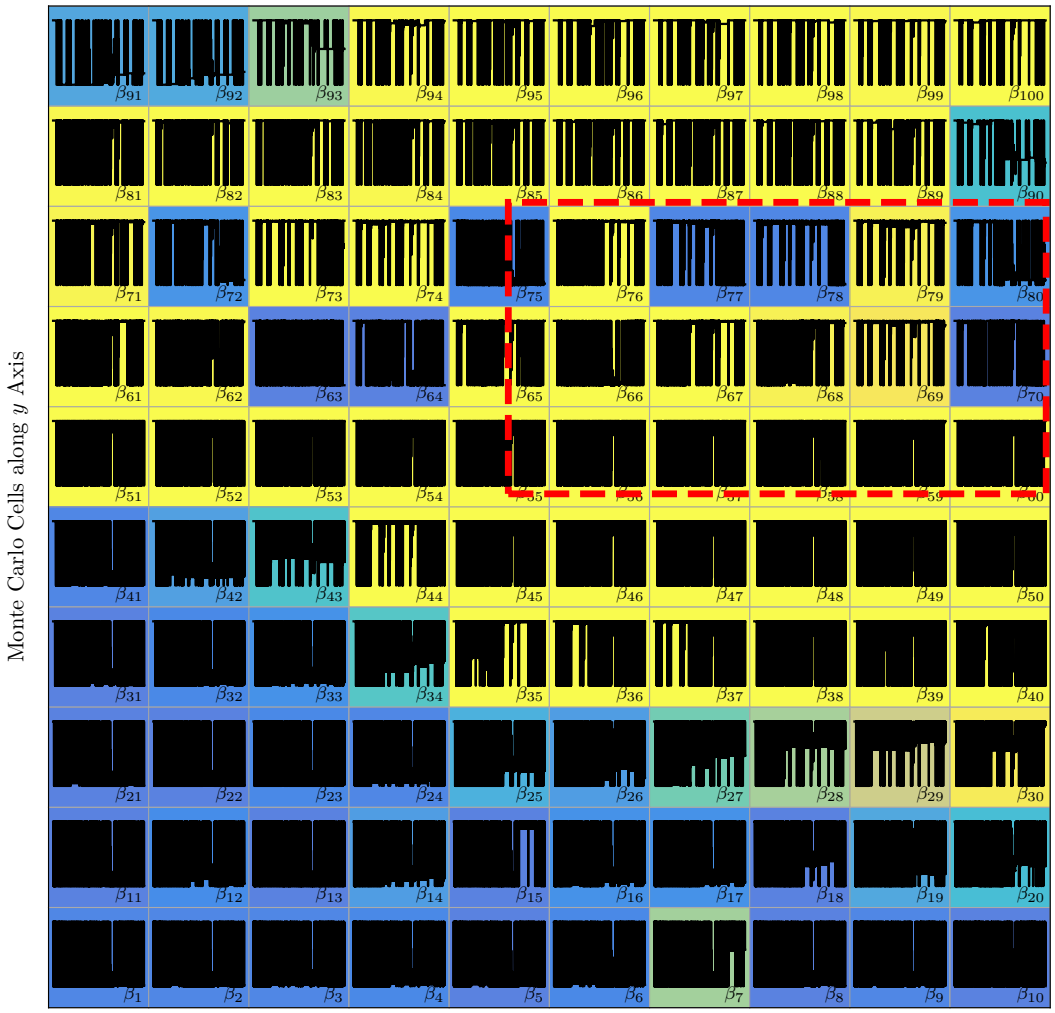



**(b)** DXTRAN Boundary

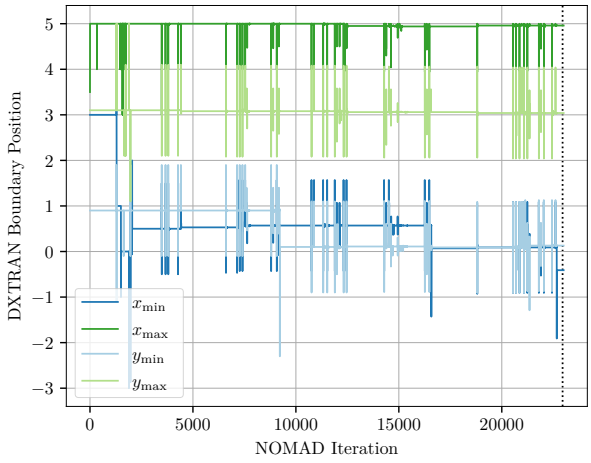



**(c)** Computational Cost Function

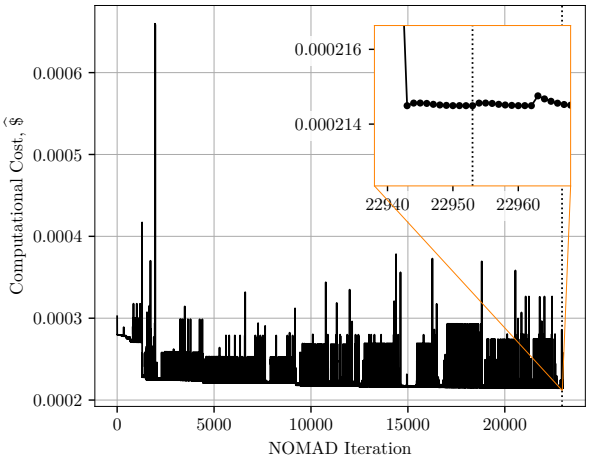
**Figure 7.12:** Test Case 2-6 Optimization Evolution




**(a)** Monte Carlo Cell-wise DXTRAN Rouletting Parameter $\beta_c$ and Predicted Optimal DXTRAN Size/Position




**(b)** DXTRAN Boundary




**(c)** Computational Cost Function

**Figure 7.13:** Test Case 2-7 Optimization Evolution

## 7.2 Summary of Optimization Results

Six 1-D and six 2-D optimization calculations are made. As long as non-trivial change in FOM is predicted by the optimizer, the optimizer identified improved DXTRAN parameters relative to the initial guess in all but one case. This is despite the angular coarseness used to permit quick-running 2-D computational cost calculations by the optimizer. For the 1-D calculations, the actual behavior is generally closer to the predicted behavior; however, there is still not perfect agreement because of the variability in execution timing. For the 2-D calculations, the angular coarseness caused greater disagreement in the actual and predicted benefit of the optimized DXTRAN parameters.

Despite being assessed as the most robust algorithm tested (see Chapter 4), the MADS algorithm implemented in the NOMAD software still seems somewhat sensitive to the quality of the initial guess. That is, if a "good" initial guess is made the optimizer seems to not walk very far from it but if a "bad" guess is made the optimizer will search a larger space and find a potentially better guess than the "good" guess. This is demonstrated by Test Cases 2-4 and 2-6.

## 7.3 Guidance Regarding the Application of DXTRAN

Throughout the course of developing the theory of this work, implementing that theory, and considering the results of test cases to verify the implementation, some general observations are available. This section is meant to capture and communicate those observations for the benefit of future researchers and users of the MCNP DXTRAN variance-reduction technique. Note that this guidance may become dated as the MCNP DXTRAN process and algorithms evolve. Naturally, this guidance is specifically applicable to MCNP DXTRAN regions; however, it is likely also relevant to DXTRAN-like variance-reduction techniques in other Monte Carlo radiation transport codes.

Perhaps the first piece of overriding guidance is that regardless of physical intuition it is valuable to perform at least scoping calculations with a cost-optimizing method. In several of the test cases executed for this work, the DXTRAN was sized and the $\beta_c$ values were set in ways that are not consistent with prevailing guidance. However, even with coarse COVRT calculations the FOM improvement prediction is generally accurate from at least a benefit/detriment standpoint (only Test Case 1-13 predicted improvement but provided inferior DXTRAN parameters).

### 7.3.1 Using DXTRAN as a "Magnet"

The most common application of a DXTRAN region is as a "magnet" where it is used to draw particles toward a region of interest (i.e., a tally). The prevailing guidance is generally that the DXTRAN region should encompass the tally. In this configuration, the DXTRAN region can draw weight to and directed toward the tally regardless of where the DXTRAN process is initiated relative to the tally. In every optimization calculation, the DXTRAN region is used in this way. If the DXTRAN region initially closely encompasses the tally region it is not significantly moved away or expanded (e.g., Test Cases 2-3, 2-4, and 2-5). However, if the DXTRAN region does not closely encompass the region the optimizer moves and resizes the DXTRAN region to do so (e.g., Test Cases 2-2, 2-6, and 2-7).

In addition, prevailing guidance notes that the tally should not exist both inside and outside the DXTRAN region. If this guidance is not followed it is possible that the relatively low-weight DXTRAN particles that contribute to the tally within the DXTRAN region will be very different in weight than the particles that contribute to the tally outside the DXTRAN region. This weight fluctuation can lead to an increased variance. However, the optimizer illustrates an interesting exception to this guidance: if the contributions to the tally are from a preferential direction the DXTRAN region can be positioned to draw particles toward the tally such that contributions to the tally are made by particles in the DXTRAN region and those streaming through and out of it. For example, Test Case 2-3 has an offset DXTRAN region relative to the expected track-length tally region where the tally exists in the shadow of the DXTRAN region. This is effective because there are few relatively high-weight non-DXTRAN particles capable of contributing to the tally that would induce undesirable weight fluctuation.

### 7.3.2 Using DXTRAN as a "Shield"

No optimization calculation suggested that the DXTRAN is most effective applied as a shield. In particular, Test Cases 2-6 and 2-7 specifically positioned the DXTRAN to act as a shield and the optimizer moved it out of that role to encompass the tally region. In particular, because of the offset source-duct configuration in Test Case 2-7, the optimizer positioned the DXTRAN region to draw particles toward the tally both through the duct and through the shield. As a result, optimization does not indicate that using a DXTRAN solely as a shield is of greater benefit than using it as a magnet and taking advantage of the shielding effect implicitly. However, none of the 2-D calculations incorporate other variance-reduction techniques to induce weight fluctuation that a DXTRAN region, acting as a shield, might benefit. Additional investigation into the use of DXTRAN with other variance-reduction techniques is left as future work.

### 7.3.3 DXTRAN Particle Rouletting

Reviewing the optimization calculations two significant trends are identified. First, the optimizer often struggled to identify optimum rouletting values with two distinct behaviors exhibited. In Test Cases 2-4 and 2-5, where the DXTRAN region was set initially to closely encompass the tally region, the value of $\beta_c$ was kept as one for all cells and the DXTRAN region was not modified significantly. As such, the optimizer did not identify a computational cost savings by rouletting particles in these cases despite optically thick regions between the forward source and tally. This behavior is counter to Test Cases 2-2, 2-3, 2-6, and 2-7, where the $\beta_c$ values generally diminished for cells moving optically far from the tally region. Indeed, for Test Cases 2-4 and 2-6 which are otherwise identical except for the initial DXTRAN size and position, this discrepant behavior is surprising. Furthermore, the results of Chapter 4 suggest that the increase in variance as a result of modifying $\beta_c$ from its default value of one is often not offset by the commensurate decrease in time.

Based on these somewhat discrepant findings, an opportunity for future work exists to perform a sensitivity study on material optical thickness to identify trends in the $\beta$ parameters in various configurations of interest. As a part of that study, it would be value to further explore the MADS algorithm, and others, to see if more consistent behavior can be achieved. Until then, the results of this work suggests that without good cause the value of $\beta$ should be left at the default: one.

# Chapter 8

# Conclusions and Future Work

## 8.1 Summary & Conclusions

This work provides the first known deduction and application of the integral transport kernels for both biasing with DXTRAN particle production (Section 3.7.1) and the associated free-flight transport with truncation of the initiating particle (Section 3.7.2). This work applies these new DXTRAN transport kernels to derive expressions for the mean tally response in Monte Carlo transport calculations involving DXTRAN. These expressions are then used to rigorously prove, for the first time, that the technique is unbiased (Section 3.7.4). This work also derives equations for the variance [Section 3.7.5] and expected computational time [Section 4.4] for Monte Carlo calculations involving DXTRAN, which are solved using the deterministic discrete ordinates method.

To verify the statistical moment derivations developed in this work, fourteen 1-D and seven 2-D test case calculations are made [Chapter 6]. Within these test cases, a variety of scenarios are examined that lead to highly angle dependent solutions for which other variance-reduction techniques that do not directly bias particle direction are challenged. For the 1-D test cases, if only DXTRAN variance reduction is used, then the Monte Carlo and deterministic results agree within 1.4%, and if DXTRAN is used in concert with importance splitting and rouletting, the agreement is within 3.5%. This indicates that the expressions derived in this work are implemented correctly. Furthermore, the 2-D test cases generally agree within 15%, which is a consistent level of agreement with prior work in the area of deterministically predicting Monte Carlo variance in 2-D geometries. More importantly, this level of agreement is sufficient for performing computational cost optimization calculations. The expected computational time derivations are validated in Section 4.5.1, which show consistent relative behavior with Monte Carlo calculations when other factors such as varying computer resource usage are considered.

Of the verification test cases, six 1-D and six 2-D test cases are processed using an automated optimization workflow to determine optimal DXTRAN variance reduction parameters [Chapter 7]. In 11 of the 12 optimization test cases, automated optimization provided DXTRAN parameters that improved the Monte Carlo FOM relative to the base calculations (or stayed consistent, suggesting an optimal initial guess was made). This verifies that the deterministic computational cost calculation suitably estimates the Monte Carlo computational cost (inverse FOM) function. Furthermore, this consistent behavior comes in spite of random variation in the Monte Carlo calculation times and a reasonably coarse angular quadrature necessary to have sufficiently fast deterministic calculations as a part of the optimization algorithm.

## 8.2 Future Work

Throughout this work opportunities for future investigation are identified. Those opportunities are categorized into near-term and long-term, and are summarized next. The near-term items for future investigation arise from observations made in this work. The long-term proposal for future work focuses on how the methods developed in this work could be made available for use in production calculations.

### 8.2.1 Near-term

To limit the scope of this work, only a single DXTRAN region is considered. However, analyzing multiple DXTRAN regions and/or wholly nested DXTRAN regions provides a natural extension of this work. Multiple nested DXTRAN regions can be used to mitigate the effect of DXTRAN particles with relatively high weights being generated from events just outside the surface of a DXTRAN region and inducing weight fluctuation in a tally within the DXTRAN region. It is unclear what the optimal size and placement of multiple nested DXTRAN regions should be.

In addition, if multiple non-nested DXTRAN regions are used, then there exists an opportunity for them to "communicate" where collisions in one region will generate DXTRAN particles in the other region(s) and vice versa. This can lead to long-running Monte Carlo calculations with many low-weight DXTRAN particles. DXTRAN weight cutoffs act to minimize the time penalty in this circumstance, but it is unclear how to define optimum weight cutoffs. Correspondingly, the introduction of weight cutoffs make DXTRAN a weight-dependent game. Investigating DXTRAN as a weight-dependent game is avoided here in order to focus on proving that DXTRAN can be characterized deterministically but such investigation in the future is valuable.

An analysis of DXTRAN spheres defined using both inner and outer spheres would be useful to confirm whether such capability provides value. If not, the associated logic could be eliminated to provide a savings in computer time. If the inner biasing sphere can be shown to provide a benefit, identifying how to specify the optimum size of the inner sphere relative to the outer sphere would be valuable. In a departure from the current MCNP behavior where the inner sphere is five times as important as the outer annulus, it would also be useful to identify an optimum importance weighting for the inner region.

Consistent with prior work the timing kernels in the FTE are taken as constant regardless of phase space. In addition, the time taken for MCNP tallies is treated as the same regardless of whether the tally is a current or expected track-length tally. When optimization is performed in this work the deterministic computational cost estimates consistently disagree to a greater degree with the observed Monte Carlo behavior when expected track-length tallies are used versus current tallies. This suggests that tally-specific timing kernels could improve the accuracy of the deterministic computational cost estimate and should be investigated further. Alternatively, short profiling calculations can be made for the scenario of interest to estimate the associated timing, though this approach puts more burden on the analyst.

In this work several optimization algorithms are investigated, and the MADS algorithm is selected for all automated optimization calculations. This is because of its suitable speed to solution balanced with apparent

robustness to poor initial guesses based on a couple simple test calculations. However, in the 2-D optimization calculations two potential aberrant behaviors were observed: (1) the MADS algorithm may spend many iterations searching for improved parameters with little change in the parameters or objective function and (2) the MADS algorithm, if given a sufficiently good initial guess, may not stray far from the guess despite a significantly better solution being available, i.e., the optimizer gets stuck in a local minimum. Additional work to better understand how to overcome these shortcomings and/or to identify a superior optimization algorithm that accommodates bounded and constrained optimization problems is of value.

As a result of the optimization calculations the user-assigned DXTRAN rouletting parameters often diminish as their optical distance from the DXTRAN region increase. A sensitivity study to characterize how they should optimally diminish to provide general guidance to DXTRAN users would be valuable. Furthermore, in several test cases all cell-wise rouletting parameters are unchanged, so a better understanding of, and general guidance for, the application of such parameters independent of optimization is of value. Finally, it would be instructive to extend the rouletting parameter to depend on additional components of phase space, and in particular, on particle energy.

A current limitation of the MCNP DXTRAN variance-reduction technique is that it cannot be used with reflective boundaries. The same limitation exists for point detectors. While devising test cases for this work and in this author's experience with other analyses, this limitation is frequently encountered. Additional effort to identify a way to overcome this limitation would be valuable.

While preparing the verification cases described in Chapter 6, a series of quadrature sets were investigated. Ultimately, the Triangular Gauss-Chebyshev quadrature was selected. However, one quadrature known to be generally well behaved but not explored beyond a cursory investigation is the quadruple range quadrature [98, 99]. A brief investigation was made but because of the combinatoric nature of the quadrature set a thorough investigation would have taken unreasonably long versus the potential benefit versus the Triangular Gauss-Chebyshev quadrature used. Nevertheless, such a thorough investigation has value. It is hoped that such a quadrature might permit using fewer quadrature directions while minimizing ray effects.

Finally, with the completion of this work the only remaining MCNP variance-reduction technique not analyzed using the HSMEs is the forced collision technique. Thus, such an analysis is a logical next step. Furthermore, because DXTRAN is (a) often used in optically thin media and (b) is initiated at non-absorptive collision sites, an analysis of the complementary effect of forced collisions and DXTRAN is valuable.

### 8.2.2 Long-term

In the longer term, it would be valuable to combine the HSME-based approach to generating variance-reduction parameters for various variance-reduction techniques with an established, and faster-to-calculate, method such as FW-CADIS. The FW-CADIS method provides a quick and systematic method to generate weight window boundaries to induce splitting and thus populate phase space leading to a reduction in tally variance. However, it has been shown that such a method can lead to inefficient Monte Carlo calculations versus a HSME-based approach [125] because of the commensurate increase in calculation time.

This author proposes that a hybrid radiation transport software product be developed to generate Monte

Carlo variance-reduction parameters based on FW-CADIS by default but having available an HSME- and FTE-based computational cost solution as an option. In this configuration, a user can quickly produce automated weight window boundaries to populate phase space and accelerate Monte Carlo tally convergence. However, if the weight windows are observed creating long-running histories as a result of over-splitting histories, then the user can re-run the deterministic calculation, enabling the HSME and FTE solutions to change the variance reduction objective function from that of a minimum variance to that of a minimum computational cost. This set of deterministic calculations will likely take longer to run than the FW-CADIS calculation; however, using a computational cost objective function should ameliorate the long-running histories and produce a more-efficient Monte Carlo calculation.

## References


1. 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, Los Alamos, NM, USA, Tech. Rep. LA-UR-03-1987, Feb. 2008.

2. C. E. Burgart and P. N. Stevens, "A General Method of Importance Sampling the Angle of Scattering in Monte Carlo Calculations," *Nuclear Science and Engineering*, vol. 42, no. 3, pp. 306–323, Dec. 1970. DOI: 10.13182/NSE70-A21220

3. E. A. Straker, "Time-dependent Neutron and Secondary Gamma-ray Transport in Infinite Air and in Air Over Ground," Oak Ridge National Laboratory, Oak Ridge, TN, USA, Tech. Rep. ORNL-TM-2781, May 1970.

4. V. R. Cain and K. D. Franz, "Comparison of Monte Carlo Calculations with Measurements of Fast-neutron Dose Transmitted from a Beam Source Through a SNAP-2 LiH Shield," Oak Ridge National Laboratory, Oak Ridge, TN, USA, Tech. Rep. ORNL-TM-2423, Dec. 1968.

5. F. A. R. Schmidt, "Biased Angle Selection in Monte Carlo Shielding Calculations," *Nuclear Engineering and Design*, vol. 15, pp. 209–231, 1971. DOI: 10.1016/0029-5493(71)90064-1

6. J. S. Tang, T. J. Hoffman, and P. N. Stevens, "Angular Biasing of the Collision Process in Multigroup Monte Carlo Calculations," *Nuclear Science and Engineering*, vol. 64, no. 4, pp. 837–842, Dec. 1977. DOI: 10.13182/NSE77-A14498

7. J. S. Tang, P. N. Stevens, and T. J. Hoffman, "Methods of Monte Carlo Biasing using Two-dimensional Discrete Ordinates Adjoint Flux," Oak Ridge National Laboratory, Oak Ridge, TN, USA, Tech. Rep. ORNL/TM-5414, Jun. 1976.

8. T. Vergnaud, G. Dejonghe, J. Gonnord, and J. C. Nimal, "Biasing Techniques in TRIPOLI-2 System Using the Monte Carlo Method in Three-dimensional Geometries," in *Monte-Carlo Methods and Applications in Neutronics, Photonics and Statistical Physics*, R. Alcouffe, R. Dautray, A. Forster, G. Ledanois, and B. Mercier, Eds. Springer-Verlag, 1985, pp. 71–82.

9. J. C. Nimal and T. Vergnaud, "TRIPOLI: A General Monte Carlo Code, Present State and Future Prospects," *Progress in Nuclear Energy*, vol. 24, no. 1–3, pp. 195–200, Sep. 1990. DOI: 10.1016/0149-1970(90)90036-5

10. J. P. Both, J. C. Nimal, and T. Vergnaud, "Automated Importance Generation and Biasing Techniques for Monte Carlo Shielding Techniques by the TRIPOLI-3 Code," *Progress in Nuclear Energy*, vol. 24, no. 1–3, pp. 273–281, Sep. 1990. DOI: 10.1016/0149-1970(90)90046-8

11. B. Morillon, J. P. Both, and J. C. Nimal, "Importance Function by Collision Probabilities for Monte Carloby Code TRIPOLI," in *Mathematical Methods and Supercomputing in Nuclear Applications*, Karlsruhe, Germany. April 19–23, 1993, pp. 745–755, also CEA Report CEA-CONF-11277.

12. T. Vergnaud, J. C. Nimal, and J. P. Both, "TRIPOLI-3: A Monte Carlo Code with a Powerful and Automatic Biasing," in *Mathematical Methods and Supercomputing in Nuclear Applications*, 1993, pp. 756–765, also CEA Report CEA-CONF-11279.

13. J. P. Both, H. Derriennic, B. Morillon, and J. C. Nimal, "A Survey of TRIPOLI-4," in *Proceedings of the 8th International Conference on Radiation Shielding*, 1994, pp. 373–381, also CEA Report CEA-CONF-11776.

14. K. N. Abotel, E. W. Larsen, and W. R. Martin, "A "Local" Exponential Transform Method for Monte Carlo Transport Calculations," in *Proceedings of the International Topical Meeting Advances in Mathematics, Computations, and Reactor Physics*, Pittsburgh, PA, USA. April 29–May 2, 1991.

15. K. N. Abotel, E. W. Larsen, and W. R. Martin, ""Local" Exponential Transform Methods for the Monte Carlo Simulation of Multigroup Transport Problems," in *Proceedings of the 1992 Topical Meeting on Advances in Reactor Physics*, vol. 2, Charleston, SC, USA. March 8–11, 1992, pp. 2–288–2–299.

16. K. N. Abotel, ""Local" Exponential Transform Methods for the Monte Carlo Simulation of Multigroup Transport Problems," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 1993.

17. R. S. Baker and E. W. Larsen, "A "Local" Exponential Transform Method for Global Variance Reduction in Monte Carlo Transport Problems," in *Mathematical Methods and Supercomputing in Nuclear Applications*, Karlsruhe, Germany. April 19–23, 1993.

18. K. A. Van Riper, T. J. Urbatsch, P. D. Soran, K. Parsons, J. E. Morel, G. W. McKinney, S. R. Lee, L. A. Crotzer, R. Alcouffe, F. W. Brinkley, T. E. Booth, and J. Anderson, "AVATAR — Automatic Variance Reduction In Monte Carlo Calculations," Los Alamos National Laboratory, Tech. Rep. LA-UR-97-0919, 1997.

19. S. A. Turner and E. W. Larsen, "Automatic Variance Reduction for Three-Dimensional Monte Carlo Simulations by the Local Importance Function Transform—I: Analysis," *Nuclear Science and Engineering*, vol. 127, no. 1, pp. 22–35, Sep. 1997. DOI: 10.13182/NSE127-36

20. S. A. Turner and E. W. Larsen, "Automatic Variance Reduction for Three-Dimensional Monte Carlo Simulations by the Local Importance Function Transform—II: Numerical Results," *Nuclear Science and Engineering*, vol. 127, no. 1, pp. 36–53, Sep. 1997. DOI: 10.13182/NSE127-36

21. S. A. Turner, "Automatic Variance Reduction for Monte Carlo Simulations via the Local Importance Function Transform," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 1997.

22. J. C. Wagner and A. Haghighat, "Automated Variance Reduction Of Monte Carlo Shielding Calculations Using The Discrete Ordinates Adjoint Function," *Nuclear Science and Engineering*, vol. 128, no. 2, pp. 186–208, Feb. 1998. DOI: 10.13182/NSE98-2

23. J. C. Wagner, "Acceleration of Monte Carlo Shielding Calculations with an Automated Variance Reduction Technique and Parallel Processing," Ph.D. dissertation, Pennsylvania State University, State College, PA, USA, 1997.

24. M. A. Cooper and E. W. Larsen, "Automated Weight Windows for Global Monte Carlo Particle Transport Calculations," *Nuclear Science and Engineering*, vol. 137, no. 1, pp. 1–13, Jan. 2001. DOI: 10.13182/NSE00-34

25. M. A. Cooper, "An Automated Variance Reduction Method for Global Monte Carlo Neutral Particle Transport Problems," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 1999.

26. T. L. Becker, A. B. Wollaber, and E. W. Larsen, "A Hybrid Monte Carlo-Deterministic Method for Global Particle Transport Calculations," *Nuclear Science and Engineering*, vol. 155, no. 2, pp. 155–167, Feb. 2007. DOI: 10.13182/NSE07-A2653

27. A. B. Wollaber and E. W. Larsen, "A Hybrid Monte Carlo-deterministic Method for Global Deep Penetration Particle Transport Calculations," in *Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications*, Palais des Papes, Avignon, France; September 12–15, 2005.

28. T. L. Becker, "Hybrid Monte Carlo/Deterministic Methods for Radiation Shielding Problems," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 2009.

29. J. C. Wagner, E. D. Blakeman, and D. E. Peplow, "Forward-Weighted CADIS Method for Global Variance Reduction," in *Transactions of the American Nuclear Society*, vol. 97, 2007, pp. 630–633.

30. J. C. Wagner, D. E. Peplow, and S. W. Mosher, "FW-CADIS Method for Global and Regional Variance Reduction of Monte Carlo Radiation Transport Calculations," *Nuclear Science and Engineering*, vol. 176, no. 1, pp. 37–57, Jan. 2014. DOI: 10.13182/NSE12-33

31. M. Munk, R. N. Slaybaugh, T. M. Pandya, S. R. Johnson, and T. M. Evans, "FW/CADIS-Ω: An Angle-informed Hybrid Method for Deep-penetration Radiation Transport," in *PHYSOR 2016: Unifying Theory and Experiments in the 21st Century*. Sun Valley, ID, USA; May 1–5, 2016: American Nuclear Society, 2016. URL: https://arxiv.org/abs/1612.00793

32. M. Munk, "FW/CADIS-Ω: An Angle-Informed Hybrid Method for Neutron Transport," Ph.D. dissertation, University of California, Berkeley, Berkeley, CA, USA, 2017.

33. R. R. Coveyou, V. R. Cain, and K. J. Yost, "Adjoint And Importance In Monte Carlo Applications," *Nuclear Science and Engineering*, vol. 27, pp. 219–234, Feb. 1967. DOI: 10.13182/NSE67-A18262

34. H. J. Amster and M. J. Djomehri, "Prediction Of Statistical Error In Monte Carlo Transport Calculations," *Nuclear Science and Engineering*, vol. 60, no. 2, pp. 131–142, Jun. 1976. DOI: 10.13182/NSE76-A26869

35. T. E. Booth and H. J. Amster, "Prediction Of Monte Carlo Errors By A Theory Generalized To Treat Track-Length Estimators," *Nuclear Science and Engineering*, vol. 65, no. 2, pp. 273–281, Feb. 1978. DOI: 10.13182/NSE78-A27156

36. C. J. Everett and E. D. Cashwell, "Cost of Splitting in Monte Carlo Transport," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-7189-MS, Mar. 1978.

37. I. Lux, "Systematic Study Of Some Standard Variance Reduction Techniques," *Nuclear Science and Engineering*, vol. 67, no. 3, pp. 317–325, Sep. 1978. DOI: 10.13182/NSE78-A27252

38. I. Lux, "Note On Prediction Of Monte Carlo Errors," *Nuclear Science and Engineering*, vol. 67, no. 1, pp. 139–140, Jul. 1978.

39. I. Lux and L. Koblinger, *Monte Carlo Particle Transport Methods: Neutron And Photon Calculations*. CRC Press, 1991.

40. P. K. Sarkar and M. A. Prasad, "Prediction Of Statistical Error And Optimization Of Biased Monte Carlo Transport Calculations," *Nuclear Science and Engineering*, vol. 70, no. 3, pp. 243–261, Jun. 1979. DOI: 10.13182/NSE79-A20146

41. T. E. Booth, "Analytic Score Distributions and Moments for a Spatially Continuous Tridirectional Monte Carlo Transport Problem," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-12570, Jul. 1993.

42. T. E. Booth and E. D. Cashwell, "Analysis Of Error In Monte Carlo Transport Calculations," *Nuclear Science and Engineering*, vol. 71, no. 2, pp. 128–142, Aug. 1979. DOI: 10.13182/NSE79-A20404

43. R. J. Juzaitis, "Predicting The Cost Of Splitting In Monte Carlo Particle Transport," *Nuclear Science and Engineering*, vol. 80, no. 3, pp. 424–447, Mar. 1982. DOI: 10.13182/NSE82-A19829

44. R. J. Juzaitis, "Minimizing the Cost of Splitting in Monte Carlo Radiation Transport Simulation," Ph.D. dissertation, University of Virginia, Charlottsville, VA, USA, 1980.

45. A. Dubi, "On the Analysis of the Variance in Monte Carlo Calculations," *Nuclear Science and Engineering*, vol. 72, no. 1, pp. 108–120, Oct. 1979. DOI: 10.13182/NSE79-A19313

46. A. Dubi and D. J. Dudziak, "Optimal Choice of Parameters for Exponential Biasing in Monte Carlo," *Nuclear Science and Engineering*, vol. 70, no. 1, pp. 1–13, Apr. 1979. DOI: 10.13182/NSE79-A18922

47. A. Dubi, T. Elperin, and D. J. Dudziak, "Geometrical Splitting in Monte Carlo," *Nuclear Science and Engineering*, vol. 80, no. 1, pp. 139–161, Jan. 1982. DOI: 10.13182/NSE82-A21411

48. A. Dubi and D. J. Dudziak, "Extended Model of Geometrical Surface Splitting in Monte Carlo," *Nuclear Science and Engineering*, vol. 83, no. 4, pp. 487–497, Apr. 1983. DOI: 10.13182/NSE83-A18653

49. A. Dubi, "General Statistical Model for Geometrical Splitting in Monte Carlo — I," *Transport Theory and Statistical Physics*, vol. 14, no. 2, pp. 167–193, 1985. DOI: 10.1080/00411458508211675

50. A. Dubi, "General Statistical Model for Geometrical Splitting in Monte Carlo — II," *Transport Theory and Statistical Physics*, vol. 14, no. 2, pp. 196–221, 1985. DOI: 10.1080/00411458508211676

51. A. Dubi, A. Goldfeld, and K. Burn, "Application of the Direct Statistical Approach on a Multisurface Splitting Problem in Monte Carlo Calculations," *Nuclear Science and Engineering*, vol. 93, no. 2, pp. 204–213, Jun. 1986. DOI: 10.13182/NSE86-A17669

52. A. Dubi, N. Gurvitz, and K. Burn, "Optimal Space-energy Splitting in MCNP with the DSA," *Progress in Nuclear Energy*, vol. 24, no. 1–3, pp. 11–12, 1990. DOI: 10.1016/0149-1970(90)90018-Z

53. K. W. Burn, "Optimizing Cell Importances Using an Extension of the DSA — Theory, Implementation, Preliminary Results," *Progress in Nuclear Energy*, vol. 24, no. 1–3, pp. 39–54, 1990. DOI: 10.1016/0149-1970(90)90021-V

54. K. W. Burn, "Complete Optimization of Space/Energy Cell Importances with the DSA Cell Importance Model," *Annals of Nuclear Energy*, vol. 19, no. 2, pp. 65–98, 1992. DOI: 10.1016/0306-4549(92)90025-7

55. K. W. Burn, "Extending The Direct Statistical Approach To Include Particle Bifurcation Between The Splitting Surfaces," *Nuclear Science and Engineering*, vol. 119, no. 1, pp. 44–79, Jan. 1995. DOI: 10.13182/NSE95-A24070

56. K. W. Burn, "A New Weight-Dependent Direct Statistical Approach Model," *Nuclear Science and Engineering*, vol. 125, no. 2, pp. 128–170, Feb. 1997. DOI: 10.13182/NSE97-A24262

57. G. A. Mikhailov, *Minimization of Computational Costs of Non-Analogue Monte Carlo Methods*. World Scientific, 1992. DOI: 10.1142/1440

58. C. J. Solomon, "Discrete-Ordinates Cost Optimization Of Weight-Dependent Variance Reduction Techniques For Monte Carlo Neutral Particle Transport," Ph.D. dissertation, Kansas State University, 2010. URL: http://hdl.handle.net/2097/7014

59. C. J. Solomon, A. Sood, T. E. Booth, and J. K. Shultis, "A Priori Deterministic Computational-Cost Optimization Of Weight-Dependent Variance-Reduction Parameters For Monte Carlo Neutral-Particle Transport," *Nuclear Science and Engineering*, vol. 176, no. 1, pp. 1–36, Jan. 2014. DOI: 10.13182/NSE12-81

60. T. E. Booth, "Analysis of Error in Monte Carlo Transport Calculations," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-7636-T, Jan. 1979.

61. LASL Group TD-6, "MCNP—A General Monte Carlo Code for Neutron and Photon Transport," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-7396-M, Rev. 1, Nov. 1979.

62. LASL Group TD-6, "MCNP—A General Monte Carlo Code for Neutron and Photon Transport," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-7396-M, Jul. 1978.

63. E. D. Cashwell, J. R. Neergaard, W. M. Taylor, and G. D. Turner, "MCN: A Neutron Monte Carlo Code," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-4751, Jan. 1972.

64. T. E. Booth, "Recollections on DXTRAN's Origin," Private Communication, Aug. 2017.

65. H. J. Kalli and E. D. Cashwell, "Evaluation of Three Monte Carlo Estimation Schemes for Flux at at Point," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-6865-MS, Sep. 1977.

66. H. J. Kalli, "Monte Carlo Studies of Neutron Transport in Cold Neutron Facilities," Ph.D. dissertation, Helsinki University of Technology, Otaniemi, Finland, 1974.

67. C. T. Parry, "Nuclear Data and Angular Biasing Aspects of Monte Carlo Shielding Calculations for Fusion Reactors," Ph.D. dissertation, Imperial College of Science and Technology, London, UK, 1985.

68. L. L. Carter, "Neutron Streaming Analysis for Shield Design of FMIT Facility," Hanford Engineering Development Laboratory, Richland, WA, USA, Tech. Rep. HEDL-S/A-2297-FP, Dec. 1981. URL: https://www.osti.gov/biblio/5947696

69. S. Chucas, I. Curl, T. Shuttleworth, and G. Morrell, "Preparing the Monte Carlo Code MCBEND for the 21st Century," in *8th International Conference on Radiation Shielding*, Arlington, TX, USA, 1994, pp. 1–10.

70. T. Shuttleworth, M. Grimstone, and S. Chucas, "Application of Acceleration Techniques in MCBEND," *Journal of Nuclear Science and Technology*, vol. 37, no. sup1, pp. 406–411, Mar. 2000. DOI: 10.1080/00223131.2000.10874916

71. I. Kawrakow, E. Mainegra-Hing, D. W. O. Rogers, F. Tessier, and B. R. B. Walters, "The EGSnrc Code System: Monte Carlo Simulation of Electron and Photon Transport," National Research Council Canada, Ottawa, Canada, Tech. Rep. PIRS-701, Feb. 2017.

72. J. R. Tickner, "Algorithm for Forcing Scattered Radiation to Arbitrary Convex Regions in Neutral Particle Monte Carlo Simulation," *Nuclear Instruments and Methods in Physics Research B*, vol. 267, pp. 2361–2364, 2009. DOI: 10.1016/j.nimb.2009.05.008

73. V. A. Nievaart, D. Légràdy, R. L. Moss, J. L. Kloosterman, T. H. J. J. van der Hagen, and H. van Dam, "Application of Adjoint Monte Carlo to Accelerate Simulations of Mono-directional Beams in Treatment Planning for Boron Neutron Capture Therapy," *Medical Physics*, vol. 34, no. 4, pp. 1321–1335, Apr. 2007. DOI: 10.1118/1.2712573

74. V. A. Nievaart, "Spectral Tailoring for Boron Neutron Capture Therapy," Ph.D. dissertation, Technische Universiteit Delft, Delft, Netherlands, 2007.

75. P. Ferrari, F. Vanhavere, E. Carinou, G. Gualdrini, I. Clairand, M. Sans-Merce, M. Ginjaume, I. Barth, J.-M. Bordy, A. Carnicer, J. Daures, J. Debroas, M. Denoziere, J. Domienik, L. Donadille, E. Fantuzzi, C. Itié, J. Jankowski, C. Koukorava, S. Krim, F. Mariotti, and F. Monteventi, "Challenges on the Radiation Protection Optimization of Medical Staff in Interventional Radiology and Nuclear Medicine: The ORAMED Project," in *Proceedings of the Third European International Radiation Protection Association (IRPA) Congress*. Helsinki, Finland. Jun. 14–18: Nordic Society for Radiation Protection and International Radiation Protection Association, 2010, pp. 1278–1289.

76. A. J. Nelson, G. W. Cooper, C. L. Ruiz, G. A. Chandler, D. L. Fehl, K. D. Hahn, R. J. Leeper, R. M. Smelser, and J. A. Torres, "A Novel Method for Modeling the Neutron Time of Flight (nTOF) Detector Response in Current Mode to Inertial Confinement Fusion Experiments," Sandia National Laboratories, Albuquerque, NM, USA, Tech. Rep. SAND2013-2343, Sep. 2013.

77. R. S. França, "Neutron Flux Characterization and Design of UFTR Radiation Beam Port Using Monte Carlo Methods," Ph.D. dissertation, University of Florida, Gainesville, FL, USA, 2012.

78. T. Schönfeldt, "Advanced Neutron Moderators for the ESS," Ph.D. dissertation, Technical University of Denmark, Kongens Lyngby, Denmark, 2016.

79. T. P. Lou, "Compact D-D/D-T Neutron Generators and Their Applications," Ph.D. dissertation, University of California, Berkeley, Berkeley, CA, USA, 2003.

80. J. A. Caffrey, C. F. Gomez, and L. L. Scharber, "Shielding Development for Nuclear Thermal Propulsion," in *Proceedings of Nuclear and Emerging Technologies for Space (NETS-2015)*, Albuquerque, NM, USA. Jun. 23–26, 2015, pp. 142–151.

81. D. Taylor, S. Lilley, A. Turner, and A. Davis, "Neutron Shielding and Activation of the MASTU Device and Surrounds," *Fusion Engineering and Design*, vol. 89, no. 9, pp. 2076–2082, Oct. 2014. DOI: 10.1016/j.fusengdes.2014.02.035

82. T. E. Booth, K. C. Kelley, and S. S. McCready, "Monte Carlo Variance Reduction Using Nested DXTRAN Spheres," *Nuclear Technology*, vol. 168, no. 3, pp. 765–767, Dec. 2009. DOI: 10.13182/NT09-A9303

83. G. I. Bell and S. Glasstone, *Nuclear Reactor Theory*. New York, NY, USA: Van Nostrand Reinhold Company, 1970, TID-25606.

84. L. L. Carter and E. D. Cashwell, "Particle-Transport Simulation with the Monte Carlo Method," Los Alamos Scientific Laboratory, Los Alamos, NM, USA, Tech. Rep. TID-26607, 1975. DOI: 10.2172/4167844

85. J. J. Duderstadt and W. R. Martin, *Transport Theory*. Wiley, 1979.

86. Y. Ronen, *CRC Handbook Of Nuclear Reactors Calculations*. CRC Press, 1986, vol. 2.

87. R. D. O'Dell and R. E. Alcouffe, "Transport Calculations for Nuclear Analyses: Theory and Guidelines for Effective Use of Transport Codes," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-10983-MS, Sep. 1987.

88. W. F. Miller and E. E. Lewis, *Computational Methods of Neutron Transport*. American Nuclear Society, 1993.

89. W. Dunn and J. K. Shultis, *Exploring Monte Carlo Methods*, 1st ed. Elsevier Science, 2011.

90. E. W. Larsen and B. C. Kiedrowski, "NERS 543: Nuclear Reactor Theory II Lecture Notes," University of Michigan Course Notes, 2018.

91. E. W. Larsen, "NERS 644: Transport Theory Lecture Notes," University of Michigan Course Notes, 2015.

92. J. E. Hoogenboom, "Adjoint Monte Carlo Methods in Neutron Transport Calculations," Ph.D. dissertation, Technische Universiteit Delft, Delft, Netherlands, 1977.

93. J. J. Casal, R. J. J. Stamm'ler, E. Villarino, and A. Ferri, "HELIOS: Geometric Capabilities of a New Fuel Assembly Program," in *Proceedings of the International Topical Meeting on Advances in Mathematics, Computations, and Reactor Physics*. Pittsburgh, PA, USA: American Nuclear Society, 1991, pp. 10.2–1–1 to 10.2–1–13.

94. J. M. Risner, D. Wiarda, M. E. Dunn, T. M. Miller, D. E. Peplow, and B. W. Patton, "Production and Testing of the VITAMIN-B7 Fine-Group and BUGLE-B7 Broad-Group Coupled Neutron/Gamma Cross-Section Libraries Derived from ENDF/B-VII.0 Nuclear Data," Oak Ridge National Laboratory, Oak Ridge, TN, USA, Tech. Rep. NUREG/CR-7045 (ORNL/TM-2011/12), Jan. 2011.

95. B. G. Carlson, "Solution of the Transport Equation by $S_N$ Approximations," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-1599, Oct. 1953.

96. J. J. Jarrell, "An Adaptive Angular Discretization Method for Neutral-particle Transport in Three-dimensional Geometries," Ph.D. dissertation, Texas A&M University, College Station, TX, USA, 2010.

97. J. J. Jarrell and M. L. Adams, "Discrete-ordinates Quadrature Sets Based on Linear Discontinuous Finite Elements," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*. Rio de Janeiro, RJ, Brazil; May 8–12: American Nuclear Society, 2011, pp. 1–23.

98. I. K. Abu-Shumays, "Compatible Product Angular Quadrature for Neutron Transport in $x$-$y$ Geometry," *Nuclear Science and Engineering*, vol. 64, no. 2, pp. 299–316, Oct. 1977. DOI: 10.13182/NSE64-299

99. I. K. Abu-Shumays, "Angular Quadratures for Improved Transport Computations," *Transport Theory and Statistical Physics*, vol. 30, no. 2–3, pp. 169–204, 2001. DOI: 10.1081/TT-100105367

100. F. B. Brown and Y. Nagaya, "The MCNP5 Random Number Generator," in *Transactions of the American Nuclear Society*, vol. 87, no. 230. Washington, DC, USA; November 17–21: American Nuclear Society, 2002.

101. F. B. Brown, "The MCNP5 Random Number Generator," Los Alamos National Laboratory, Los Alamos, NM, USA, Tech. Rep. LA-UR-07-7961, 2007.

102. S. Širca, *Probability for Physicists*, ser. Graduate Texts in Physics. Switzerland: Springer, 2016. DOI: 10.1007/978-3-319-31611-6

103. T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. J. Cox, J. Durkee, J. Elson, M. Fensin, R. A. Forster, J. Hendricks, H. G. Hughes, R. Johns, B. Kiedrowski, R. Martz, S. Mashnik, G. McKinney, D. Pelowitz, R. Prael, J. Sweezy, L. Waters, T. Wilcox, and T. Zukaitis, "Initial MCNP6 Release Overview," *Nuclear Technology*, vol. 180, no. 3, pp. 298–315, Dec. 2012. DOI: 10.13182/NT11-135

104. P. K. Romano, N. E. Horelik, B. R. Herman, A. G. Nelson, B. Forget, and K. Smith, "OpenMC: A State-of-the-art Monte Carlo Code for Research and Development," *Annals of Nuclear Energy*, vol. 82, pp. 90–97, Aug. 2015. DOI: 10.1016/j.anucene.2014.07.048

105. N. Nethercote and J. Seward, "Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation," in *Proceedings of ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (PLDI 2007)*, San Diego, CA, USA, Jun. 2007. DOI: 10.1145/1250734.1250746

106. J. Weidendorfer, M. Kowarschik, and C. Trinitis, "A Tool Suite for Simulation Based Analysis of Memory Access Behavior," in *Proceedings of the 4th International Conference on Computational Science (ICCS 2004)*, Krakow, Poland, Jun. 2004. DOI: 10.1007/978-3-540-24688-6_58

107. M. R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, Dec. 1952. DOI: 10.6028/jres.049.044

108. Y. Saad and M. H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *Journal of Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, Jul. 1986. DOI: 10.1137/0907058

109. C. Audet, S. Le Digabel, and C. Tribes, "NOMAD User Guide," Les Cahiers du GERAD, Tech. Rep. G-2009-37, 2009.

110. S. Le Digabel, "Algorithm 909: NOMAD: Nonlinear Optimization with the MADS Algorithm," *ACM Transactions on Mathematical Software*, vol. 37, no. 4, Feb. 2011. DOI: 10.1145/1916461.1916468

111. E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open Source Scientific Tools for Python," Website, 2018. URL: www.scipy.org

112. D. Kraft, "A Software Package for Sequential Quadratic Programming," *Institut für Dynamik der Flugsysteme der DFVLR*, vol. 88, no. 28, Jul. 1988.

113. M. J. D. Powell, "A Direct Search Optimization Method that Models the Objective and Constraint Functions by Linear Approximation," in *Advances in Optimization and Numerical Analysis — Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, S. Gomez and J.-P. Hennart, Eds. Springer Science+Business Media Dordrecht, 1994, pp. 51–67.

114. D. J. Wales and J. P. K. Doye, "Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms," *The Journal of Physical Chemistry A*, vol. 101, no. 28, pp. 5111–5116, Jul. 1997. DOI: 10.1021/jp970984n

115. R. Storn and K. Price, "Differential Evolution — A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997. DOI: 10.1023/A:1008202821328

116. "Particle Swarm Optimization (PSO) with Constraint Support," Website, 2018. URL: https://pythonhosted.org/pyswarm

117. P. R. Miles, "A Python Program for Running Markov Chain Monte Carlo (MCMC) Simulations: pymcmcstat," Website, 2018. URL: https://prmiles.wordpress.ncsu.edu/codes/python-packages/pymcmcstat/

118. C. Audet and J. E. Dennis, Jr., "Mesh Adaptive Direct Search Algorithms for Constrained Optimization," *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188–217, Jul. 2006. DOI: 10.1137/040603371

119. J. C. Armstrong, Private Communication, Jun. 2018.

120. A. Donev, "Interoperability with C in Fortran 2003," *ACM Fortran Forum,*, vol. 25, no. 1, pp. 8–12, Apr. 2006. DOI: 10.1145/1124708.1124710

121. O. Tange, "GNU Parallel - The Command-Line Power Tool," *The USENIX Magazine*, vol. 36, no. 1, pp. 42–47, Feb. 2011. DOI: 10.5281/zenodo.16303

122. W. H. Reed, "New Difference Schemes for the Neutron Transport Equation," *Nuclear Science and Engineering*, vol. 46, no. 2, pp. 309–314, Nov. 1971. DOI: 10.13182/NSE46-309

123. T. E. Booth, "A Sample Problem For Variance Reduction In MCNP," Los Alamos National Laboratory, Tech. Rep. LA-10363-MS, Oct. 1985.

124. E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Graphics Press, 2001.

125. J. A. Kulesza, C. J. Solomon, B. C. Kiedrowski, and E. W. Larsen, "Performance Assessment of Cost-Optimized Variance Reduction Parameters in Radiation Shielding Scenarios," in *Proceedings of International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering*. Jeju, South Korea; April 16–20: American Nuclear Society, 2017.

126. J. A. Kulesza, C. J. Solomon, and B. C. Kiedrowski, "An Angular Biasing Method Using Arbitrary Convex Polyhedra for Monte Carlo Radiation Transport Calculations," *Annals of Nuclear Energy*, vol. 114, pp. 437–450, Apr. 2018. DOI: 10.1016/j.anucene.2017.12.045

127. C. J. Ong and E. G. Gilbert, "Fast Versions of the Gilbert-Johnson-Keerthi Distance Algorithm: Additional Results and Comparisons," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 531–539, 2001. DOI: 10.1109/70.954768

128. T. M. Chan, "Optimal Output-Sensitive Convex Hull Algorithms in Two and Three Dimensions," *Discrete & Computational Geometry*, vol. 16, no. 4, pp. 361–368, Apr. 1996. DOI: 10.1007/BF02712873

129. R. A. Jarvis, "On the Identification of the Convex Hull of a Finite Set of Points in the Plane," *Information Processing Letters*, vol. 2, no. 1, pp. 18–21, Mar. 1973. DOI: 10.1016/0020-0190(73)90020-3

130. B. O. Rodrigues, "Des Lois Géometriques qui Regissent les Déplacements d'Un Systéme Solide dans L'Espace, et de la Variation des Coordonnées Provenant de ces Déplacement Considérées Indépendent des Causes qui Peuvent les Produire," *Journal de Mathématiques Pures et Appliquées*, vol. 5, pp. 380–440, 1840.

131. M. Cyrus and J. Beck, "Generalized Two- and Three-dimensional Clipping," *Computers & Graphics*, vol. 3, no. 1, pp. 23–28, Jan. 1978. DOI: 10.1016/0097-8493(78)90021-3

132. P. Jiménez, F. Thomas, and C. Torras, "3D Collision Detection: A Survey," *Computers & Graphics*, vol. 25, no. 2, pp. 269–285, Apr. 2001. DOI: 10.1016/S0097-8493(00)00130-8

133. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. U.S. Department of Commerce, NIST, 1972.

134. G. Longoni, "Advanced Quadrature Sets, Acceleration and Preconditioning Techniques for the Discrete Ordinates Method in Parallel Computing Environments," Ph.D. dissertation, University of Florida, Gainesville, FL, USA, 2004.

135. C. Brewer, M. Harrower, B. Sheesley, A. Woodruff, and D. Heyman, "Colorbrewer 2.0," Website, 2018, accessed Oct. 2018. URL: http://www.colorbrewer2.org

## Appendix A

## Forward and Adjoint Operator Interchange Procedure

The following derivation is reproduced with minor modification from University of Michigan Transport Theory Course Notes (course number NERS 644) [91].

One can express the inner product of two functions $f(\boldsymbol{x}, \boldsymbol{\Omega}, E)$ and $f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E)$ and the forward transport migration operator, $M$, as

$$\begin{aligned}\langle f^*, Mf \rangle = \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V\, f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) \Bigg[ & \boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \\ & + \Sigma_\mathrm{t}(\boldsymbol{x}, E) f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \\ & - \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \Sigma_\mathrm{s}(\boldsymbol{x}, \boldsymbol{\Omega}', E' \to \boldsymbol{\Omega}, E) f(\boldsymbol{x}, \boldsymbol{\Omega}', E') \\ & - \frac{\chi(\boldsymbol{x}, E)}{4\pi} \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \nu\Sigma_\mathrm{f}(\boldsymbol{x}, E') f(\boldsymbol{x}, \boldsymbol{\Omega}', E') \Bigg]. \end{aligned} \tag{A.1}$$

The streaming term on the right side can be written as

$$\begin{aligned}& \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V\, \{f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) [\boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f(\boldsymbol{x}, \boldsymbol{\Omega}, E)]\} \\ & \qquad = \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V\, \{\boldsymbol{\Omega} \cdot \boldsymbol{\nabla} [f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) f(\boldsymbol{x}, \boldsymbol{\Omega}, E)] - [\boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E)] f(\boldsymbol{x}, \boldsymbol{\Omega}, E)\}, \end{aligned}$$

where the Divergence Theorem can be applied to obtain the surface and volume terms

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \left\{ \boldsymbol{\Omega} \cdot \boldsymbol{\nabla} \left[ f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \right] - \left[ \boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \right] f \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \right\}$$

$$= \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_{\partial V} \mathrm{d}S \, \boldsymbol{\Omega} \cdot \boldsymbol{n} \left[ f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \right]$$

$$+ \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \left[ -\boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \right] f \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right). \quad \text{(A.2a)}$$

The collision term can be rearranged as

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \, f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \left[ \Sigma_\mathrm{t} \left( \boldsymbol{x}, E \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \right]$$

$$= \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \, f \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \left[ \Sigma_\mathrm{t} \left( \boldsymbol{x}, E \right) f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \right]. \quad \text{(A.2b)}$$

The scattering term can be rearranged by rewriting as

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \, f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) \left[ \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \Sigma_\mathrm{s} \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \to \boldsymbol{\Omega}, E \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \right) \right]$$

$$= \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, \Sigma_\mathrm{s} \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \to \boldsymbol{\Omega}, E \right) f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \right),$$

where $\boldsymbol{\Omega}, E$ and $\boldsymbol{\Omega}', E'$ are interchanged to obtain

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, \Sigma_\mathrm{s} \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \to \boldsymbol{\Omega}, E \right) f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \right)$$

$$= \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, \Sigma_\mathrm{s} \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E' \right) f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right),$$

where the order of integrations are also rearranged to obtain

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, \Sigma_\mathrm{s} \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \to \boldsymbol{\Omega}, E \right) f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \right)$$

$$= \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \, \Sigma_\mathrm{s} \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E' \right) f^* \left( \boldsymbol{x}, \boldsymbol{\Omega}', E' \right) f \left( \boldsymbol{x}, \boldsymbol{\Omega}, E \right),$$

where finally regrouping can occur to obtain

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, \Sigma_\mathrm{s}\left(\boldsymbol{x}, \boldsymbol{\Omega}', E' \to \boldsymbol{\Omega}, E\right) f^*\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right)$$
$$= \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \int_V \mathrm{d}V \left[ \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \, \Sigma_\mathrm{s}\left(\boldsymbol{x}, \boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E'\right) f^*\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right) \right] f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right). \quad \text{(A.2c)}$$

Lastly, the fission term can be rearranged as

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \, f^*\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right) \left[ \frac{\chi\left(\boldsymbol{x}, E\right)}{4\pi} \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \nu\Sigma_\mathrm{f}\left(\boldsymbol{x}, E'\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right) \right]$$
$$= \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, f^*\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right) \frac{\chi\left(\boldsymbol{x}, E\right)}{4\pi} \nu\Sigma_\mathrm{f}\left(\boldsymbol{x}, E'\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right),$$

where the $\chi\left(\boldsymbol{x}, E\right)$ and $\nu\Sigma_\mathrm{f}\left(\boldsymbol{x}, E'\right)$ terms can be rearranged along with $\boldsymbol{\Omega}, E$ and $\boldsymbol{\Omega}', E'$ to obtain

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, f^*\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right) \frac{\chi\left(\boldsymbol{x}, E\right)}{4\pi} \nu\Sigma_\mathrm{f}\left(\boldsymbol{x}, E'\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right)$$
$$= \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, f^*\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right) \frac{\nu\Sigma_\mathrm{f}\left(\boldsymbol{x}, E\right)}{4\pi} \chi\left(\boldsymbol{x}, E'\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right),$$

regrouped as

$$\int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, f^*\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right) \frac{\nu\Sigma_\mathrm{f}\left(\boldsymbol{x}, E\right)}{4\pi} \chi\left(\boldsymbol{x}, E'\right) f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right)$$
$$= \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V \frac{\nu\Sigma_\mathrm{f}\left(\boldsymbol{x}, E\right)}{4\pi} \left[ \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \, \chi\left(\boldsymbol{x}, E'\right) f^*\left(\boldsymbol{x}, \boldsymbol{\Omega}', E'\right) \right] f\left(\boldsymbol{x}, \boldsymbol{\Omega}, E\right). \quad \text{(A.2d)}$$

Introducing Eq. (A.2) into Eq. (A.1) yields

$$\begin{aligned}\langle f, Mf^* \rangle = \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_V \mathrm{d}V\, f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \Bigg[ &- \boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) \\ &+ \Sigma_\mathrm{t}(\boldsymbol{x}, E) f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) \\ &- \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \Sigma_\mathrm{s}(\boldsymbol{x}, \boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E') f^*(\boldsymbol{x}, \boldsymbol{\Omega}', E') \\ &- \frac{\nu\Sigma_\mathrm{f}(\boldsymbol{x}, E)}{4\pi} \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \chi(\boldsymbol{x}, E') f^*(\boldsymbol{x}, \boldsymbol{\Omega}', E') \Bigg] \\ &+ \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_{\partial V} \mathrm{d}S\, \boldsymbol{\Omega} \cdot \boldsymbol{n} \left[ f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \right]. \end{aligned} \tag{A.3}$$

The adjoint transport migration operator, $M^*$, can be defined using

$$\begin{aligned} M^* f^* = &- \boldsymbol{\Omega} \cdot \boldsymbol{\nabla} f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) + \Sigma_\mathrm{t}(\boldsymbol{x}, E) f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) \\ &- \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \Sigma_\mathrm{s}(\boldsymbol{x}, \boldsymbol{\Omega}, E \to \boldsymbol{\Omega}', E') f^*(\boldsymbol{x}, \boldsymbol{\Omega}', E') \\ &- \frac{\nu\Sigma_\mathrm{f}(\boldsymbol{x}, E)}{4\pi} \int_0^\infty \mathrm{d}E' \int_{4\pi} \mathrm{d}\Omega' \chi(\boldsymbol{x}, E') f^*(\boldsymbol{x}, \boldsymbol{\Omega}', E'), \end{aligned} \tag{A.4}$$

so Eq. (A.3) can be rewritten more compactly as:

$$\langle f^*, Mf \rangle = \langle M^* f^*, f \rangle + \int_0^\infty \mathrm{d}E \int_{4\pi} \mathrm{d}\Omega \int_{\partial V} \mathrm{d}S\, \boldsymbol{\Omega} \cdot \boldsymbol{n} \left[ f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \right], \tag{A.5}$$

which holds for all suitably smooth functions $f(\boldsymbol{x}, \boldsymbol{\Omega}, E)$ and $f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E)$. Furthermore, the surface integral term can be separated rewritten as

$$\begin{aligned} &\int_0^\infty \mathrm{d}E \int_{\partial V} \mathrm{d}S \int_{4\pi} \mathrm{d}\Omega\, \boldsymbol{\Omega} \cdot \boldsymbol{n} \left[ f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \right] \\ &\qquad = \int_0^\infty \mathrm{d}E \int_{\partial V} \mathrm{d}S \int_{\boldsymbol{\Omega} \cdot \boldsymbol{n} > 0} \mathrm{d}\Omega\, \boldsymbol{\Omega} \cdot \boldsymbol{n} \left[ f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \right] \\ &\qquad\qquad - \int_0^\infty \mathrm{d}E \int_{\partial V} \mathrm{d}S \int_{\boldsymbol{\Omega} \cdot \boldsymbol{n} < 0} \mathrm{d}\Omega\, |\boldsymbol{\Omega} \cdot \boldsymbol{n}| \left[ f^*(\boldsymbol{x}, \boldsymbol{\Omega}, E) f(\boldsymbol{x}, \boldsymbol{\Omega}, E) \right] \end{aligned} \tag{A.6}$$

by separating the angular integral into two hemispheres corresponding to outward and inward-directions of

travel relative to the surface. This allows Eq. (A.5) to be written as

$$\langle f^*, Mf\rangle + \int_0^\infty \mathrm{d}E \int_{\partial V} \mathrm{d}S \int_{\mathbf{\Omega}\cdot\boldsymbol{n}<0} \mathrm{d}\Omega\, |\mathbf{\Omega}\cdot\boldsymbol{n}|\, [f^*(\boldsymbol{x},\mathbf{\Omega},E)\, f(\boldsymbol{x},\mathbf{\Omega},E)]$$
$$= \langle M^* f^*, f\rangle + \int_0^\infty \mathrm{d}E \int_{\partial V} \mathrm{d}S \int_{\mathbf{\Omega}\cdot\boldsymbol{n}>0} \mathrm{d}\Omega\, \mathbf{\Omega}\cdot\boldsymbol{n}\, [f^*(\boldsymbol{x},\mathbf{\Omega},E)\, f(\boldsymbol{x},\mathbf{\Omega},E)]\,, \quad \text{(A.7)}$$

which also holds for suitably smooth functions $f(\boldsymbol{x},\mathbf{\Omega},E)$ and $f^*(\boldsymbol{x},\mathbf{\Omega},E)$. Using the forward transport equation and its associated boundary conditions:

$$Mf = Q\,,\ \boldsymbol{x}\in V,\ \mathbf{\Omega}\in 4\pi,\ 0<E<\infty, \quad \text{(A.8a)}$$
$$f = f_\mathrm{b}\,,\ \boldsymbol{x}\in \partial V,\ \mathbf{\Omega}\cdot\boldsymbol{n}<0,\ 0<E<\infty, \quad \text{(A.8b)}$$

and the adjoint transport equation and its associated boundary conditions:

$$M^* f^* = Q^*\,,\ \boldsymbol{x}\in V,\ \mathbf{\Omega}\in 4\pi,\ 0<E<\infty, \quad \text{(A.9a)}$$
$$f^* = f_\mathrm{b}^*\,,\ \boldsymbol{x}\in \partial V,\ \mathbf{\Omega}\cdot\boldsymbol{n}>0,\ 0<E<\infty, \quad \text{(A.9b)}$$

Eq. (A.7) can be written as

$$\langle f^*, Q\rangle + \int_0^\infty \mathrm{d}E \int_{\partial V} \mathrm{d}S \int_{\mathbf{\Omega}\cdot\boldsymbol{n}<0} \mathrm{d}\Omega\, |\mathbf{\Omega}\cdot\boldsymbol{n}|\, [f^*(\boldsymbol{x},\mathbf{\Omega},E)\, f_\mathrm{b}(\boldsymbol{x},\mathbf{\Omega},E)]$$
$$= \langle Q^*, f\rangle + \int_0^\infty \mathrm{d}E \int_{\partial V} \mathrm{d}S \int_{\mathbf{\Omega}\cdot\boldsymbol{n}>0} \mathrm{d}\Omega\, \mathbf{\Omega}\cdot\boldsymbol{n}\, [f_\mathrm{b}^*(\boldsymbol{x},\mathbf{\Omega},E)\, f(\boldsymbol{x},\mathbf{\Omega},E)]\,. \quad \text{(A.10)}$$

## Appendix B

## Arbitrary Convex Polyhedra as DXTRAN Regions

To verify and validate the modifications made to COVRT to simulate the behavior of MCNP6 DXTRAN regions, it is desirable to make as close of a comparison between the behavior of both codes as possible. When this work began, MCNP6 could only make use of spherical DXTRAN regions. Comparing results obtained using Cartesian COVRT DXTRAN regions to results using spherical MCNP6 DXTRAN regions may introduce potentially confounding disagreement. As such, an algorithm to use arbitrary convex polyhedral DXTRAN regions was developed and implemented in a research version of MCNP6 as part of this work. Initially, a strictly Cartesian algorithm was developed and tested; however, extending it to arbitrary convex polyhedra (hereafter polyhedra with arbitrary and convex implied) required only changes to the algorithm's implementation.

The algorithm developed and implemented is described in [126] along with an extensive set of test cases to exercise it. The algorithm's method is extracted from [126] and provided below with some modification to improve its ability to stand alone and to remove supplemental implementation details. For the upcoming discussion, it is assumed that the reader is familiar with DXTRAN processing as described in Section 2.4.6.2.

To begin, assume the particle initiating DXTRAN (either through undergoing source emission or non-absorptive collision) is at position $\boldsymbol{p} = (p_x, p_y, p_z)$. DXTRAN can only be initiated external to the DXTRAN region so $\boldsymbol{p}$ must not be within the DXTRAN region or on its surface. Rather than assuming a circular projection as with spherical DXTRAN, calculate the 2-D projection of the DXTRAN region as viewed by the initiating particle. Find this 2-D projection by projecting the vertices of the polyhedron defining the DXTRAN region into a projection plane as viewed from $\boldsymbol{p}$ and calculating the 2-D convex hull. The convex hull gives the angular extents that a DXTRAN particle can be projected toward and the interior points of the convex hull are all valid points to project toward. To construct the convex hull, begin with a DXTRAN region with the boundary specified by an arbitrary convex polyhedron defined with vertices. Number the vertices 1, 2, ... $N$ in an arbitrary order which then have coordinates

$$\boldsymbol{v}_n = (v_{n,x}, v_{n,y}, v_{n,z}) \ , \ n = 1, 2, ..., N. \tag{B.1}$$

See Fig. B.1 for a representative arrangement of $\boldsymbol{p}$ and a DXTRAN region. Determine the closest point $\boldsymbol{c} = (c_x, c_y, c_z)$ on the surface of the DXTRAN region relative to $\boldsymbol{p}$. This may be either on a vertex, edge, or face of the DXTRAN region. For an arbitrary polyhedron, the RGJK algorithm [127] can accomplish this.

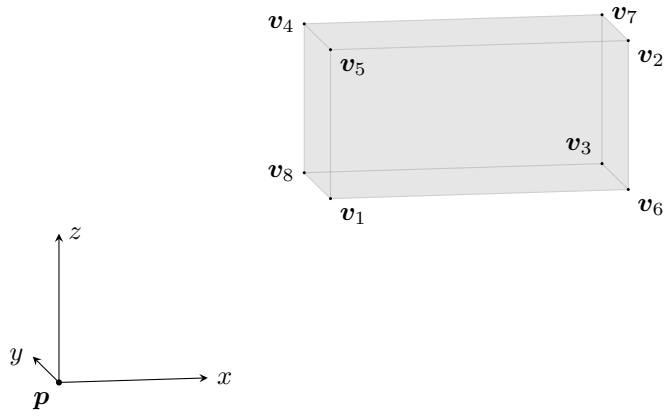



**Figure B.1:** Representative DXTRAN Region and Initiation Point $\boldsymbol{p}$

For the special case of an AARPP, do this more simply by computing

$$c_i = \begin{cases} \min\left(v_{n,i}\right) & , p_i < \min\left(v_{n,i}\right) \\ p_i & , \min\left(v_{n,i}\right) \leq p_i \leq \max\left(v_{n,i}\right) \\ \max\left(v_{n,i}\right) & , p_i > \max\left(v_{n,i}\right) \end{cases} \tag{B.2}$$

for $i = x, y, z$. Next, compute a unit normal vector $\boldsymbol{n}$ that defines the plane containing $\boldsymbol{c}$:

$$\boldsymbol{n} = \frac{\boldsymbol{c} - \boldsymbol{p}}{\|\boldsymbol{c} - \boldsymbol{p}\|} = \left(n_x, n_y, n_z\right), \tag{B.3}$$

where $\boldsymbol{c}$ is on the plane, by definition. Throughout this discussion $\|\cdot\|$ indicates the Euclidean-norm calculation for a vector.

Next, form the equation for the plane containing $\boldsymbol{c}$ with normal $\boldsymbol{n}$:

$$n_x\left(x - c_x\right) + n_y\left(y - c_y\right) + n_z\left(z - c_z\right) = 0. \tag{B.4}$$

The equation for the line passing from $\boldsymbol{p}$ to $\boldsymbol{v}_n$ is

$$L_n\left(s_n\right) = \boldsymbol{p} + \left(\boldsymbol{v}_n - \boldsymbol{p}\right) s_n, \tag{B.5}$$

which can be decomposed into components as

$$x_n = p_x + \left(v_{n,x} - p_x\right) s_n, \tag{B.6a}$$

$$y_n = p_y + \left(v_{n,y} - p_y\right) s_n, \tag{B.6b}$$

$$z_n = p_z + \left(v_{n,z} - p_z\right) s_n. \tag{B.6c}$$

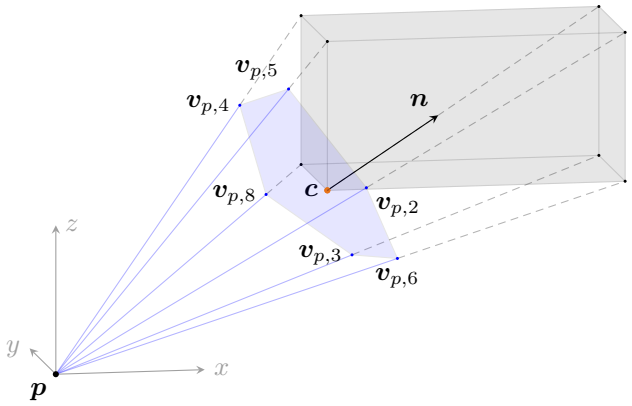



**Figure B.2:** Convex Hull of Projected DXTRAN Vertices

Then introduce Eqs. (B.6a)–(B.6c) into Eq. (B.4) to obtain

$$\begin{aligned} 0 = {} & n_x \left(p_x + \left(v_{n,x} - p_x\right) s_n - c_x\right) \\ & + n_y \left(p_y + \left(v_{n,y} - p_y\right) s_n - c_y\right) \\ & + n_z \left(p_z + \left(v_{n,z} - p_z\right) s_n - c_z\right) \end{aligned} \tag{B.7}$$

and solve for each $s_n$ to find the distance necessary to project the vertices onto the plane. Solving for $s_n$ yields

$$\begin{aligned} s_n &= \frac{\left(n_x c_x + n_y c_y + n_z c_z\right) - \left(n_x p_x + n_y p_y + n_z p_z\right)}{n_x \left(v_{n,x} - p_x\right) + n_y \left(v_{n,y} - p_y\right) + n_z \left(v_{n,z} - p_z\right)} \\ &= \frac{n_x \left(c_x - p_x\right) + n_y \left(c_y - p_y\right) + n_z \left(c_z - p_z\right)}{n_x \left(v_{n,x} - p_x\right) + n_y \left(v_{n,y} - p_y\right) + n_z \left(v_{n,z} - p_z\right)} \\ &= \frac{\boldsymbol{n} \cdot \left(\boldsymbol{c} - \boldsymbol{p}\right)}{\boldsymbol{n} \cdot \left(\boldsymbol{v}_n - \boldsymbol{p}\right)}. \end{aligned} \tag{B.8}$$

There are now $N$ values of $s_n$ that can be used to find the $N$ projected vertices on the plane, $\boldsymbol{v}_{p,n}$, $n = 1, 2, \ldots, N$. The dashed lines in Fig. B.2 show the vertex projections to the plane containing $\boldsymbol{c}$, which will then be used to construct a 2-D convex hull bounding the space that DXTRAN particles can be projected toward. Note that earlier the closest point on the DXTRAN region was used to form the projection plane rather than another easier-to-calculate point such as the DXTRAN region's centroid to ensure that all vertices can be projected to the plane by eliminating the possibility of back projection from a vertex through $\boldsymbol{p}$. Thus, all $s_n$ are expected to be non-negative.

From the view of $\boldsymbol{p}$ toward $\boldsymbol{c}$ a subset of the $\boldsymbol{v}_{p,n}$ points form a convex hull in the plane so some points can be disregarded. Form the convex hull consisting of $H$ points $\boldsymbol{v}_h$, $h = 1, 2, \ldots, H$, $H < N$ with adjacent vertices connected through $H$ edges that form the boundary of the convex hull. For arbitrary polyhedra, the convex hull algorithm by Chan [128] may be best suited to provide optimum speed. However, for AARPPs, which only have eight vertices, the Jarvis marching algorithm [129] is adequate and arguably the easiest to implement.

With the convex hull showing the area that the DXTRAN particle can project into, the new direction of flight for the DXTRAN particle is calculated. In the spherical DXTRAN process, the polar angle is calculated and then the azimuthal angle is calculated. Unlike in the case of spherical DXTRAN, the range of polar cosines depends on the azimuthal angle for polyhedral DXTRAN. Thus, one must determine the new azimuthal angle, calculate the range of polar cosines, sample the polar direction, and finally calculate the new direction of flight. Begin this process by calculating a temporary and arbitrary unit vector in the previously constructed plane (thus normal to $\boldsymbol{n}$),

$$\boldsymbol{f}' = \begin{cases} \frac{\boldsymbol{i}\times\boldsymbol{n}}{\|\boldsymbol{i}\times\boldsymbol{n}\|} & , \boldsymbol{i} \neq \boldsymbol{n} \\ \frac{\boldsymbol{j}\times\boldsymbol{n}}{\|\boldsymbol{j}\times\boldsymbol{n}\|} & , \boldsymbol{i} = \boldsymbol{n} \end{cases}, \tag{B.9}$$

where $\boldsymbol{i}$ and $\boldsymbol{j}$ are unit vectors along the $x$- and $y$-axes, respectively. Rotate $\boldsymbol{f}'$ using Rodrigues's rotation formula [130],

$$\boldsymbol{f} = \boldsymbol{f}'\cos\gamma + (\boldsymbol{n}\times\boldsymbol{f}')\sin\gamma + \cancelto{0}{\boldsymbol{n}\,(\boldsymbol{n}\cdot\boldsymbol{f}')\,(1-\cos\gamma)}, \tag{B.10}$$

where $\gamma$ is sampled uniformly from 0 to $2\pi$ as $\gamma = 2\pi\zeta$ ($\zeta$ is uniformly distributed in $0 \leq \zeta < 1$) and where the last term goes to zero because $\boldsymbol{f}'$ is normal to $\boldsymbol{n}$, by definition. This determines the azimuthal angle of scatter uniformly about the polar axis $\boldsymbol{n}$ (i.e., with rotational invariance).

The point on the edge of the convex hull pointed to by $\boldsymbol{f}$ then determines the minimum polar cosine of scattering, which in turn can define an (arbitrary) polar cosine pdf, $\widetilde{p}\,(\mu)$. One way to determine the edge pointed to by $\boldsymbol{f}$ is to order (moving either clockwise or counterclockwise from an arbitrary vertex on the convex hull) adjacent vertices on the convex hull as $\boldsymbol{v}_1, \boldsymbol{v}_2, ...\boldsymbol{v}_H$. Then test whether $\boldsymbol{f}$ is between $\boldsymbol{v}_h$ and $\boldsymbol{v}_{h+1}$ by iterating through the cyclic set of ordered edges and performing the comparison

$$\left\| \frac{\boldsymbol{f}\times\boldsymbol{h}_1}{\|\boldsymbol{f}\times\boldsymbol{h}_1\|} + \frac{\boldsymbol{f}\times\boldsymbol{h}_2}{\|\boldsymbol{f}\times\boldsymbol{h}_2\|} \right\| - \left\| \frac{\boldsymbol{h}_2\times\boldsymbol{h}_1}{\|\boldsymbol{h}_2\times\boldsymbol{h}_1\|} - \frac{\boldsymbol{h}_2\times\boldsymbol{f}}{\|\boldsymbol{h}_2\times\boldsymbol{f}\|} \right\| \overset{?}{<} -1 \tag{B.11}$$

where

$$\boldsymbol{h}_1 = \boldsymbol{c} - \boldsymbol{v}_h, \tag{B.12a}$$

$$\boldsymbol{h}_2 = \boldsymbol{c} - \boldsymbol{v}_{h+1}. \tag{B.12b}$$

If Inequality B.11 is true, then $\boldsymbol{f}$ is between $\boldsymbol{v}_h$ and $\boldsymbol{v}_{h+1}$.

Having determined which vertices on the convex hull bound the edge pointed to by $\boldsymbol{f}$, the distance $d_h$ from $\boldsymbol{c}$ to the edge is needed to calculate the minimum polar cosine of scattering. See Fig. B.3 for reference. For the upcoming discussion, the following identities are used:

$$\cos\left(\tan^{-1}(x)\right) = \frac{1}{\sqrt{1+x^2}}, \tag{B.13a}$$

$$\tan\left(\cos^{-1}(x)\right) = \frac{\sqrt{1-x^2}}{x}. \tag{B.13b}$$

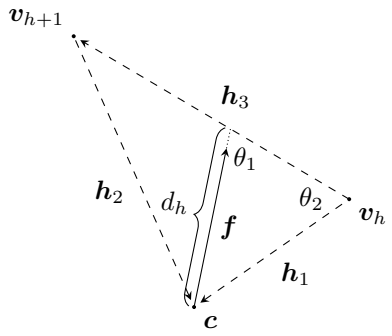



**Figure B.3:** Representative Diagram of Relationship Between $\boldsymbol{c}$ and the Convex Hull Edge Pointed to by $\boldsymbol{f}$

Begin by calculating

$$\boldsymbol{h}_3 = \boldsymbol{v}_{h+1} - \boldsymbol{v}_h, \tag{B.14a}$$

$$\theta_1 = \cos^{-1}\left(\frac{\boldsymbol{h}_3 \cdot \boldsymbol{f}}{\|\boldsymbol{h}_3\|}\right), \tag{B.14b}$$

$$\theta_2 = \cos^{-1}\left(\frac{\boldsymbol{h}_1 \cdot \boldsymbol{h}_3}{\|\boldsymbol{h}_1\| \, \|\boldsymbol{h}_3\|}\right), \tag{B.14c}$$

and by the "Law of Sines":

$$d_h = \|\boldsymbol{h}_1\| \frac{\sin(\theta_2)}{\sin(\theta_1)}. \tag{B.15}$$

The maximum angle of scatter is then

$$\theta_{\max} = \tan^{-1}\left(\frac{\|d_h \boldsymbol{f}\|}{\|\boldsymbol{c} - \boldsymbol{p}\|}\right) = \tan^{-1}\left(\frac{d_h}{\|\boldsymbol{c} - \boldsymbol{p}\|}\right). \tag{B.16}$$

Thus, the minimum polar cosine of scattering is

$$\mu_{\min} = \cos(\theta_{\max}) = \frac{1}{\sqrt{1 + \left(\frac{d_h}{\|\boldsymbol{c}-\boldsymbol{p}\|}\right)^2}} \tag{B.17}$$

with the maximum polar cosine corresponding to the forward direction $\mu = 1$ along $\boldsymbol{n}$ giving the range of polar cosines with azimuthal angle $\gamma$ (shown as the triangular orange shaded region in Fig. B.4). The probability density function (arbitrarily chosen) for selecting $\mu$ is

$$\widetilde{p}(\mu) = \begin{cases} \frac{1}{1-\mu_{\min}}, & \mu_{\min} \leq \mu < 1 \\ 0, & \text{else} \end{cases} \tag{B.18}$$

so $\mu$ can be sampled using $0 \leq \zeta < 1$ as

$$\mu = \zeta(1 - \mu_{\min}) + \mu_{\min}. \tag{B.19}$$

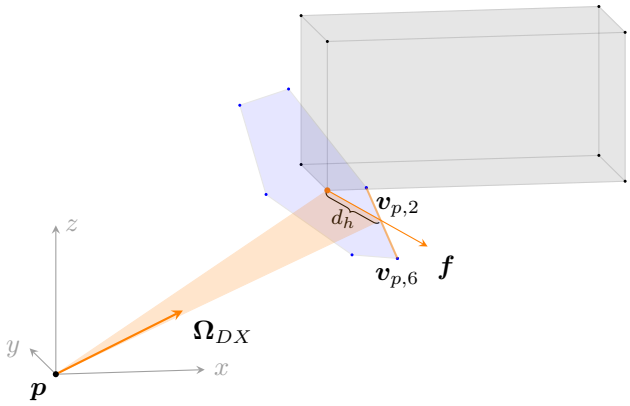



**Figure B.4:** Direction of Flight $\boldsymbol{\Omega}_{\mathrm{DX}}$ Based on Sampled $\gamma$,$\mu$

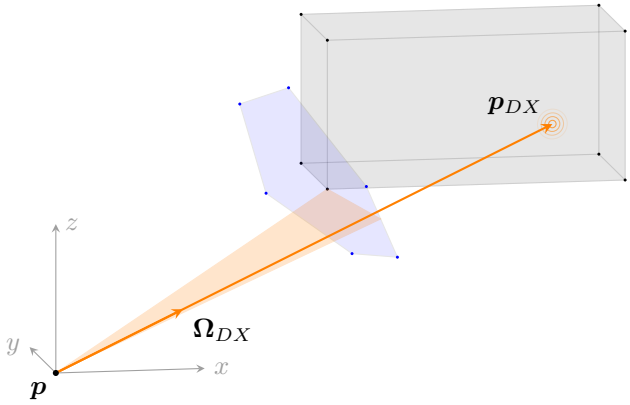



**Figure B.5:** DXTRAN Particle Projected onto DXTRAN Region Boundary

Having selected $\mu$ (and $\gamma$), calculate the direction of projection toward the DXTRAN region as

$$\begin{aligned}\boldsymbol{\Omega}_{\mathrm{DX}} &= \frac{\boldsymbol{c}+\|\boldsymbol{c}-\boldsymbol{p}\|\tan\left(\cos^{-1}(\mu)\right)\boldsymbol{f}-\boldsymbol{p}}{\left\|\boldsymbol{c}+\|\boldsymbol{c}-\boldsymbol{p}\|\tan\left(\cos^{-1}(\mu)\right)\boldsymbol{f}-\boldsymbol{p}\right\|} \\ &= \frac{\boldsymbol{c}+\|\boldsymbol{c}-\boldsymbol{p}\|\left(\frac{\sqrt{1-\mu^2}}{\mu}\right)\boldsymbol{f}-\boldsymbol{p}}{\left\|\boldsymbol{c}+\|\boldsymbol{c}-\boldsymbol{p}\|\left(\frac{\sqrt{1-\mu^2}}{\mu}\right)\boldsymbol{f}-\boldsymbol{p}\right\|}\end{aligned} \tag{B.20}$$

(also shown in Fig. B.4), which determines the direction of flight from $\boldsymbol{p}$ toward the DXTRAN region and the initial direction of flight of the DXTRAN particle.

Finally, project to the nearest face on the DXTRAN region pointed to by $\boldsymbol{\Omega}_{\mathrm{DX}}$ to find the DXTRAN particle's starting point $\boldsymbol{p}_{DX}$ on the DXTRAN region surface. This point on the DXTRAN region surface is shown in Fig. B.5. This projection can be accomplished by initiating a ray at $\boldsymbol{p}$ projected along $\boldsymbol{\Omega}_{\mathrm{DX}}$ and subsequently using the Cyrus-Beck algorithm [131] to determine the point of intersection (i.e., the DXTRAN particle's starting point). For RPPs, axis-aligned bounding box (AABB) collision detection techniques can be used (if the RPP is not axis aligned, a coordinate transformation can be used to so align it). An overview of these types of collision detection techniques is available, for example, in [132].

# Appendix C

## Angular Quadrature Set Generation

This appendix describes the calculations used to generate the angular quadrature sets used for the 1-D and 2-D test cases in this work. The 1-D quadrature set used is the Gauss-Legendre quadrature set, described next. It is followed by a description of the Triangular Gauss-Chebyshev quadrature set, suitable for 2-D and 3-D calculations.

### C.1 Gauss-Legendre Angular Quadrature Set Generation

The Gauss-Legendre quadrature set is defined with $N$ quadrature points (i.e., the polar cosine levels), $\mu_n$, calculated as the $N$ roots of the $N^{\text{th}}$-order Legendre polynomial,

$$P_N(\mu_n) = 0\,,\; n = 1, 2, \ldots, N. \tag{C.1}$$

The associated quadrature weights are calculated directly [133, page 887] as

$$q_n = \frac{2}{\left(1-\mu_n^2\right)\left(P'_N(\mu_n)\right)^2}, \tag{C.2}$$

where the prime indicates the first derivative with respect to $\mu$.

The *Mathematica* inputs to perform these calculations are shown in Fig. C.1. The resulting $S_8$ Gauss-Legendre polar cosine levels and associated quadrature weights are given in Table C.1 and illustrated in Fig. C.2.

```
n = 8;
μ = x /. Sort@N[Solve[LegendreP[n, x] == 0, x]];
q = 2 / ((1 - μ^2) (D[LegendreP[n, x], x] /. {x → μ})^2);
SetPrecision[TableForm[Transpose[Partition[Flatten[{μ, q}], n]]], 15] // Chop;
```

**Figure C.1:** Gauss-Legendre Quadrature Generator *Mathematica* Input

**Table C.1:** $S_8$ Gauss-Legendre Quadrature Set

| $n$ | $\mu_n$ | $q_n$ |
|---|---|---|
| 1 | -0.96028 98564 97536 | 0.10122 85362 90376 |
| 2 | -0.79666 64774 13627 | 0.22238 10344 53374 |
| 3 | -0.52553 24099 16329 | 0.31370 66458 77887 |
| 4 | -0.18343 46424 95650 | 0.36268 37833 78362 |
| 5 | 0.18343 46424 95650 | 0.36268 37833 78362 |
| 6 | 0.52553 24099 16329 | 0.31370 66458 77887 |
| 7 | 0.79666 64774 13627 | 0.22238 10344 53374 |
| 8 | 0.96028 98564 97536 | 0.10122 85362 90376 |

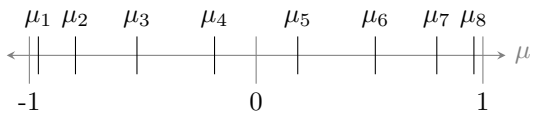



**Figure C.2:** $S_8$ Gauss-Legendre Quadrature Polar Levels

## C.2 Triangular Gauss-Chebyshev Angular Quadrature Set Generation

The Triangular Gauss-Chebyshev quadrature set is suitable for 2-D or 3-D calculations. For the calculations in this work, the polar direction is $\xi$ with polar levels $\xi_n$ also taken as the $N$ roots of the $N^{\text{th}}$-order Legendre polynomial,

$$P_N(\xi_n) = 0\,,\ n = 1, 2, \ldots, N. \tag{C.3}$$

Each polar level has total quadrature weight $q_n$ obtained with Eq. (C.2).

At each polar level $l = 1, 2, \ldots, N/2$ in an octant (with $l = 1$ corresponding to the $\xi_n$ with the greatest magnitude), a given polar level has $l$ azimuthal directions. This results in the quadrature points assuming a triangular shape in an octant. The azimuthal directions at each polar level $l$ are the roots of the $l^{\text{th}}$-order Chebyshev polynomials of the first kind,

$$T_l(\gamma_{k,l}) = 0\,,\ k = 1, 2, \ldots, l. \tag{C.4}$$

Usually, the Chebyshev polynomials are defined in the domain $[-1, 1]$. However, the roots of the Chebyshev polynomials of the first kind, transformed into the domain $[0, \pi/2]$, can be obtained directly [134, page 32] by evaluating

$$\gamma_{k,l} = \frac{\pi}{2}\left(\frac{2l - 2k + 1}{2l}\right). \tag{C.5}$$

The corresponding direction cosines are calculated with

$$\mu_{n,k} = \sqrt{1 - \xi_n^2}\cos(\gamma_{k,l})\,, \tag{C.6a}$$

$$\eta_{n,k} = \sqrt{1 - \xi_n^2}\sin(\gamma_{k,l})\,. \tag{C.6b}$$

Within the polar level, with total quadrature weight for the level, $q_l$, is evenly distributed over the azimuthal angles.

The *Mathematica* inputs to perform these calculations are shown in Fig. C.3. The resulting $S_8$ Triangular Gauss-Chebyshev direction cosines and associated quadrature weights are given in Table C.2 and illustrated in Fig. C.4.

```
n = 8;
ξ = Reverse[x /. Sort@N[Solve[LegendreP[n, x] == 0, x]]];
q = 2 / ((1 - ξ^2) (D[LegendreP[n, x], x] /. {x → ξ})^2);
μ = Table[Table[Sqrt[1 - ξ[[l]]^2] Cos[π/2 * ((2*l - 2*k + 1)/(2*l))], {k, 1, l}], {l, 1, n/2}];
η = Table[Table[Sqrt[1 - ξ[[l]]^2] Sin[π/2 * ((2*l - 2*k + 1)/(2*l))], {k, 1, l}], {l, 1, n/2}];
ξ = Table[Table[ξ[[l]], {k, 1, l}], {l, 1, n/2}];
q = Table[Table[q[[l]]/l, {k, 1, l}], {l, 1, n/2}];
SetPrecision[TableForm[Transpose[Partition[Flatten[{μ, η, ξ, q}], n (2 + n)/8]]], 15] // Chop;
```

**Figure C.3:** Triangular Gauss-Chebyshev Quadrature Generator *Mathematica* Input

**Table C.2:** $S_8$ Triangular Gauss-Chebyshev Quadrature Set in an Octant

| $n$ | $\mu_n$ | $\eta_n$ | $\xi_n$ | $q_n$ |
|---|---|---|---|---|
| 1 | 0.19728 58224 85982 | 0.19728 58224 85982 | 0.96028 98564 97536 | 0.10122 85362 90376 |
| 2 | 0.55841 04931 14176 | 0.23130 11996 19340 | 0.79666 64774 13627 | 0.11119 05172 26687 |
| 3 | 0.23130 11996 19340 | 0.55841 04931 14176 | 0.79666 64774 13627 | 0.11119 05172 26687 |
| 4 | 0.82178 41742 12319 | 0.22019 64058 32868 | 0.52553 24099 16329 | 0.10456 88819 59296 |
| 5 | 0.60158 77683 79451 | 0.60158 77683 79451 | 0.52553 24099 16329 | 0.10456 88819 59296 |
| 6 | 0.22019 64058 32868 | 0.82178 41742 12319 | 0.52553 24099 16329 | 0.10456 88819 59296 |
| 7 | 0.96414 32254 26533 | 0.19178 00114 62651 | 0.18343 46424 95650 | 0.09067 09458 44591 |
| 8 | 0.81736 11593 35446 | 0.54614 32661 32897 | 0.18343 46424 95650 | 0.09067 09458 44591 |
| 9 | 0.54614 32661 32897 | 0.81736 11593 35446 | 0.18343 46424 95650 | 0.09067 09458 44591 |
| 10 | 0.19178 00114 62651 | 0.96414 32254 26533 | 0.18343 46424 95650 | 0.09067 09458 44591 |

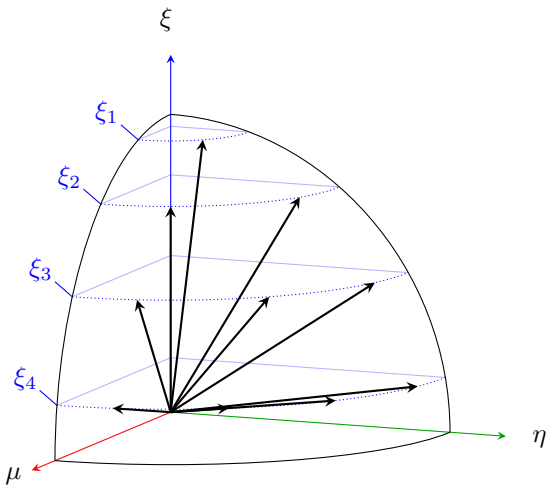



**Figure C.4:** $S_8$ Triangular Gauss-Chebyshev Quadrature Directions Shown in One Octant

## Appendix D

## Demonstration of DXTRAN's Effect on Variance

To demonstrate the effect of DXTRAN on the variance of a solution, consider the trivial case of a mono-energetic, purely scattering ($\Sigma_t = \Sigma_s$), and mono-directional forward transport problem

$$\mu \frac{df}{dx} + \Sigma_t f(x) = \Sigma_s f(x), \; 0 \leq x \leq X, \tag{D.1a}$$

$$f(0) = 1. \tag{D.1b}$$

This problem has the trivial solution $f(x) = 1$. That is, a source particle at $x = 0$ will eventually leak out of the system at $x = X$ regardless of whether it does so via free-flight from 0 to $X$ or has some number of collisions en route.

First, a series of Monte Carlo calculations using the pseudo-adjoint approach described in Chapter 2 are performed. For these calculations, $X = 2$ cm, $\mu = \sqrt{1/3}$, and $\Sigma_t = \Sigma_s = 1$ cm$^{-1}$. The resulting $\Psi_1(x)$ and $\Psi_2(x)$ values are shown in Fig. D.1. Because the problem is analog and considers only leakage, $\Psi_1(x) = \Psi_2(x)$.

However, if a DXTRAN region is introduced in the region $1.5 \leq x \leq 2$, the resulting moment behavior is shown in Fig. D.2. Now, $\Psi_2(x) > \Psi_1(x)$ where $x < x_{DX}$. This is because uncollided particles score with a weight of one; however, particles that have undergone a collision will have a dispersion of weights. The dispersion in scoring particle weights grows by collision as shown in Fig. D.3. In Fig. D.3, the dispersion of weight is observed by the broadening of the PDF. The increasing probability of scoring is observed by recognizing that the portion of the PDF at $s = 0$ diminishes (those non-scoring particles will eventually contribute some score $s$ eventually if allowed to survive and collide enough). Because particles will *eventually* score in this scenario, the dispersion in scoring particle weight introduces variance whereas DXTRAN does not improve scoring, as the number of collisions approaches infinity, because the particle would have scored regardless. This increased variance is what causes $\Psi_2(x) > \Psi_1(x)$ (and indicates that DXTRAN is an inappropriate technique to reduce variance in this scenario unless there are four or fewer collisions permitted).

Because most scenarios under consideration will not lead to guaranteed scoring if a particle is permitted to survive long enough, DXTRAN can (if encompassing a tally region) increase the probability of scoring such that the increased dispersion in weight is offset by the benefit of having scored. This is the scenario that DXTRAN regions are intended to address, where the low-probability of scoring is because of small solid angle subtended by the tally region relative to the remaining phase space thus leading to a low probability of

particles transporting toward and into the tally region.

Other observations from this simple scenario include:

1. For uncollided particles, the particle must score during its first free flight from the source so Fig. D.2a shows scores only for $s = 0, 1$.

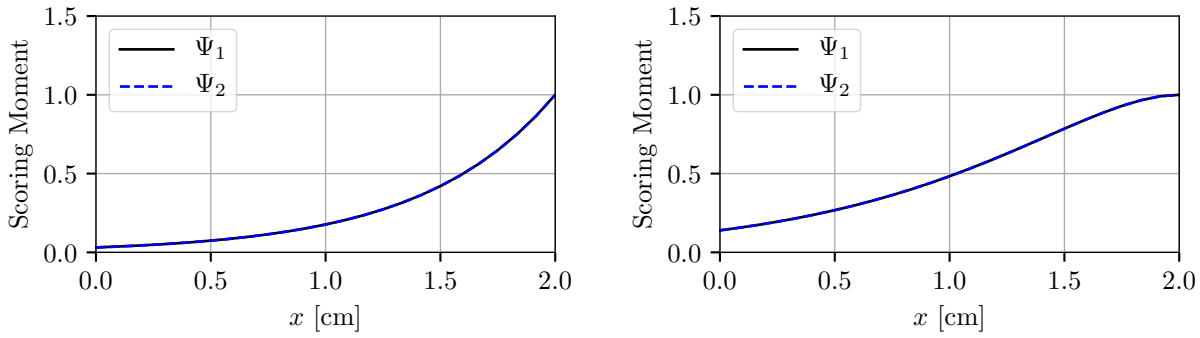
2. For once-collided particles, Fig. D.2b continues to show scores for $s = 0, 1$ but also a continuum for

$$\exp\left(-\frac{\Sigma_{\mathrm{t}}\left(x_{\mathrm{DX}} - x_{\mathrm{coll}}\right)}{\mu}\right) < s < 1,$$

   where the minimum score corresponds to the weight correction performed as a part of the DXTRAN process. No scores are made for $s > 1$ because the particle can only (a) score with $s = 1$ on its free-flight from the source, (b) score with $s = 1$ following its first collision in the region $x_{\mathrm{DX}} \leq x \leq X$, (c) score within the range described previously after having collided in $0 < x < x_{\mathrm{DX}}$ and then undergoing free flight from $x_{\mathrm{DX}}$ to $X$, or (d) failing to score ($s = 0$).

3. For particles with two or more collisions an opportunity exists to have a history score with $s > 1$. This is because each collision outside the DXTRAN region produces a DXTRAN particle, which in turn has an opportunity to score, regardless of whether or not the initiating particle is killed as it tries to enter the DXTRAN region.

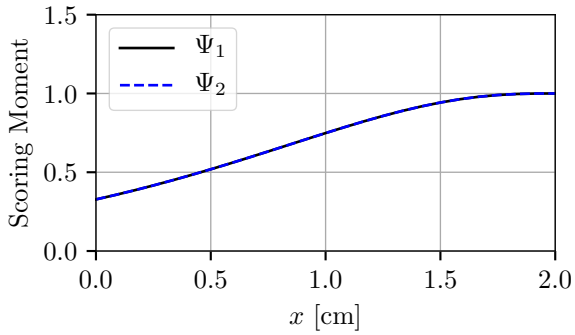
4. The zeroth, first, and second moments ($M_0$, $M_1$, and $M_2$) of the score PDF are given for Fig. D.3. As expected, $M_0 = 1$ always satisfying one property required for a valid PDF. As the number of collisions increases, $M_1 \to 1$ confirming the physical intuition that as particles if particles are allowed to continue colliding they will eventually leak out of the system and score.



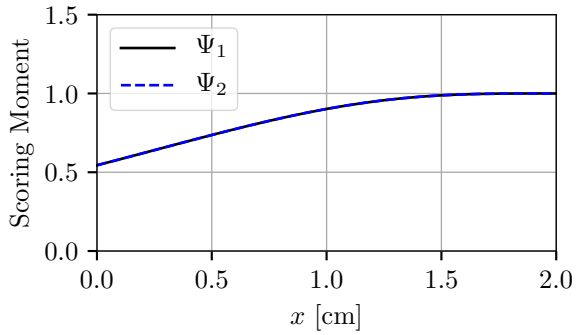

**(a)** 0 Collisions



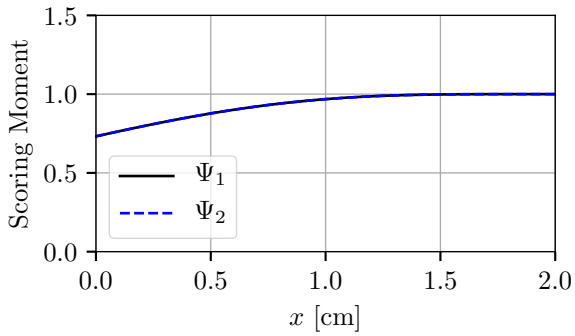

**(b)** 1 Collision



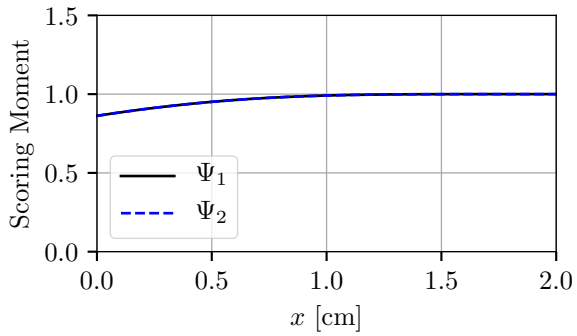

**(c)** 2 Collisions




**(d)** 3 Collisions

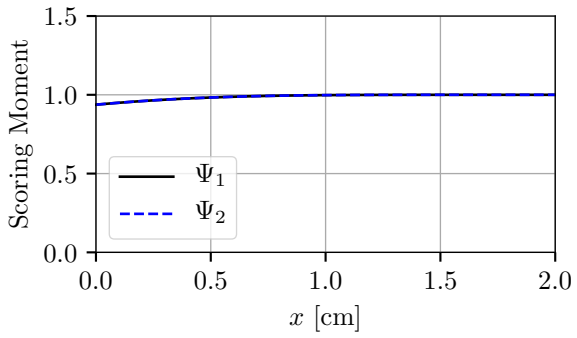



**(e)** 4 Collisions

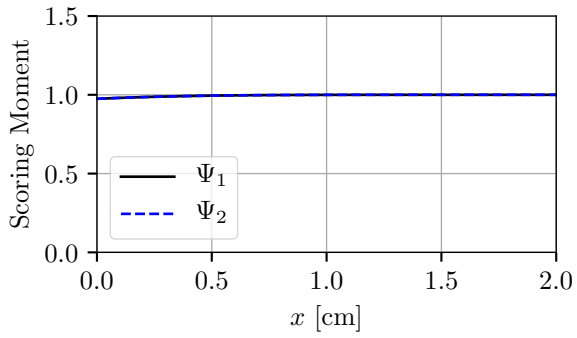



**(f)** 5 Collisions

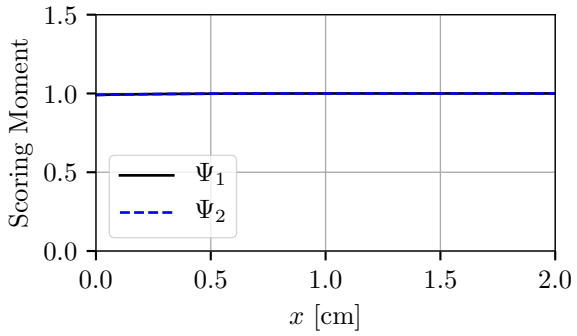
**Figure D.1:** Analog monodirectional purely scattering Monte Carlo scoring moments by collision.



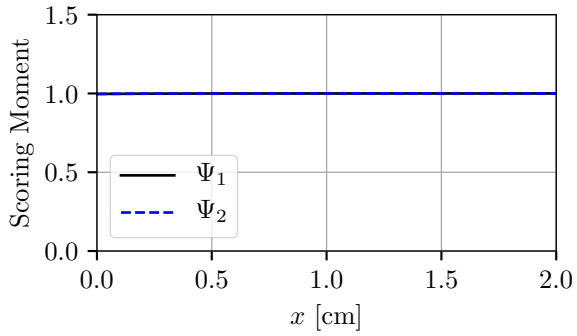

**(g)** 6 Collisions



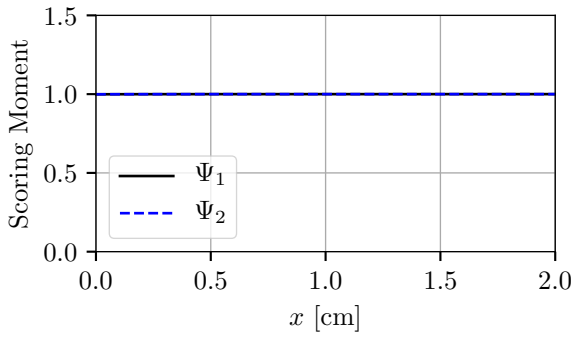

**(h)** 7 Collisions



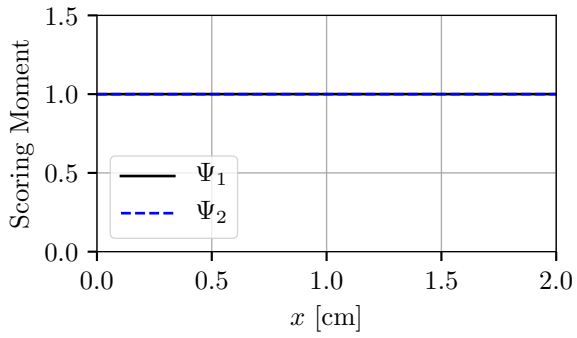

**(i)** 8 Collisions




**(j)** 9 Collisions

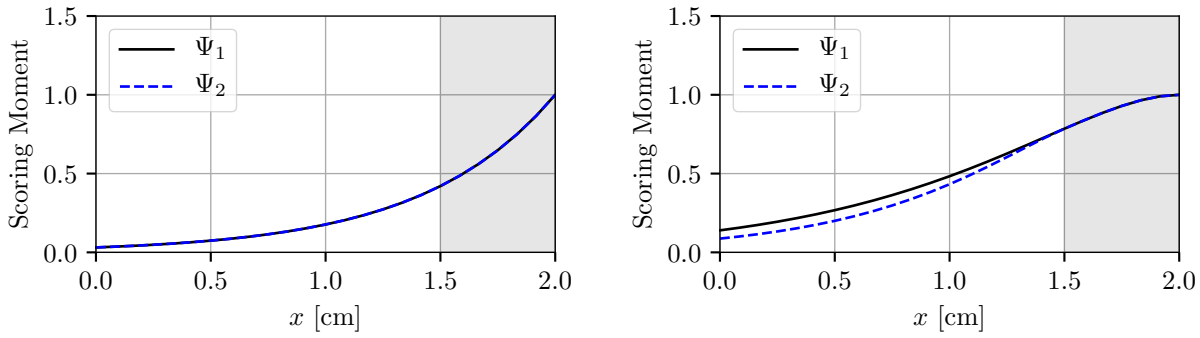



**(k)** 10 Collisions




**(l)** 11 Collisions ($\epsilon \approx 0.002$)

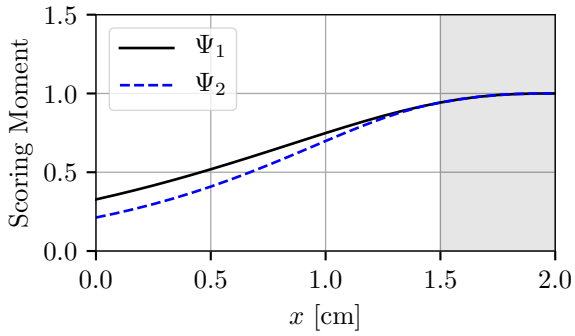
**Figure D.1:** Analog monodirectional purely scattering Monte Carlo scoring moments by collision., continued.



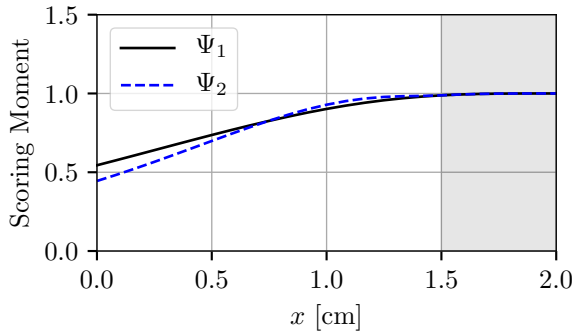

**(a)** 0 Collisions



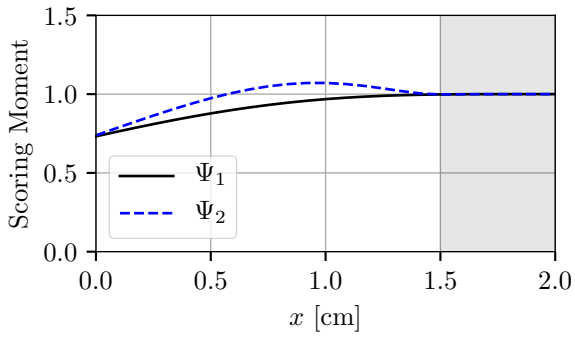

**(b)** 1 Collision



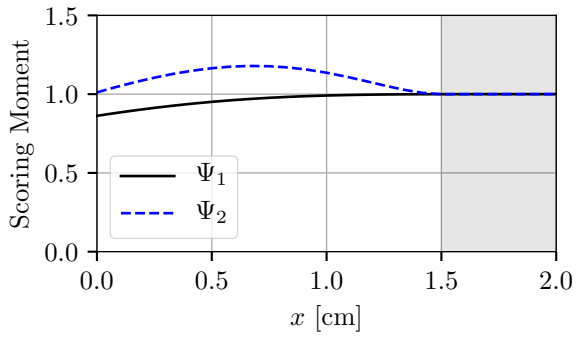

**(c)** 2 Collisions




**(d)** 3 Collisions

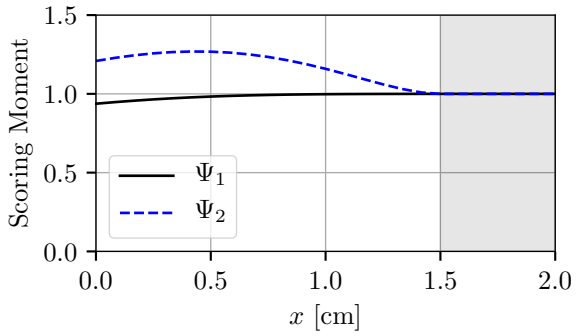



**(e)** 4 Collisions

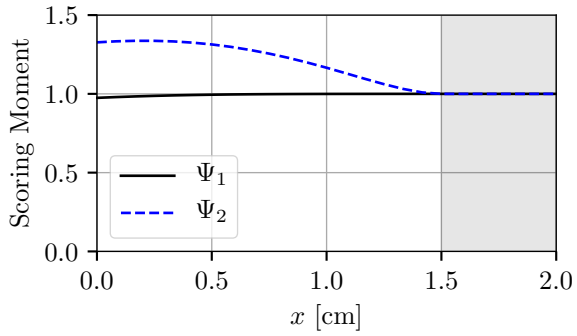



**(f)** 5 Collisions

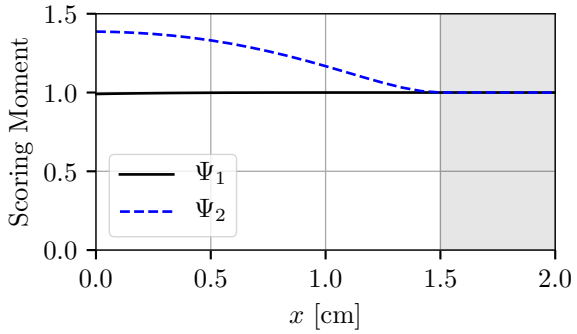
**Figure D.2:** DXTRAN monodirectional purely scattering Monte Carlo scoring moments by collision.



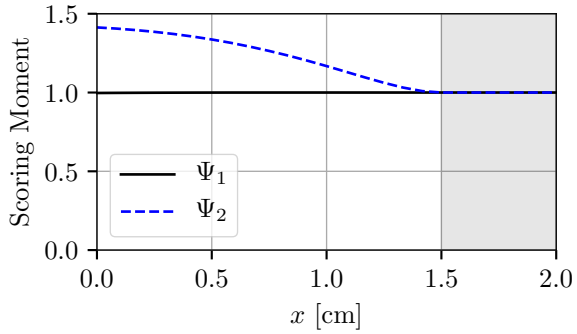

**(g)** 6 Collisions



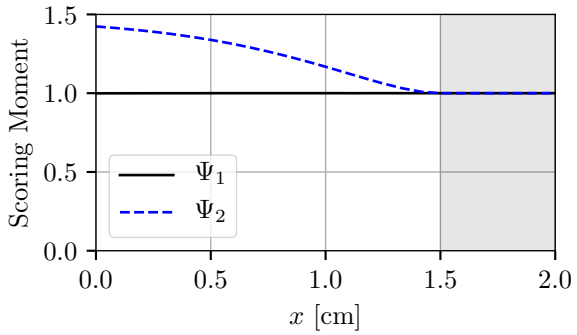

**(h)** 7 Collisions



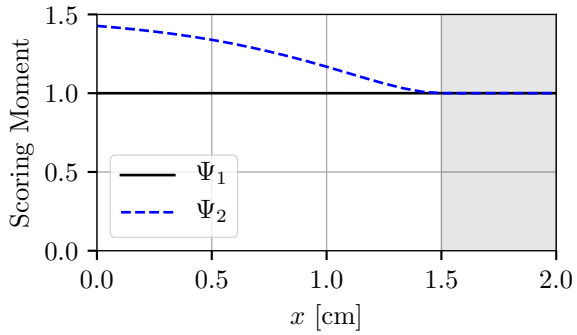

**(i)** 8 Collisions




**(j)** 9 Collisions

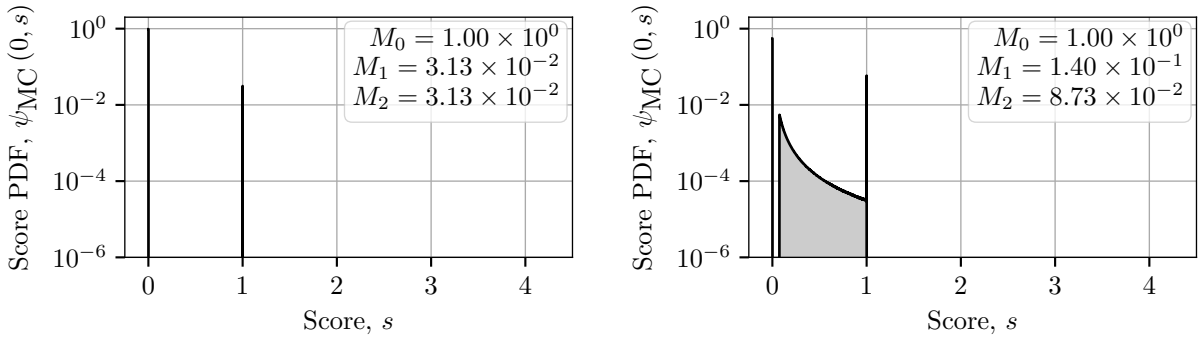



**(k)** 10 Collisions




**(l)** 11 Collisions ($\epsilon \approx 0.002$)

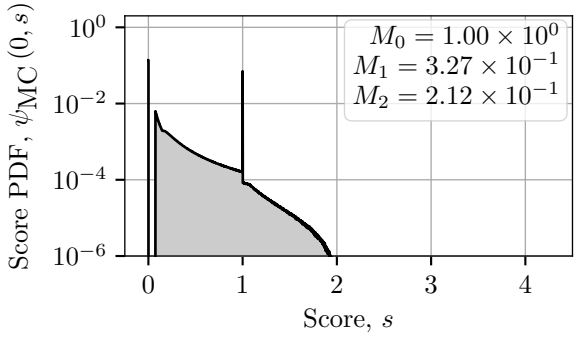
**Figure D.2:** DXTRAN monodirectional purely scattering Monte Carlo scoring moments by collision., continued.



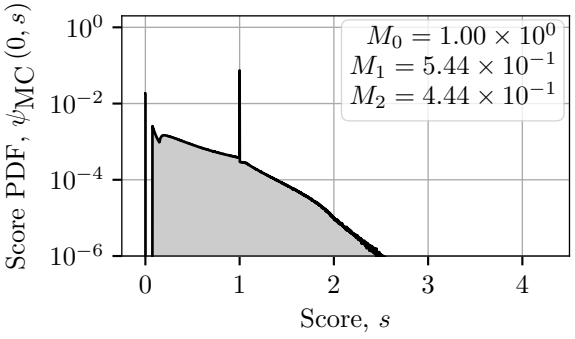

**(a)** 0 Collisions



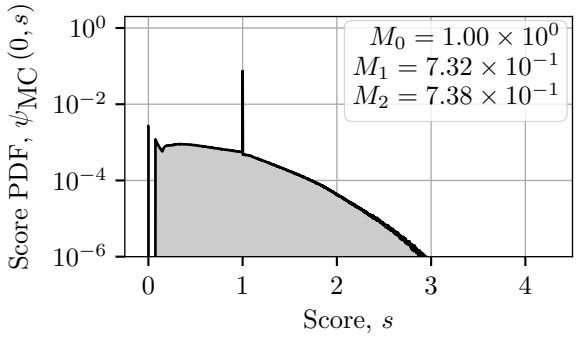

**(b)** 1 Collision



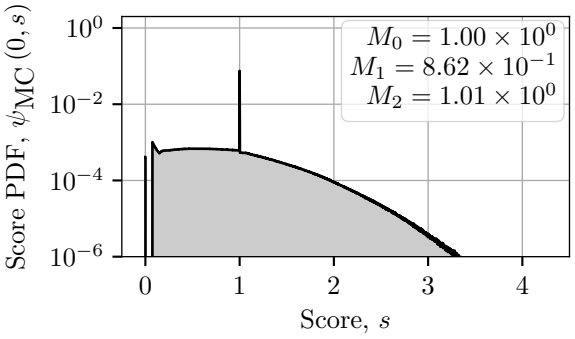

**(c)** 2 Collisions




**(d)** 3 Collisions

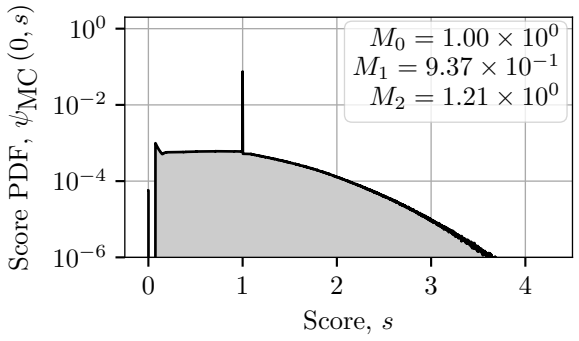



**(e)** 4 Collisions

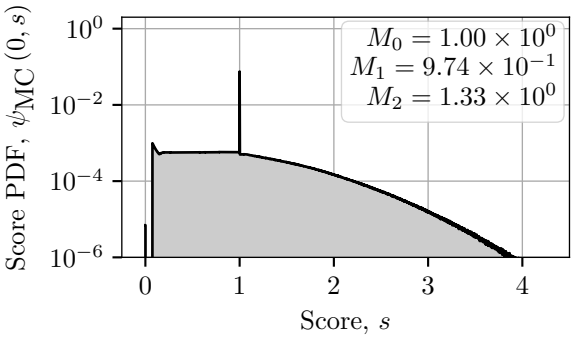



**(f)** 5 Collisions

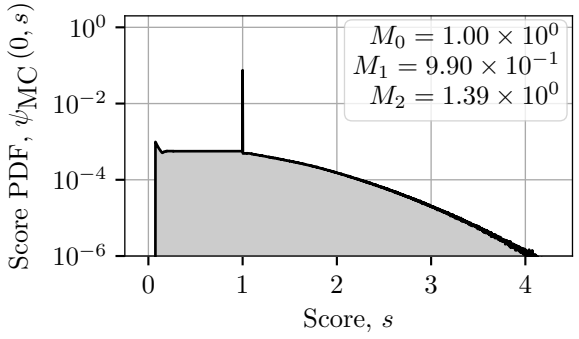
**Figure D.3:** Monodirectional purely scattering DXTRAN Monte Carlo score PDF at $x = 0$.



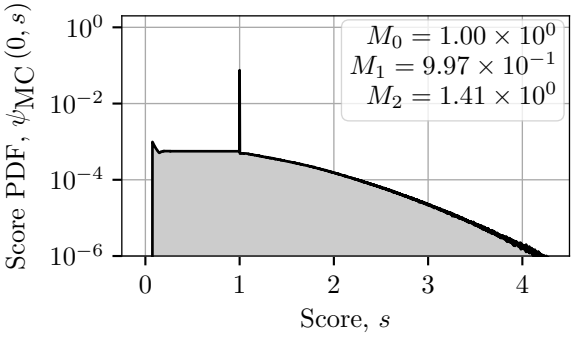

**(g)** 6 Collisions



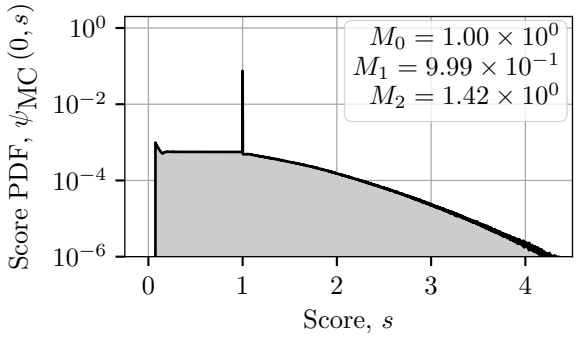

**(h)** 7 Collisions



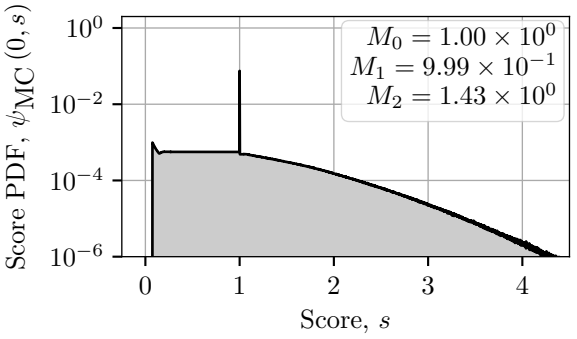

**(i)** 8 Collisions




**(j)** 9 Collisions




**(k)** 10 Collisions




**(l)** 11 Collisions

**Figure D.3:** Monodirectional purely scattering DXTRAN Monte Carlo score PDF at $x = 0.$, continued.

## Appendix E

## Valgrind-based Profiling to Obtain Subroutine Times

As noted in Chapter 4, the times for certain operations in MCNP6 are obtained from profiling using Valgrind's [105] `callgrind` [106] functionality. This appendix describes the profiling process and how the resulting times used in COVRT are obtained. In addition to the profiling process, it is valuable for future readers to be aware of the `qcachegrind` tool. This tool can be used to inspect `callgrind` output interactively (making it *much* easier to understand). An example of `qcachegrind's` interface is shown in Fig. E.1. This information is provided to benefit future readers who may wish to perform similar time profiling.

Two activities are assumed to dominate the time required by DXTRAN. The first source of significant computational expense is the calculation of weight attenuation caused by free-flight from the point of DXTRAN initiation, $\boldsymbol{p}$, to the DXTRAN particle creation site on the DXTRAN surface, $\boldsymbol{p}_{\mathrm{DX}}$. To compute the optical thickness between these points necessitates raytracing through the Monte Carlo geometry between material boundaries (Monte Carlo surfaces). Thus, the time for this activity is assumed to scale linearly according to the number of Monte Carlo surfaces between $\boldsymbol{p}$ and $\boldsymbol{p}_{\mathrm{DX}}$. The second source of significant computational expense is the remainder of DXTRAN processing. The major activities that constitute the remaining DXTRAN processing are banking the initiating particle, determining $\boldsymbol{\Omega}_{\mathrm{DX}}$, determining $\boldsymbol{p}_{\mathrm{DX}}$, calculating $w_{\mathrm{DX}}$, banking the DXTRAN particle, and unbanking the initiating particle. Because the DXTRAN process only generates one DXTRAN particle, there is no need to scale these constituent components of the total DXTRAN time.

To obtain DXTRAN subroutine times, and up-to-date subroutine times for other operations, the input files for Test Case 1-2 (see Section 6.2.2) are reused. Test Case 1-2 consists of 12 input files: one analog calculation and 11 DXTRAN calculations ($x_{\mathrm{DX}} = 0, 1, 2, \ldots, 10$) . Thus, 24 calculations are required to perform profiling: 12 without profiling (to obtain unprofiled total execution times) and 12 with profiling. All calculations are made on a Los Alamos National Laboratory dedicated-use computing node through an allocation on a supercomputer. Each node has two Intel Xeon CPUs (model E5-2695 v4) operating at 2.10 GHz. The profiled calculations give information on the number of instructions performed within all subroutines, the number of times a subroutine is called, and the total number of instructions performed in each of the calculations. By assuming that all instructions are performed in the same amount of unit time, the time to perform a given operation can be computed as

$$\tau_{\mathrm{sub}} = \frac{I_{\mathrm{sub}}}{I_{\mathrm{tot}}} \tau_{\mathrm{calc}}, \tag{E.1}$$

where

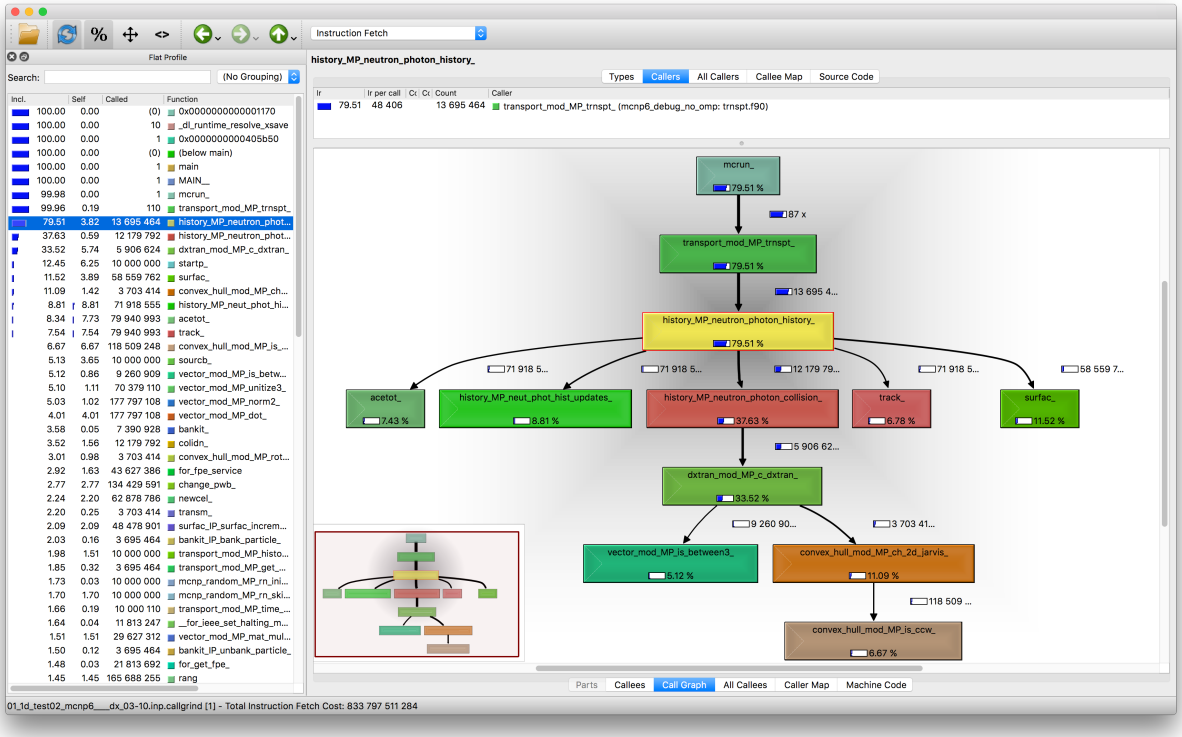



**Figure E.1:** `qcachegrind` View of Profiled Test Case 1-2 Calculation with $x_{\mathrm{DX}} = 3$ cm

$\tau_{\mathrm{sub}}$ is the unprofiled amount of time taken to execute a particular subroutine,

$I_{\mathrm{sub}}$ is the total number of instructions performed within the subroutine,

$I_{\mathrm{tot}}$ is the total number of instructions performed for the calculation, and

$\tau_{\mathrm{calc}}$ is the total amount of computer time for the unprofiled calculation.

The relationships between the COVRT input variables required in Chapter 5 and the corresponding MCNP6 subroutines are given in Table E.1. Also given are the corresponding FTE terms from Chapter 4.

To compute each of the COVRT input variables in Table E.1, the time taken to call the corresponding subroutines within MCNP6 is calculated using Eq. (E.1). Particular care is taken to avoid double counting. That is, if one subroutine calls another the number of instructions of the callee must be subtracted from the caller. As such, the number of `tally` instructions are subtracted from the number of `surfac` instructions (because Test Case 1-2 uses a leakage current tally on the right-hand boundary surface). Also, the number of `transm` instructions are subtracted from the number of `dxtran_mod_MP_c_dxtran` instructions. Otherwise, all subroutine instruction counts are used directly to compute the corresponding subroutine times for each of the 12 cases.

With the subroutine times calculated for each case, the time to perform the ray trace between Monte Carlo surfaces, $\tau_{\mathrm{ff}}$, requires additional processing. For this work, the `transm` times for each case are normalized

**Table E.1:** FTE Parameters, COVRT Input Variables, and Corresponding MCNP Subroutines

| FTE Term | COVRT Input | `callgrind` Subroutine ID |
|---|---|---|
| $\tau_{\text{xs}}$ | `acetot` | `cell_properties_MP_acetot_embed` |
| $\tau_{\text{bank}}$ | `bankit` | `bankit` |
| $\tau_{\text{coll}}$ | `colidn` | `colidn` |
| $\tau_{\text{DX}}$ | `dxtran` | `dxtran_mod_MP_c_dxtran` |
| $\tau_{\text{src}}$ | `startp` | `mcnp_random_mp_rn_init_particle` |
| $\tau_{\text{surf}}$ | `surfac` | `surfac` |
| $\tau_{\text{tally}}$ | `tally` | `tally` |
| $\tau_{\text{geom}}$ | `track` | `track` |
| $\tau_{\text{ff}}$ | `transm` | `transm` |
| $\tau_{\text{ww}}$ | `wtwndo` | `wtwndo` |

by the number of surfaces between $x = 0$ and $x_{\text{DX}}$ (equal to $x_{\text{DX}}$ for $x_{\text{DX}} > 1$ cm). The normalized `transm` times are then averaged over cases (with non-zero times) to compute the average time per Monte Carlo surface between $\boldsymbol{p}$ and $\boldsymbol{p}_{\text{DX}}$. This average time is directly used as input to COVRT.

The individual results for each of the 12 Valgrind profiling executions are given in Tables E.2–E.13. These times are used with the previous discussion to compute the COVRT inputs given in Chapter 7.

**Table E.2:** 20% Absorber Valgrind Profiling Results, Analog, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 58582618381 | 4.11e+00 | 39988357 | 1.03e-07 |
| bankit | 0 | 0.00e+00 | 0 | 0.00e+00 |
| acetot | 42002497494 | 2.94e+00 | 48129921 | 6.12e-08 |
| transm | 0 | 0.00e+00 | 0 | 0.00e+00 |
| track | 37878969093 | 2.66e+00 | 48129921 | 5.52e-08 |
| startp | 103766122730 | 7.28e+00 | 10000000 | 7.28e-07 |
| tally | 5235767944 | 3.67e-01 | 1719464 | 2.13e-07 |
| colidn | 19632511044 | 1.38e+00 | 8141564 | 1.69e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 1.01e+00 | 10000000 | 1.01e-07 |
| dxtran_mod_MP_c_dxtran | 0 | 0.00e+00 | 0 | 0.00e+00 |
| cell_properties_MP_acetot_embed | 0 | 0.00e+00 | 0 | 0.00e+00 |

Obtained from a total of 397491997607 instructions in a 27.87 minute run.

**Table E.3:** 20% Absorber Valgrind Profiling Results, $x_{\mathrm{DX}} = 0$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 58582429498 | 4.55e+00 | 39987304 | 1.14e-07 |
| bankit | 0 | 0.00e+00 | 0 | 0.00e+00 |
| acetot | 41998362169 | 3.26e+00 | 48125722 | 6.78e-08 |
| transm | 0 | 0.00e+00 | 0 | 0.00e+00 |
| track | 37875616436 | 2.94e+00 | 48125722 | 6.11e-08 |
| startp | 103766122730 | 8.06e+00 | 10000000 | 8.06e-07 |
| tally | 5241127144 | 4.07e-01 | 1721224 | 2.36e-07 |
| colidn | 19624833670 | 1.52e+00 | 8138418 | 1.87e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 1.12e+00 | 10000000 | 1.12e-07 |
| dxtran_mod_MP_c_dxtran | 32777171860 | 2.54e+00 | 6509865 | 3.91e-07 |
| cell_properties_MP_acetot_embed | 0 | 0.00e+00 | 0 | 0.00e+00 |

Obtained from a total of 441658842239 instructions in a 34.29 minute run.

**Table E.4:** 20% Absorber Valgrind Profiling Results, $x_{\mathrm{DX}} = 1$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 69608463960 | 4.84e+00 | 47113294 | 1.03e-07 |
| bankit | 12525736315 | 8.70e-01 | 3103118 | 2.81e-07 |
| acetot | 51870575746 | 3.60e+00 | 58962964 | 6.11e-08 |
| transm | 0 | 0.00e+00 | 0 | 0.00e+00 |
| track | 46446939112 | 3.23e+00 | 58962964 | 5.47e-08 |
| startp | 103766122730 | 7.21e+00 | 10000000 | 7.21e-07 |
| tally | 6297749149 | 4.38e-01 | 2058216 | 2.13e-07 |
| colidn | 23292836876 | 1.62e+00 | 9659413 | 1.68e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 1.00e+00 | 10000000 | 1.00e-07 |
| dxtran_mod_MP_c_dxtran | 130529592799 | 9.07e+00 | 5832004 | 1.56e-06 |
| cell_properties_MP_acetot_embed | 0 | 0.00e+00 | 0 | 0.00e+00 |

Obtained from a total of 595148092636 instructions in a 41.36 minute run.

**Table E.5:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 2$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 78989369937 | 5.26e+00 | 53158510 | 9.90e-08 |
| bankit | 21687364405 | 1.44e+00 | 5372814 | 2.69e-07 |
| acetot | 60800797374 | 4.05e+00 | 69381690 | 5.84e-08 |
| transm | 10145110971 | 6.76e-01 | 2690205 | 2.51e-07 |
| track | 54630116310 | 3.64e+00 | 69381690 | 5.24e-08 |
| startp | 103766122730 | 6.91e+00 | 10000000 | 6.91e-07 |
| tally | 7407697735 | 4.93e-01 | 2421306 | 2.04e-07 |
| colidn | 26502026810 | 1.76e+00 | 10990169 | 1.61e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 9.60e-01 | 10000000 | 9.60e-08 |
| dxtran_mod_MP_c_dxtran | 197095158426 | 1.31e+01 | 5746462 | 2.28e-06 |
| cell_properties_MP_acetot_embed | 0 | 0.00e+00 | 0 | 0.00e+00 |

Obtained from a total of 718944674776 instructions in a 47.88 minute run.

**Table E.6:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 3$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 87353373653 | 5.59e+00 | 58559762 | 9.54e-08 |
| bankit | 29833481748 | 1.91e+00 | 7390928 | 2.58e-07 |
| acetot | 69569700257 | 4.45e+00 | 79940993 | 5.57e-08 |
| transm | 18370101431 | 1.18e+00 | 3703414 | 3.17e-07 |
| track | 62894992419 | 4.02e+00 | 79940993 | 5.03e-08 |
| startp | 103766122730 | 6.64e+00 | 10000000 | 6.64e-07 |
| tally | 8718657535 | 5.58e-01 | 2850356 | 1.96e-07 |
| colidn | 29370590911 | 1.88e+00 | 12179792 | 1.54e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 9.23e-01 | 10000000 | 9.23e-08 |
| dxtran_mod_MP_c_dxtran | 261080123457 | 1.67e+01 | 5906624 | 2.83e-06 |
| cell_properties_MP_acetot_embed | 8264243712 | 5.29e-01 | 8022438 | 6.59e-08 |

Obtained from a total of 833797511284 instructions in a 53.35 minute run.

**Table E.7:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 4$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 94191688342 | 5.86e+00 | 63006473 | 9.30e-08 |
| bankit | 37244277916 | 2.32e+00 | 9226874 | 2.51e-07 |
| acetot | 77896140240 | 4.84e+00 | 90180165 | 5.37e-08 |
| transm | 28420212406 | 1.77e+00 | 4626511 | 3.82e-07 |
| track | 70891303245 | 4.41e+00 | 90180165 | 4.89e-08 |
| startp | 103766122730 | 6.45e+00 | 10000000 | 6.45e-07 |
| tally | 10246376287 | 6.37e-01 | 3350552 | 1.90e-07 |
| colidn | 31739606290 | 1.97e+00 | 13162548 | 1.50e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 8.97e-01 | 10000000 | 8.97e-08 |
| dxtran_mod_MP_c_dxtran | 320152133590 | 1.99e+01 | 6204701 | 3.21e-06 |
| cell_properties_MP_acetot_embed | 12223437654 | 7.60e-01 | 12685869 | 5.99e-08 |

Obtained from a total of 939259026852 instructions in a 58.41 minute run.

**Table E.8:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 5$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 98864344899 | 6.00e+00 | 66101976 | 9.08e-08 |
| bankit | 43943422994 | 2.67e+00 | 10886516 | 2.45e-07 |
| acetot | 85337909464 | 5.18e+00 | 99481216 | 5.21e-08 |
| transm | 40007686107 | 2.43e+00 | 5462697 | 4.44e-07 |
| track | 78142575728 | 4.74e+00 | 99481216 | 4.77e-08 |
| startp | 103766122730 | 6.30e+00 | 10000000 | 6.30e-07 |
| tally | 11997912703 | 7.28e-01 | 3924228 | 1.86e-07 |
| colidn | 33370026424 | 2.03e+00 | 13838697 | 1.46e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 8.75e-01 | 10000000 | 8.75e-08 |
| dxtran_mod_MP_c_dxtran | 374093564925 | 2.27e+01 | 6568075 | 3.46e-06 |
| cell_properties_MP_acetot_embed | 16672024290 | 1.01e+00 | 18119427 | 5.58e-08 |

Obtained from a total of 1032190305903 instructions in a 62.64 minute run.

**Table E.9:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 6$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 100566482634 | 6.06e+00 | 67337813 | 9.00e-08 |
| bankit | 49647449730 | 2.99e+00 | 12299628 | 2.43e-07 |
| acetot | 91234585671 | 5.50e+00 | 106991295 | 5.14e-08 |
| transm | 52654793517 | 3.17e+00 | 6176422 | 5.14e-07 |
| track | 83986147245 | 5.06e+00 | 106991295 | 4.73e-08 |
| startp | 103766122730 | 6.26e+00 | 10000000 | 6.26e-07 |
| tally | 13943832568 | 8.41e-01 | 4561742 | 1.84e-07 |
| colidn | 33973789356 | 2.05e+00 | 14089167 | 1.45e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 8.69e-01 | 10000000 | 8.69e-08 |
| dxtran_mod_MP_c_dxtran | 420320849076 | 2.53e+01 | 6923373 | 3.66e-06 |
| cell_properties_MP_acetot_embed | 21426046307 | 1.29e+00 | 24099361 | 5.36e-08 |

Obtained from a total of 1105812283509 instructions in a 66.67 minute run.

**Table E.10:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 7$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 98548602404 | 5.90e+00 | 66238328 | 8.90e-08 |
| bankit | 53977839343 | 3.23e+00 | 13372436 | 2.42e-07 |
| acetot | 94869458032 | 5.68e+00 | 111786115 | 5.08e-08 |
| transm | 65654852771 | 3.93e+00 | 6720482 | 5.85e-07 |
| track | 87703750715 | 5.25e+00 | 111786115 | 4.70e-08 |
| startp | 103766122730 | 6.21e+00 | 10000000 | 6.21e-07 |
| tally | 16018057420 | 9.59e-01 | 5241412 | 1.83e-07 |
| colidn | 33269367826 | 1.99e+00 | 13797386 | 1.44e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 8.63e-01 | 10000000 | 8.63e-08 |
| dxtran_mod_MP_c_dxtran | 455538357747 | 2.73e+01 | 7193000 | 3.79e-06 |
| cell_properties_MP_acetot_embed | 26216129684 | 1.57e+00 | 30293350 | 5.18e-08 |

Obtained from a total of 1152555624891 instructions in a 68.98 minute run.

**Table E.11:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 8$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 92397346612 | 5.56e+00 | 62542724 | 8.88e-08 |
| bankit | 56558228558 | 3.40e+00 | 14011700 | 2.43e-07 |
| acetot | 95675572168 | 5.75e+00 | 113152489 | 5.08e-08 |
| transm | 78254360456 | 4.71e+00 | 7047491 | 6.68e-07 |
| track | 88739745857 | 5.34e+00 | 113152489 | 4.72e-08 |
| startp | 103766122730 | 6.24e+00 | 10000000 | 6.24e-07 |
| tally | 18149765920 | 1.09e+00 | 5940002 | 1.84e-07 |
| colidn | 31038487801 | 1.87e+00 | 12872236 | 1.45e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 8.67e-01 | 10000000 | 8.67e-08 |
| dxtran_mod_MP_c_dxtran | 476445235657 | 2.87e+01 | 7311825 | 3.92e-06 |
| cell_properties_MP_acetot_embed | 30756556755 | 1.85e+00 | 36346326 | 5.09e-08 |

Obtained from a total of 1165983049127 instructions in a 70.12 minute run.

**Table E.12:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 9$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 81598016655 | 4.93e+00 | 55912073 | 8.82e-08 |
| bankit | 56626824891 | 3.42e+00 | 14028694 | 2.44e-07 |
| acetot | 92732670996 | 5.61e+00 | 109980513 | 5.10e-08 |
| transm | 89165462881 | 5.39e+00 | 7062705 | 7.63e-07 |
| track | 86225547569 | 5.21e+00 | 109980513 | 4.74e-08 |
| startp | 103766122730 | 6.27e+00 | 10000000 | 6.27e-07 |
| tally | 20199519220 | 1.22e+00 | 6611729 | 1.85e-07 |
| colidn | 26916484736 | 1.63e+00 | 11162728 | 1.46e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 8.72e-01 | 10000000 | 8.72e-08 |
| dxtran_mod_MP_c_dxtran | 476621791588 | 2.88e+01 | 7172761 | 4.02e-06 |
| cell_properties_MP_acetot_embed | 34551979968 | 2.09e+00 | 41654841 | 5.02e-08 |

Obtained from a total of 1134810147581 instructions in a 68.61 minute run.

**Table E.13:** 20% Absorber Valgrind Profiling Results, $x_{DX} = 10$ cm, Times in Minutes

| MCNP Routine | # Instructions | Time In Routine | Calls | Time Per Call |
|---|---|---|---|---|
| surfac | 64824714659 | 3.86e+00 | 45467520 | 8.49e-08 |
| bankit | 52128121360 | 3.10e+00 | 12914188 | 2.40e-07 |
| acetot | 83635451927 | 4.98e+00 | 99504617 | 5.00e-08 |
| transm | 95590651517 | 5.69e+00 | 6510042 | 8.74e-07 |
| track | 77985444551 | 4.64e+00 | 99504617 | 4.66e-08 |
| startp | 103766122730 | 6.18e+00 | 10000000 | 6.18e-07 |
| tally | 21997155001 | 1.31e+00 | 7200641 | 1.82e-07 |
| colidn | 19626150156 | 1.17e+00 | 8138965 | 1.44e-07 |
| mcnp_random_mp_rn_init_particle | 14418536758 | 8.58e-01 | 10000000 | 8.58e-08 |
| dxtran_mod_MP_c_dxtran | 438749455676 | 2.61e+01 | 6510049 | 4.01e-06 |
| cell_properties_MP_acetot_embed | 36500775959 | 2.17e+00 | 44920585 | 4.84e-08 |

Obtained from a total of 1028929017564 instructions in a 61.24 minute run.

# Appendix F

## Separated- and Combined-moment Solver Examples

This appendix includes concrete implementations of the separated- and combined-moment DXTRAN $\widehat{M}_1$ and $\widehat{M}_2$ solvers to supplement the implementation description in Chapter 5. This appendix is included to provide a clear demonstration of the algorithms described in Chapter 5 and avoids much of the extraneous coding necessary for a more-general tool such as COVRT. As such, the solvers in this appendix have the following limits on their applicability:

1. Only 1-D Cartesian geometry,
2. Only Gauss-Legendre quadrature,
3. Only a leakage current tally on the right-hand boundary,
4. Only an isotropic forward surface source on the left-hand boundary,
5. Only a half-space DXTRAN boundary with its left-hand boundary within the problem geometry and its right-hand boundary coincident with the geometry right-hand boundary.

The separated-moment solver is given in Listing F.1 and the combined-moment solver is given in Listing F.2. Both Listings F.1 and F.2 are complete implementations. That is, either can be extracted into a Python source code file and executed. Because of this (and the necessary significant duplication of code between them), the differences between both solvers are shown in Listing F.3. The differences are obtained with the `diff` command as `diff DXTRAN_Separated_Moments.py DXTRAN_Combined_Moments.py`.

To verify both solvers, Test Case 1-2 [Section 6.2] is executed with 100 spatial mesh ($\Delta_x = 0.1$ cm), $x_{\text{DX}} = 8$ cm, an $S_{32}$ quadrature, and a relative response convergence criterion of $\epsilon = 10^{-4}$. The scattering source iterations for $\widehat{M}_1$ and $\widehat{M}_2$ for the separated- and combined-moment solvers are shown in Listings F.4 and F.5, respectively. The behavior between both solvers is consistent, as expected. Moreover, the values calculated for $\widehat{M}_1$ and $\widehat{M}_2 = \widehat{\sigma}^2 + \widehat{M}_1$ are consistent with Table 6.4. The converged $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$ traverses for direction $n = 32$ (the most-rightward directed) from the separated- and combined-moment solvers are shown in Fig. F.1, which demonstrate the effect of separating the moments where the moments are sent to zero at the DXTRAN boundary.

**Listing F.1:** DXTRAN_Separated_Moments.py

```
#!/usr/bin/env python

def plot_Psi_1_and_2( m, outfilename = None ):
    import matplotlib as mpl
    mpl.use('Agg')
    import matplotlib.pyplot as plt
    plt.figure( figsize = ( 3.25, 3.25/1.62 ) )

    plt.plot( m.x_midpoints,
              m.Psi_1_midpoints[ -1, :, -1 ],
              color = '#000000',
              linestyle = '-',
              label = r'$\Psi_{{1,u,i,n}}$'.format( m.ndirs ) )

    plt.plot( m.x_midpoints,
              m.Psi_2_midpoints[ -1, :, -1 ],
              color = '#0000ff',
              linestyle = '--',
              label = r'$\Psi_{{2,u,i,n}}$'.format( m.ndirs ) )

    plt.axvspan( x_DX, xmax, facecolor = '#00274c', alpha = 0.1, zorder = -10 )

    plt.grid()
    plt.xlabel( r'$x$ [cm]' )
    plt.xlim( ( xmin, xmax ) )
    plt.legend( loc = 'best' )
    plt.tight_layout()
    if( outfilename != None ):
        plt.savefig( outfilename )
    return

class quadrature_Gauss_Legendre( ):
    def __init__( self, ndirs = 2 ):
        from numpy.polynomial import legendre as L
        self.ndirs = ndirs
        # Legendre polynomial coefficients.
        lc = [ 0 for n in range( ndirs ) ] + [ 1 ]
        self.mu = L.legroots( lc )
        # Evaluated polynomial derivatives.
        Pnp = L.legval( self.mu, L.legder( lc ) )
        self.weight = 1.0 / ( (1 - self.mu**2 ) * ( Pnp )**2 )
        return

    def print_quadrature( self ):
        label = 'Quadrature Set'
        print( '\n{:-^80}\n'.format( ' ' + label + ' ' ) )
        print('{:>15s}{:>30s}{:>30s}'.format(
            'n', 'mu', 'weight' ) )
        for n in range( self.ndirs ):
            print('{:15d}{:30.15f}{:30.15f}'.format(
                n, self.mu[n], self.weight[n] ) )
        print( '\n{:-^80}\n'.format( ' End ' + label + ' ' ) )
        return

def calc_pff( mesh, quadrature ):
    # For each mesh outside the DXTRAN region, perform an optical distance
    # integration for it.

    for n in range( quadrature.ndirs ):
        # Skip those directions that cannot hit the DXTRAN region.
        if( n < quadrature.ndirs / 2 ): continue

        for i in range( mesh.idx - 1, -1, -1 ):
```

```
            dist_mesh = np.ones( mesh.idx - i )
            opt_dist_edge = np.sum(
                mesh.Sigma_t[i:mesh.idx] * dist_mesh * mesh.dx[i:mesh.idx] )
            mesh.pff_edges[ i, n ] = np.exp( -opt_dist_edge / quadrature.mu[n] )

            dist_mesh[0] = 0.5
            opt_dist_mid = np.sum(
                mesh.Sigma_t[i:mesh.idx] * dist_mesh * mesh.dx[i:mesh.idx] )
            mesh.pff_midpoints[ i, n ] = np.exp( -opt_dist_mid / quadrature.mu[n] )

    return m

def sweep_Psi_1_DXTRAN( mesh, quadrature, eps = 1e-4, nitermax = 100 ):
    M_1_old = 1
    M_1_new = 0
    eps_rel = 1

    # Setup aliases.
    m = mesh
    q = quadrature
    N = q.ndirs

    u = 0
    for u in range( nitermax-1 ):
        u = u + 1

        # Update scattering source for current iteration based on prior
        # iteration.
        for i in range( m.nmesh ):
            for n in range( N ):
                m.Q_1s_midpoints[ u - 1, i, n ] = \
                    m.Sigma_s[i] * m.Phi_1_midpoints[ u - 1, i ]

        for n in range( N ):
            di = ( 1 if n < N / 2 else -1 )
            iwalk = ( range( 0, m.nmesh , di )
                      if n < N / 2
                      else range( m.nmesh-1, -1, di ) )

            # Perform sweep and accumulate angular moments.
            for i in iwalk:
                # Alias diamond difference factor.
                dd = 2.0 * np.abs( q.mu[ n ] ) / ( m.dx[ i ] )
                if( n < N / 2 ):
                    m.Psi_1_midpoints[ u, i, n ] = \
                        ( dd * m.Psi_1_edges[ u, i, n ] \
                             + m.Q_1s_midpoints[ u - 1, i, n ] ) \
                        / ( dd + m.Sigma_t[ i ] )

                    m.Psi_1_edges[ u, i + di, n ] = \
                        2.0 * m.Psi_1_midpoints[ u, i, n ] \
                            - m.Psi_1_edges[ u, i, n ]
                else:
                    Psi_1_in = ( 0 if i == m.idx - 1 else m.Psi_1_edges[ u, i - di, n ] )
                    m.Psi_1_midpoints[ u, i, n ] = \
                        (dd * Psi_1_in \
                            + m.Q_1s_midpoints[ u - 1, i, n ] ) \
                        / ( dd + m.Sigma_t[ i ] )

                    m.Psi_1_edges[ u, i, n ] = \
                        2.0 * m.Psi_1_midpoints[ u, i, n ] \
                            - Psi_1_in

                    if( i < m.idx ):
                        m.Psi_1_midpoints_DX[ u, i, n ] = \
```

```
                            m.pff_midpoints[ i, n ] * m.Psi_1_edges[ u, m.idx, n ]
                        m.Psi_1_edges_DX[ u, i, n ] = \
                            m.pff_edges[ i, n ] * m.Psi_1_edges[ u, m.idx, n ]

                # Accumulate angular moments of statistical moment.
                m.Phi_1_midpoints[ u, i ] = m.Phi_1_midpoints[ u, i ] \
                    + q.weight[ n ] * m.Psi_1_midpoints[ u, i, n ] \
                    + q.weight[ n ] * m.Psi_1_midpoints_DX[ u, i, n ]

                # Accumulate select angular moments for DXTRAN Q_2.
                m.Phi_1a_midpoints[ u, i ] = m.Phi_1a_midpoints[ u, i ] \
                    + q.weight[ n ] * m.Psi_1_midpoints[ u, i, n ]

                m.Phi_1b_midpoints[ u, i ] = m.Phi_1b_midpoints[ u, i ] \
                    + q.weight[ n ] \
                    * ( m.pff_midpoints[ i, n ] * m.Psi_1_edges[ u, m.idx, n ] )

        # Determine convergence.
        M_1_new = m.calc_M_m( 1 )
        eps_rel = np.abs( M_1_new - M_1_old ) / M_1_old
        print( '  M_1 = {:.5e} ( iter: {:3d}, eps: {:.3e} )'.format(
            M_1_new, u, eps_rel ) )
        if( eps_rel < eps ):
            print( '  M_1 Converged.' )
            break
        else:
            m.grow_Psi_1()
            M_1_old = M_1_new

    return m

def calc_Q_2_DXTRAN( mesh, quadrature ):
    for n in range( quadrature.ndirs ):
        mesh.Q_2_midpoints[ -1, :, n ] = 2.0 \
            * mesh.Sigma_s * mesh.Phi_1a_midpoints[-1,:] \
            * mesh.Sigma_s * mesh.Phi_1b_midpoints[-1,:] \
            / mesh.Sigma_s
    return mesh

def sweep_Psi_2_DXTRAN( mesh, quadrature, eps = 1e-4, nitermax = 100 ):
    M_2_old = 1
    M_2_new = 0
    eps_rel = 1

    # Setup aliases.
    m = mesh
    q = quadrature
    N = q.ndirs

    u = 0
    for u in range( nitermax-1 ):
        u = u + 1

        # Update scattering source for current iteration based on prior
        # iteration.
        for i in range( m.nmesh ):
            for n in range( N ):
                m.Q_2s_midpoints[ u - 1, i, n ] = \
                    m.Sigma_s[i] * m.Phi_2_midpoints[ u - 1, i ]

        for n in range( N ):
            di = ( 1 if n < N / 2 else -1 )
            iwalk = ( range( 0, m.nmesh , di )
                      if n < N / 2
                      else range( m.nmesh-1, -1, di ) )
```

```
194
195             # Perform sweep and accumulate angular moments.
196             for i in iwalk:
197                 # Alias diamond difference factor.
198                 dd = 2.0 * np.abs( q.mu[ n ] ) / ( m.dx[ i ] )
199                 if( n < N / 2 ):
200                     m.Psi_2_midpoints[ u, i, n ] = \
201                         ( dd * m.Psi_2_edges[ u, i, n ] \
202                              + m.Q_2s_midpoints[ u - 1, i, n ]
203                              + m.Q_2_midpoints[ -1, i, n ] ) \
204                         / ( dd + m.Sigma_t[ i ] )
205
206                     m.Psi_2_edges[ u, i + di, n ] = \
207                         2.0 * m.Psi_2_midpoints[ u, i, n ] \
208                             - m.Psi_2_edges[ u, i, n ]
209                 else:
210                     Psi_2_in = ( 0 if i == m.idx - 1 else m.Psi_2_edges[ u, i - di, n ] )
211                     m.Psi_2_midpoints[ u, i, n ] = \
212                         (dd * Psi_2_in \
213                             + m.Q_2s_midpoints[ u - 1, i, n ] \
214                             + m.Q_2_midpoints[ -1, i, n ] ) \
215                         / ( dd + m.Sigma_t[ i ] )
216
217                     m.Psi_2_edges[ u, i, n ] = \
218                         2.0 * m.Psi_2_midpoints[ u, i, n ] \
219                             - Psi_2_in
220
221                     if( i < m.idx ):
222                         m.Psi_2_midpoints_DX[ u, i, n ] = \
223                             m.pff_midpoints[ i, n ] * m.Psi_2_edges[ u, m.idx, n ]
224                         m.Psi_2_edges_DX[ u, i, n ] = \
225                             m.pff_edges[ i, n ] * m.Psi_2_edges[ u, m.idx, n ]
226
227                 # Accumulate angular moments of statistical moment.
228                 p = 0.5
229                 parb = 1.0 * 0.5 # Factor of 0.5 for quadrature weight correction
230                 beta = 1.0
231                 m.Phi_2_midpoints[ u, i ] = m.Phi_2_midpoints[ u, i ] \
232                     + q.weight[ n ] * m.Psi_2_midpoints[ u, i, n ] \
233                     + q.weight[ n ] \
234                     * ( m.pff_midpoints[ i, n ]**2 / beta * p**2 / parb * m.Psi_2_edges[ u, m.idx, n ] )
235
236         # Compute M_2.
237         # Integrate over cell edges.
238         M_2_new = m.calc_M_m( 2 )
239         eps_rel = np.abs( M_2_new - M_2_old ) / M_2_old
240         print( '  M_2 = {:.5e} ( iter: {:3d}, eps: {:.3e} )'.format(
241             M_2_new, u, eps_rel ) )
242         if( eps_rel < eps ):
243             print( '  M_2 Converged.' )
244             break
245         else:
246             m.grow_Psi_2()
247             M_2_old = M_2_new
248
249     return m
250
251 class mesh( ):
252     def __init__( self,
253         quadrature = quadrature_Gauss_Legendre( 2 ),
254         nmesh = 100,
255         nitermax = 10,
256         xmin = 0,
257         x_DX = 5,
258         xmax = 10,
```

```
259         Sigma_t = None,
260         c = None,
261         Qt_midpoints = None,
262         Qt_edges = None,
263         Q_midpoints = None,
264         Q_edges = None  ):
265 
266         import numpy as np
267 
268         # Alias quadrature.
269         q = quadrature
270 
271         # Setup spatial mesh.
272         self.nmesh = nmesh
273         self.ndirs = q.ndirs
274         self.x_edges = np.linspace( xmin, xmax, self.nmesh + 1 )
275         self.x_midpoints = self.x_edges[:-1] + 0.5 * np.diff( self.x_edges )
276         self.dx = np.diff( self.x_edges )
277 
278         # Assign materials to mesh cells.
279         if( Sigma_t == None ):
280             self.Sigma_t = np.ones( nmesh )
281         else:
282             self.Sigma_t = Sigma_t( self.x_midpoints )
283         if( c == None ):
284             self.Sigma_s = np.zeros( nmesh )
285         else:
286             self.Sigma_s = c( self.x_midpoints ) * self.Sigma_t
287 
288         # Assign cell-center and cell-edge tally sources.
289         if( Qt_midpoints == None ):
290             self.Qt_midpoints = np.zeros( ( nmesh, q.ndirs ) )
291         else:
292             self.Qt_midpoints = Qt_midpoints( self.x_midpoints, q )
293         if( Qt_edges == None ):
294             self.Qt_edges = np.zeros( ( nmesh, q.ndirs ) )
295         else:
296             self.Qt_edges = Qt_edges( self.x_edges, q )
297 
298         # Assign cell-center and cell-edge forward sources.
299         if( Q_midpoints == None ):
300             self.Q_midpoints = np.zeros( ( nmesh, q.ndirs ) )
301         else:
302             self.Q_midpoints = Q_midpoints( self.x_midpoints, q )
303         if( Q_edges == None ):
304             self.Q_edges = np.zeros( ( nmesh, q.ndirs ) )
305         else:
306             self.Q_edges = Q_edges( self.x_edges, q )
307 
308         # Initialize angular moment storage arrays, Psi_1, Phi_1, Q_1s.
309         self.Psi_1_midpoints = np.zeros(
310             ( 2, nmesh,  ndirs ) ) # u, i, n
311         self.Psi_1_edges = np.zeros(
312             ( 2, nmesh + 1, ndirs) ) # u, i, n
313         self.Phi_1_midpoints = np.zeros(
314             ( 2, nmesh ) ) # u, i, r = 1
315         self.Phi_1a_midpoints = np.zeros(
316             ( 2, nmesh ) ) # u, i, r = 1
317         self.Phi_1b_midpoints = np.zeros(
318             ( 2, nmesh ) ) # u, i, r = 1
319         self.Q_1s_midpoints = np.zeros(
320             ( 2, nmesh, ndirs ) ) # u, i, n
321 
322         # Initialize DXTRAN Q_2.
323         self.Q_2_midpoints = np.zeros(
```

```
            ( 1, nmesh, ndirs ) ) # u, i, n

        # Initialize angular moment storage arrays, Psi_2, Phi_2, Q_2s.
        self.Psi_2_midpoints = np.zeros(
            ( 2, nmesh, ndirs ) ) # u, i, n
        self.Psi_2_edges = np.zeros(
            ( 2, nmesh + 1, ndirs ) ) # u, i, n
        self.Phi_2_midpoints = np.zeros(
            ( 2, nmesh ) ) # u, i, r = 1
        self.Q_2s_midpoints = np.zeros(
            ( 2, nmesh, ndirs ) ) # u, i, n

        # Place boundary sources onto mesh.
        #         psi_edges[ 0, :, : ] = src_edges[ 0, : ]
        for u in range( 2 ):
            self.Psi_1_edges[ u, :, : ] = self.Qt_edges[ :, : ]
            self.Psi_2_edges[ u, :, : ] = self.Qt_edges[ :, : ]**2

        # DXTRAN cell-edge index for DXTRAN surface.
        self.idx = np.where( self.x_edges == x_DX )[0][0]

        # DXTRAN free-flight probabilities.
        self.pff_midpoints = np.zeros( ( nmesh, ndirs ) )
        self.pff_edges = np.zeros( ( nmesh + 1, ndirs ) )

        # DXTRAN analytic components.
        self.Psi_1_midpoints_DX = np.zeros(
            ( 2, nmesh,  ndirs ) ) # u, i, n
        self.Psi_1_edges_DX= np.zeros(
            ( 2, nmesh + 1, ndirs) ) # u, i, n
        self.Psi_2_midpoints_DX = np.zeros(
            ( 2, nmesh,  ndirs ) ) # u, i, n
        self.Psi_2_edges_DX= np.zeros(
            ( 2, nmesh + 1, ndirs) ) # u, i, n

        return

    def grow_Psi_1( self ):
        # Grow arrays for next iteration.
        self.Psi_1_midpoints = np.vstack(
            ( self.Psi_1_midpoints, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )
        self.Psi_1_edges = np.vstack(
            ( self.Psi_1_edges, np.zeros(
                ( 1, self.nmesh + 1, self.ndirs ) ) ) )
        self.Phi_1_midpoints = np.vstack(
            ( self.Phi_1_midpoints, np.zeros(
                ( 1, self.nmesh ) ) ) )
        self.Phi_1a_midpoints = np.vstack(
            ( self.Phi_1a_midpoints, np.zeros(
                ( 1, self.nmesh ) ) ) )
        self.Phi_1b_midpoints = np.vstack(
            ( self.Phi_1b_midpoints, np.zeros(
                ( 1, self.nmesh ) ) ) )
        self.Q_1s_midpoints = np.vstack(
            ( self.Q_1s_midpoints, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )
        self.Psi_1_midpoints_DX = np.vstack(
            ( self.Psi_1_midpoints_DX, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )
        self.Psi_1_edges_DX = np.vstack(
            ( self.Psi_1_edges_DX, np.zeros(
                ( 1, self.nmesh + 1, self.ndirs ) ) ) )

        # Emplace tally source(s).
```

```
        self.Psi_1_edges[ -1, :, : ] = self.Qt_edges[ :, : ]
        return

    def grow_Psi_2( self ):
        # Grow arrays for next iteration.
        self.Psi_2_midpoints = np.vstack(
            ( self.Psi_2_midpoints, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )
        self.Psi_2_edges = np.vstack(
            ( self.Psi_2_edges, np.zeros(
                ( 1, self.nmesh + 1, self.ndirs ) ) ) )
        self.Phi_2_midpoints = np.vstack(
            ( self.Phi_2_midpoints, np.zeros(
                ( 1, self.nmesh ) ) ) )
        self.Q_2s_midpoints = np.vstack(
            ( self.Q_2s_midpoints, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )
        self.Psi_2_midpoints_DX = np.vstack(
            ( self.Psi_2_midpoints_DX, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )
        self.Psi_2_edges_DX = np.vstack(
            ( self.Psi_2_edges_DX, np.zeros(
                ( 1, self.nmesh + 1, self.ndirs ) ) ) )

        # Emplace tally source(s).
        self.Psi_2_edges[ -1, :, : ] = self.Qt_edges[ :, : ]**2
        return

    def calc_M_m( self, m = 1 ):
        if( m == 1 ):
            Psi_m = self.Psi_1_edges[ -1, :, : ] + self.Psi_1_edges_DX[ -1, :, : ]
        else:
            Psi_m = self.Psi_2_edges[ -1, :, : ] + self.Psi_2_edges_DX[ -1, :, : ]

        # Integrate over cell edges.
        M_m = 0
        for i in range( self.nmesh + 1 ):
            for n in range( q.ndirs ):
                M_m = M_m + \
                    self.Q_edges[ i, n ] \
                    * q.weight[ n ] * Psi_m[ i, n ]

        # Integrate over cell centers.
        for i in range( self.nmesh ):
            for n in range( q.ndirs ):
                M_m = M_m + \
                    self.Q_midpoints[ i, n ] \
                    * q.weight[ n ] * Psi_m[ i, n ]
        return M_m

################################################################################

import numpy as np

# Setup materials, tallies, and forward sources.

def Sigma_t( x ):
    Sigma_t = np.zeros( len( x ) )
    Sigma_t[:] = 0.1
    return Sigma_t

def c( x ):
    Sigma_s = 0.8 * np.ones( len( x ) )
    return Sigma_s

```

```
# Adjoint source from tally.
def Qt_midpoints( xmesh, quadrature ):
    nmesh = len( xmesh )
    ndirs = quadrature.ndirs
    Q = np.zeros( ( nmesh, ndirs ) )
    return Q

def Qt_edges( xmesh, quadrature ):
    nmesh = len( xmesh )
    ndirs = quadrature.ndirs
    Q = np.zeros( ( nmesh, ndirs ) )
    for n in range( ndirs ):
        if( n >= ndirs / 2 ):
            Q[ -1, n ] = 1
    return Q

# Forward source.
def Q_midpoints( xmesh, quadrature ):
    nmesh = len( xmesh )
    ndirs = quadrature.ndirs
    Q = np.zeros( ( nmesh, ndirs ) )
    return Q

def Q_edges( xmesh, quadrature ):
    nmesh = len( xmesh )
    ndirs = quadrature.ndirs
    Q = np.zeros( ( nmesh, ndirs ) )
    for n in range( ndirs ):
        if( n >= ndirs / 2 ):
            Q[ 0, n ] = 1
    return Q

# Initialize quadrature set and mesh.

ndirs = 32
nmesh = 1000
xmin = 0
x_DX = 8
xmax = 10
q = quadrature_Gauss_Legendre( ndirs )
m = mesh( q,
          nmesh = nmesh,
          xmin = xmin,
          x_DX = x_DX,
          xmax = xmax,
          Sigma_t = Sigma_t,
          c = c,
          Qt_midpoints = Qt_midpoints,
          Qt_edges = Qt_edges,
          Q_midpoints = Q_midpoints,
          Q_edges = Q_edges )

# Calculate free-flight probabilities from cell-center and cell-edge positions
# to the DXTRAN region edge at x_DX.
m = calc_pff( m, q )

# Solve for Psi_1.
m = sweep_Psi_1_DXTRAN( m, q )

# Calculate Q_2 from Psi_1-a and Psi_1-b.
m = calc_Q_2_DXTRAN( m, q )

# Solve for Psi_2.
m = sweep_Psi_2_DXTRAN( m, q )

```

```
519 # Plot angular statistical moments for n = 1.
520 plot_Psi_1_and_2( m, 'DXTRAN_Psi_1_2_Separated.pdf' )
```

**Listing F.2:** DXTRAN_Combined_Moments.py

```
#!/usr/bin/env python

def plot_Psi_1_and_2( m, outfilename = None ):
    import matplotlib as mpl
    mpl.use('Agg')
    import matplotlib.pyplot as plt
    plt.figure( figsize = ( 3.25, 3.25/1.62 ) )

    plt.plot( m.x_midpoints,
              m.Psi_1_midpoints[ -1, :, -1 ],
              color = '#000000',
              linestyle = '-',
              label = r'$\Psi_{{1,u,i,n}}$'.format( m.ndirs ) )

    plt.plot( m.x_midpoints,
              m.Psi_2_midpoints[ -1, :, -1 ],
              color = '#0000ff',
              linestyle = '--',
              label = r'$\Psi_{{2,u,i,n}}$'.format( m.ndirs ) )

    plt.axvspan( x_DX, xmax, facecolor = '#00274c', alpha = 0.1, zorder = -10 )

    plt.grid()
    plt.xlabel( r'$x$ [cm]' )
    plt.xlim( ( xmin, xmax ) )
    plt.legend( loc = 'best' )
    plt.tight_layout()
    if( outfilename != None ):
        plt.savefig( outfilename )
    return

class quadrature_Gauss_Legendre( ):
    def __init__( self, ndirs = 2 ):
        from numpy.polynomial import legendre as L
        self.ndirs = ndirs
        # Legendre polynomial coefficients.
        lc = [ 0 for n in range( ndirs ) ] + [ 1 ]
        self.mu = L.legroots( lc )
        # Evaluated polynomial derivatives.
        Pnp = L.legval( self.mu, L.legder( lc ) )
        self.weight = 1.0 / ( (1 - self.mu**2 ) * ( Pnp )**2 )
        return

    def print_quadrature( self ):
        label = 'Quadrature Set'
        print( '\n{:-^80}\n'.format( ' ' + label + ' ' ) )
        print('{:>15s}{:>30s}{:>30s}'.format(
            'n', 'mu', 'weight' ) )
        for n in range( self.ndirs ):
            print('{:15d}{:30.15f}{:30.15f}'.format(
                n, self.mu[n], self.weight[n] ) )
        print( '\n{:-^80}\n'.format( ' End ' + label + ' ' ) )
        return

def calc_pff( mesh, quadrature ):
    # For each mesh outside the DXTRAN region, perform an optical distance
    # integration for it.

    for n in range( quadrature.ndirs ):
        # Skip those directions that cannot hit the DXTRAN region.
        if( n < quadrature.ndirs / 2 ): continue

        for i in range( mesh.idx - 1, -1, -1 ):
```

```
            dist_mesh = np.ones( mesh.idx - i )
            opt_dist_edge = np.sum(
                mesh.Sigma_t[i:mesh.idx] * dist_mesh * mesh.dx[i:mesh.idx] )
            mesh.pff_edges[ i, n ] = np.exp( -opt_dist_edge / quadrature.mu[n] )

            dist_mesh[0] = 0.5
            opt_dist_mid = np.sum(
                mesh.Sigma_t[i:mesh.idx] * dist_mesh * mesh.dx[i:mesh.idx] )
            mesh.pff_midpoints[ i, n ] = np.exp( -opt_dist_mid / quadrature.mu[n] )

    return m

def sweep_Psi_1_DXTRAN( mesh, quadrature, eps = 1e-4, nitermax = 100 ):
    M_1_old = 1
    M_1_new = 0
    eps_rel = 1

    # Setup aliases.
    m = mesh
    q = quadrature
    N = q.ndirs

    u = 0
    for u in range( nitermax-1 ):
        u = u + 1

        # Update scattering source for current iteration based on prior
        # iteration.
        for i in range( m.nmesh ):
            for n in range( N ):
                m.Q_1s_midpoints[ u - 1, i, n ] = \
                    m.Sigma_s[i] * m.Phi_1_midpoints[ u - 1, i ]

        for n in range( N ):
            di = ( 1 if n < N / 2 else -1 )
            iwalk = ( range( 0, m.nmesh , di )
                      if n < N / 2
                      else range( m.nmesh-1, -1, di ) )

            # Perform sweep and accumulate angular moments.
            for i in iwalk:
                # Alias diamond difference factor.
                dd = 2.0 * np.abs( q.mu[ n ] ) / ( m.dx[ i ] )
                if( n < N / 2 ):
                    m.Psi_1_midpoints[ u, i, n ] = \
                        ( dd * m.Psi_1_edges[ u, i, n ] \
                             + m.Q_1s_midpoints[ u - 1, i, n ] ) \
                        / ( dd + m.Sigma_t[ i ] )

                    m.Psi_1_edges[ u, i + di, n ] = \
                        2.0 * m.Psi_1_midpoints[ u, i, n ] \
                            - m.Psi_1_edges[ u, i, n ]
                else:
                    m.Psi_1_midpoints[ u, i, n ] = \
                        (dd * m.Psi_1_edges[ u, i - di, n ] \
                            + m.Q_1s_midpoints[ u - 1, i, n ] ) \
                        / ( dd + m.Sigma_t[ i ] )

                    m.Psi_1_edges[ u, i, n ] = \
                        2.0 * m.Psi_1_midpoints[ u, i, n ] \
                            - m.Psi_1_edges[ u, i - di, n ]

                # Accumulate angular moments of statistical moment.
                m.Phi_1_midpoints[ u, i ] = m.Phi_1_midpoints[ u, i ] \
                    + q.weight[ n ] * m.Psi_1_midpoints[ u, i, n ]
```

```
129
130                 # Accumulate select angular moments for DXTRAN Q_2.
131                 m.Phi_1a_midpoints[ u, i ] = m.Phi_1a_midpoints[ u, i ] \
132                     + q.weight[ n ] \
133                     * ( m.Psi_1_midpoints[ u, i, n ] - \
134                           m.pff_midpoints[ i, n ] * \
135                               m.Psi_1_edges[ u, m.idx, n ] )
136
137                 m.Phi_1b_midpoints[ u, i ] = m.Phi_1b_midpoints[ u, i ] \
138                     + q.weight[ n ] \
139                     * ( m.pff_midpoints[ i, n ] * m.Psi_1_edges[ u, m.idx, n ] )
140
141         # Determine convergence.
142         M_1_new = m.calc_M_m( 1 )
143         eps_rel = np.abs( M_1_new - M_1_old ) / M_1_old
144         print( '  M_1 = {:.5e} ( iter: {:3d}, eps: {:.3e} )'.format(
145             M_1_new, u, eps_rel ) )
146         if( eps_rel < eps ):
147             print( '  M_1 Converged.' )
148             break
149         else:
150             m.grow_Psi_1()
151             M_1_old = M_1_new
152
153     return m
154
155 def calc_Q_2_DXTRAN( mesh, quadrature ):
156     for n in range( quadrature.ndirs ):
157         mesh.Q_2_midpoints[ -1, :, n ] = 2.0 \
158             * mesh.Sigma_s * mesh.Phi_1a_midpoints[-1,:] \
159             * mesh.Sigma_s * mesh.Phi_1b_midpoints[-1,:] \
160             / mesh.Sigma_s
161     return mesh
162
163 def sweep_Psi_2_DXTRAN( mesh, quadrature, eps = 1e-4, nitermax = 100 ):
164     M_2_old = 1
165     M_2_new = 0
166     eps_rel = 1
167
168     # Setup aliases.
169     m = mesh
170     q = quadrature
171     N = q.ndirs
172
173     u = 0
174     for u in range( nitermax-1 ):
175         u = u + 1
176
177         # Update scattering source for current iteration based on prior
178         # iteration.
179         for i in range( m.nmesh ):
180             for n in range( N ):
181                 m.Q_2s_midpoints[ u - 1, i, n ] = \
182                     m.Sigma_s[i] * m.Phi_2_midpoints[ u - 1, i ]
183
184         for n in range( N ):
185             di = ( 1 if n < N / 2 else -1 )
186             iwalk = ( range( 0, m.nmesh , di )
187                       if n < N / 2
188                       else range( m.nmesh-1, -1, di ) )
189
190             # Perform sweep and accumulate angular moments.
191             for i in iwalk:
192                 # Alias diamond difference factor.
193                 dd = 2.0 * np.abs( q.mu[ n ] ) / ( m.dx[ i ] )
```

```
                if( n < N / 2 ):
                    m.Psi_2_midpoints[ u, i, n ] = \
                        ( dd * m.Psi_2_edges[ u, i, n ] \
                             + m.Q_2s_midpoints[ u - 1, i, n ] \
                             + m.Q_2_midpoints[ -1, i, n ] ) \
                        / ( dd + m.Sigma_t[ i ] )

                    m.Psi_2_edges[ u, i + di, n ] = \
                        2.0 * m.Psi_2_midpoints[ u, i, n ] \
                            - m.Psi_2_edges[ u, i, n ]
                else:
                    m.Psi_2_midpoints[ u, i, n ] = \
                        (dd * m.Psi_2_edges[ u, i - di, n ] \
                            + m.Q_2s_midpoints[ u - 1, i, n ] \
                            + m.Q_2_midpoints[ -1, i, n ] ) \
                        / ( dd + m.Sigma_t[ i ] )

                    m.Psi_2_edges[ u, i, n ] = \
                        2.0 * m.Psi_2_midpoints[ u, i, n ] \
                            - m.Psi_2_edges[ u, i - di, n ]

                # Accumulate angular moments of statistical moment.
                p = 0.5
                parb = 1.0 * 0.5 # Factor of 0.5 for quadrature weight correction
                beta = 1.0
                m.Phi_2_midpoints[ u, i ] = m.Phi_2_midpoints[ u, i ] \
                    + q.weight[ n ] \
                    * ( m.Psi_2_midpoints[ u, i, n ] \
                        - m.pff_midpoints[ i, n ] * m.Psi_2_edges[ u, m.idx, n ] ) \
                    + q.weight[ n ] \
                    * ( m.pff_midpoints[ i, n ]**2 / beta * p**2 / parb * m.Psi_2_edges[ u, m.idx, n ] )

        # Compute M_2.
        # Integrate over cell edges.
        M_2_new = m.calc_M_m( 2 )
        eps_rel = np.abs( M_2_new - M_2_old ) / M_2_old
        print( '  M_2 = {:.5e} ( iter: {:3d}, eps: {:.3e} )'.format(
            M_2_new, u, eps_rel ) )
        if( eps_rel < eps ):
            print( '  M_2 Converged.' )
            break
        else:
            m.grow_Psi_2()
            M_2_old = M_2_new

    return m

class mesh( ):
    def __init__( self,
        quadrature = quadrature_Gauss_Legendre( 2 ),
        nmesh = 100,
        nitermax = 10,
        xmin = 0,
        x_DX = 5,
        xmax = 10,
        Sigma_t = None,
        c = None,
        Qt_midpoints = None,
        Qt_edges = None,
        Q_midpoints = None,
        Q_edges = None  ):

        import numpy as np

        # Alias quadrature.
```

```
259         q = quadrature
260 
261         # Setup spatial mesh.
262         self.nmesh = nmesh
263         self.ndirs = q.ndirs
264         self.x_edges = np.linspace( xmin, xmax, self.nmesh + 1 )
265         self.x_midpoints = self.x_edges[:-1] + 0.5 * np.diff( self.x_edges )
266         self.dx = np.diff( self.x_edges )
267 
268         # Assign materials to mesh cells.
269         if( Sigma_t == None ):
270             self.Sigma_t = np.ones( nmesh )
271         else:
272             self.Sigma_t = Sigma_t( self.x_midpoints )
273         if( c == None ):
274             self.Sigma_s = np.zeros( nmesh )
275         else:
276             self.Sigma_s = c( self.x_midpoints ) * self.Sigma_t
277 
278         # Assign cell-center and cell-edge tally sources.
279         if( Qt_midpoints == None ):
280             self.Qt_midpoints = np.zeros( ( nmesh, q.ndirs ) )
281         else:
282             self.Qt_midpoints = Qt_midpoints( self.x_midpoints, q )
283         if( Qt_edges == None ):
284             self.Qt_edges = np.zeros( ( nmesh, q.ndirs ) )
285         else:
286             self.Qt_edges = Qt_edges( self.x_edges, q )
287 
288         # Assign cell-center and cell-edge forward sources.
289         if( Q_midpoints == None ):
290             self.Q_midpoints = np.zeros( ( nmesh, q.ndirs ) )
291         else:
292             self.Q_midpoints = Q_midpoints( self.x_midpoints, q )
293         if( Q_edges == None ):
294             self.Q_edges = np.zeros( ( nmesh, q.ndirs ) )
295         else:
296             self.Q_edges = Q_edges( self.x_edges, q )
297 
298         # Initialize angular moment storage arrays, Psi_1, Phi_1, Q_1s.
299         self.Psi_1_midpoints = np.zeros(
300             ( 2, nmesh,  ndirs ) ) # u, i, n
301         self.Psi_1_edges = np.zeros(
302             ( 2, nmesh + 1, ndirs) ) # u, i, n
303         self.Phi_1_midpoints = np.zeros(
304             ( 2, nmesh ) ) # u, i, r = 1
305         self.Phi_1a_midpoints = np.zeros(
306             ( 2, nmesh ) ) # u, i, r = 1
307         self.Phi_1b_midpoints = np.zeros(
308             ( 2, nmesh ) ) # u, i, r = 1
309         self.Q_1s_midpoints = np.zeros(
310             ( 2, nmesh, ndirs ) ) # u, i, n
311 
312         # Initialize DXTRAN Q_2.
313         self.Q_2_midpoints = np.zeros(
314             ( 1, nmesh, ndirs ) ) # u, i, n
315 
316         # Initialize angular moment storage arrays, Psi_2, Phi_2, Q_2s.
317         self.Psi_2_midpoints = np.zeros(
318             ( 2, nmesh, ndirs ) ) # u, i, n
319         self.Psi_2_edges = np.zeros(
320             ( 2, nmesh + 1, ndirs ) ) # u, i, n
321         self.Phi_2_midpoints = np.zeros(
322             ( 2, nmesh ) ) # u, i, r = 1
323         self.Q_2s_midpoints = np.zeros(
```

```
            ( 2, nmesh, ndirs ) ) # u, i, n

        # Place boundary sources onto mesh.
        #         psi_edges[ 0, :, : ] = src_edges[ 0, : ]
        for u in range( 2 ):
            self.Psi_1_edges[ u, :, : ] = self.Qt_edges[ :, : ]
            self.Psi_2_edges[ u, :, : ] = self.Qt_edges[ :, : ]**2


        # DXTRAN cell-edge index for DXTRAN surface.
        self.idx = np.where( self.x_edges == x_DX )[0][0]

        # DXTRAN free-flight probabilities.
        self.pff_midpoints = np.zeros( ( nmesh, ndirs ) )
        self.pff_edges = np.zeros( ( nmesh + 1, ndirs ) )

        return

    def grow_Psi_1( self ):
        # Grow arrays for next iteration.
        self.Psi_1_midpoints = np.vstack(
            ( self.Psi_1_midpoints, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )
        self.Psi_1_edges = np.vstack(
            ( self.Psi_1_edges, np.zeros(
                ( 1, self.nmesh + 1, self.ndirs ) ) ) )
        self.Phi_1_midpoints = np.vstack(
            ( self.Phi_1_midpoints, np.zeros(
                ( 1, self.nmesh ) ) ) )
        self.Phi_1a_midpoints = np.vstack(
            ( self.Phi_1a_midpoints, np.zeros(
                ( 1, self.nmesh ) ) ) )
        self.Phi_1b_midpoints = np.vstack(
            ( self.Phi_1b_midpoints, np.zeros(
                ( 1, self.nmesh ) ) ) )
        self.Q_1s_midpoints = np.vstack(
            ( self.Q_1s_midpoints, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )

        # Emplace tally source(s).
        self.Psi_1_edges[ -1, :, : ] = self.Qt_edges[ :, : ]
        return

    def grow_Psi_2( self ):
        # Grow arrays for next iteration.
        self.Psi_2_midpoints = np.vstack(
            ( self.Psi_2_midpoints, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )
        self.Psi_2_edges = np.vstack(
            ( self.Psi_2_edges, np.zeros(
                ( 1, self.nmesh + 1, self.ndirs ) ) ) )
        self.Phi_2_midpoints = np.vstack(
            ( self.Phi_2_midpoints, np.zeros(
                ( 1, self.nmesh ) ) ) )
        self.Q_2s_midpoints = np.vstack(
            ( self.Q_2s_midpoints, np.zeros(
                ( 1, self.nmesh, self.ndirs ) ) ) )

        # Emplace tally source(s).
        self.Psi_2_edges[ -1, :, : ] = self.Qt_edges[ :, : ]**2
        return

    def calc_M_m( self, m = 1 ):
        if( m == 1 ):
            Psi_m = self.Psi_1_edges[ -1, :, : ]
```

```
        else:
            Psi_m = self.Psi_2_edges[ -1, :, : ]

        # Integrate over cell edges.
        M_m = 0
        for i in range( self.nmesh + 1 ):
            for n in range( q.ndirs ):
                M_m = M_m + \
                    self.Q_edges[ i, n ] \
                    * q.weight[ n ] * Psi_m[ i, n ]

        # Integrate over cell centers.
        for i in range( self.nmesh ):
            for n in range( q.ndirs ):
                M_m = M_m + \
                    self.Q_midpoints[ i, n ] \
                    * q.weight[ n ] * Psi_m[ i, n ]
        return M_m

################################################################################

import numpy as np

# Setup materials, tallies, and forward sources.

def Sigma_t( x ):
    Sigma_t = np.zeros( len( x ) )
    Sigma_t[:] = 0.1
    return Sigma_t

def c( x ):
    Sigma_s = 0.8 * np.ones( len( x ) )
    return Sigma_s

# Adjoint source from tally.
def Qt_midpoints( xmesh, quadrature ):
    nmesh = len( xmesh )
    ndirs = quadrature.ndirs
    Q = np.zeros( ( nmesh, ndirs ) )
    return Q

def Qt_edges( xmesh, quadrature ):
    nmesh = len( xmesh )
    ndirs = quadrature.ndirs
    Q = np.zeros( ( nmesh, ndirs ) )
    for n in range( ndirs ):
        if( n >= ndirs / 2 ):
            Q[ -1, n ] = 1
    return Q

# Forward source.
def Q_midpoints( xmesh, quadrature ):
    nmesh = len( xmesh )
    ndirs = quadrature.ndirs
    Q = np.zeros( ( nmesh, ndirs ) )
    return Q

def Q_edges( xmesh, quadrature ):
    nmesh = len( xmesh )
    ndirs = quadrature.ndirs
    Q = np.zeros( ( nmesh, ndirs ) )
    for n in range( ndirs ):
        if( n >= ndirs / 2 ):
            Q[ 0, n ] = 1
    return Q
```

```
454
455 # Initialize quadrature set and mesh.
456
457 ndirs = 32
458 nmesh = 1000
459 xmin = 0
460 x_DX = 8
461 xmax = 10
462 q = quadrature_Gauss_Legendre( ndirs )
463 m = mesh( q,
464           nmesh = nmesh,
465           xmin = xmin,
466           x_DX = x_DX,
467           xmax = xmax,
468           Sigma_t = Sigma_t,
469           c = c,
470           Qt_midpoints = Qt_midpoints,
471           Qt_edges = Qt_edges,
472           Q_midpoints = Q_midpoints,
473           Q_edges = Q_edges )
474
475 # Calculate free-flight probabilities from cell-center and cell-edge positions
476 # to the DXTRAN region edge at x_DX.
477 m = calc_pff( m, q )
478
479 # Solve for Psi_1.
480 m = sweep_Psi_1_DXTRAN( m, q )
481
482 # Calculate Q_2 from Psi_1-a and Psi_1-b.
483 m = calc_Q_2_DXTRAN( m, q )
484
485 # Solve for Psi_2.
486 m = sweep_Psi_2_DXTRAN( m, q )
487
488 # Plot angular statistical moments for n = 1.
489 plot_Psi_1_and_2( m, 'DXTRAN_Psi_1_2_Combined.pdf' )
```

**Listing F.3:** Differences between Separated- and Combined-moment Files

```
117d116
<                       Psi_1_in = ( 0 if i == m.idx - 1 else m.Psi_1_edges[ u, i - di, n ] )
119c118
<                           (dd * Psi_1_in \
---
>                           (dd * m.Psi_1_edges[ u, i - di, n ] \
125,131c124
<                               - Psi_1_in
<
<                       if( i < m.idx ):
<                           m.Psi_1_midpoints_DX[ u, i, n ] = \
<                               m.pff_midpoints[ i, n ] * m.Psi_1_edges[ u, m.idx, n ]
<                           m.Psi_1_edges_DX[ u, i, n ] = \
<                               m.pff_edges[ i, n ] * m.Psi_1_edges[ u, m.idx, n ]
---
>                               - m.Psi_1_edges[ u, i - di, n ]
135,136c128
<                       + q.weight[ n ] * m.Psi_1_midpoints[ u, i, n ] \
<                       + q.weight[ n ] * m.Psi_1_midpoints_DX[ u, i, n ]
---
>                       + q.weight[ n ] * m.Psi_1_midpoints[ u, i, n ]
140c132,135
<                       + q.weight[ n ] * m.Psi_1_midpoints[ u, i, n ]
---
>                       + q.weight[ n ] \
>                       * ( m.Psi_1_midpoints[ u, i, n ] - \
>                               m.pff_midpoints[ i, n ] * \
>                                   m.Psi_1_edges[ u, m.idx, n ] )
210d204
<                       Psi_2_in = ( 0 if i == m.idx - 1 else m.Psi_2_edges[ u, i - di, n ] )
212c206
<                           (dd * Psi_2_in \
---
>                           (dd * m.Psi_2_edges[ u, i - di, n ] \
219,225c213
<                               - Psi_2_in
<
<                       if( i < m.idx ):
<                           m.Psi_2_midpoints_DX[ u, i, n ] = \
<                               m.pff_midpoints[ i, n ] * m.Psi_2_edges[ u, m.idx, n ]
<                           m.Psi_2_edges_DX[ u, i, n ] = \
<                               m.pff_edges[ i, n ] * m.Psi_2_edges[ u, m.idx, n ]
---
>                               - m.Psi_2_edges[ u, i - di, n ]
232c220,222
<                       + q.weight[ n ] * m.Psi_2_midpoints[ u, i, n ] \
---
>                       + q.weight[ n ] \
>                       * ( m.Psi_2_midpoints[ u, i, n ] \
>                           - m.pff_midpoints[ i, n ] * m.Psi_2_edges[ u, m.idx, n ] ) \
341a332
>
349,358d339
<         # DXTRAN analytic components.
<         self.Psi_1_midpoints_DX = np.zeros(
<             ( 2, nmesh,  ndirs ) ) # u, i, n
<         self.Psi_1_edges_DX= np.zeros(
<             ( 2, nmesh + 1, ndirs) ) # u, i, n
<         self.Psi_2_midpoints_DX = np.zeros(
<             ( 2, nmesh,  ndirs ) ) # u, i, n
<         self.Psi_2_edges_DX= np.zeros(
<             ( 2, nmesh + 1, ndirs) ) # u, i, n
<
```

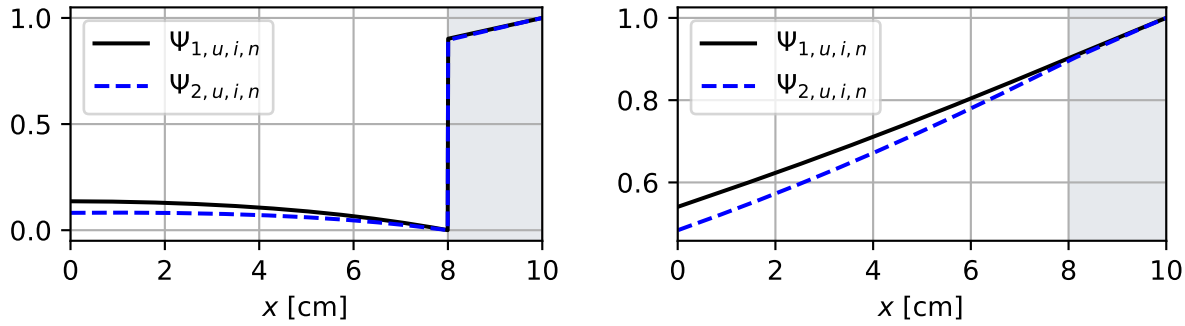
```
381,386d361
<          self.Psi_1_midpoints_DX = np.vstack(
<              ( self.Psi_1_midpoints_DX, np.zeros(
<                  ( 1, self.nmesh, self.ndirs ) ) ) )
<          self.Psi_1_edges_DX = np.vstack(
<              ( self.Psi_1_edges_DX, np.zeros(
<                  ( 1, self.nmesh + 1, self.ndirs ) ) ) )
406,411d380
<          self.Psi_2_midpoints_DX = np.vstack(
<              ( self.Psi_2_midpoints_DX, np.zeros(
<                  ( 1, self.nmesh, self.ndirs ) ) ) )
<          self.Psi_2_edges_DX = np.vstack(
<              ( self.Psi_2_edges_DX, np.zeros(
<                  ( 1, self.nmesh + 1, self.ndirs ) ) ) )
419c388
<              Psi_m = self.Psi_1_edges[ -1, :, : ] + self.Psi_1_edges_DX[ -1, :, : ]
---
>              Psi_m = self.Psi_1_edges[ -1, :, : ]
421c390
<              Psi_m = self.Psi_2_edges[ -1, :, : ] + self.Psi_2_edges_DX[ -1, :, : ]
---
>              Psi_m = self.Psi_2_edges[ -1, :, : ]
520c489
< plot_Psi_1_and_2( m, 'DXTRAN_Psi_1_2_Separated.pdf' )
---
> plot_Psi_1_and_2( m, 'DXTRAN_Psi_1_2_Combined.pdf' )
```

**Listing F.4:** Separated-moment Solver Iterations

```
  M_1 = 7.42469e-02 ( iter:   1, eps: 9.258e-01 )
  M_1 = 1.19176e-01 ( iter:   2, eps: 6.051e-01 )
  M_1 = 1.44616e-01 ( iter:   3, eps: 2.135e-01 )
  M_1 = 1.58144e-01 ( iter:   4, eps: 9.354e-02 )
  M_1 = 1.65095e-01 ( iter:   5, eps: 4.396e-02 )
  M_1 = 1.68604e-01 ( iter:   6, eps: 2.125e-02 )
  M_1 = 1.70358e-01 ( iter:   7, eps: 1.041e-02 )
  M_1 = 1.71231e-01 ( iter:   8, eps: 5.124e-03 )
  M_1 = 1.71664e-01 ( iter:   9, eps: 2.530e-03 )
  M_1 = 1.71879e-01 ( iter:  10, eps: 1.251e-03 )
  M_1 = 1.71986e-01 ( iter:  11, eps: 6.191e-04 )
  M_1 = 1.72038e-01 ( iter:  12, eps: 3.064e-04 )
  M_1 = 1.72064e-01 ( iter:  13, eps: 1.517e-04 )
  M_1 = 1.72077e-01 ( iter:  14, eps: 7.510e-05 )
  M_1 Converged.
  M_2 = 8.65744e-02 ( iter:   1, eps: 9.134e-01 )
  M_2 = 1.10983e-01 ( iter:   2, eps: 2.819e-01 )
  M_2 = 1.22862e-01 ( iter:   3, eps: 1.070e-01 )
  M_2 = 1.28641e-01 ( iter:   4, eps: 4.704e-02 )
  M_2 = 1.31412e-01 ( iter:   5, eps: 2.154e-02 )
  M_2 = 1.32724e-01 ( iter:   6, eps: 9.986e-03 )
  M_2 = 1.33340e-01 ( iter:   7, eps: 4.643e-03 )
  M_2 = 1.33628e-01 ( iter:   8, eps: 2.159e-03 )
  M_2 = 1.33762e-01 ( iter:   9, eps: 1.004e-03 )
  M_2 = 1.33825e-01 ( iter:  10, eps: 4.665e-04 )
  M_2 = 1.33854e-01 ( iter:  11, eps: 2.167e-04 )
  M_2 = 1.33867e-01 ( iter:  12, eps: 1.006e-04 )
  M_2 = 1.33874e-01 ( iter:  13, eps: 4.674e-05 )
  M_2 Converged.
```

**Listing F.5:** Combined-moment Solver Iterations

```
  M_1 = 7.42469e-02 ( iter:   1, eps: 9.258e-01 )
  M_1 = 1.19176e-01 ( iter:   2, eps: 6.051e-01 )
  M_1 = 1.44616e-01 ( iter:   3, eps: 2.135e-01 )
  M_1 = 1.58144e-01 ( iter:   4, eps: 9.354e-02 )
  M_1 = 1.65095e-01 ( iter:   5, eps: 4.396e-02 )
  M_1 = 1.68604e-01 ( iter:   6, eps: 2.125e-02 )
  M_1 = 1.70358e-01 ( iter:   7, eps: 1.041e-02 )
  M_1 = 1.71231e-01 ( iter:   8, eps: 5.124e-03 )
  M_1 = 1.71664e-01 ( iter:   9, eps: 2.530e-03 )
  M_1 = 1.71879e-01 ( iter:  10, eps: 1.251e-03 )
  M_1 = 1.71986e-01 ( iter:  11, eps: 6.191e-04 )
  M_1 = 1.72038e-01 ( iter:  12, eps: 3.064e-04 )
  M_1 = 1.72064e-01 ( iter:  13, eps: 1.517e-04 )
  M_1 = 1.72077e-01 ( iter:  14, eps: 7.510e-05 )
  M_1 Converged.
  M_2 = 8.65744e-02 ( iter:   1, eps: 9.134e-01 )
  M_2 = 1.10983e-01 ( iter:   2, eps: 2.819e-01 )
  M_2 = 1.22862e-01 ( iter:   3, eps: 1.070e-01 )
  M_2 = 1.28641e-01 ( iter:   4, eps: 4.704e-02 )
  M_2 = 1.31412e-01 ( iter:   5, eps: 2.154e-02 )
  M_2 = 1.32724e-01 ( iter:   6, eps: 9.986e-03 )
  M_2 = 1.33340e-01 ( iter:   7, eps: 4.643e-03 )
  M_2 = 1.33628e-01 ( iter:   8, eps: 2.159e-03 )
  M_2 = 1.33763e-01 ( iter:   9, eps: 1.004e-03 )
  M_2 = 1.33825e-01 ( iter:  10, eps: 4.665e-04 )
  M_2 = 1.33854e-01 ( iter:  11, eps: 2.167e-04 )
  M_2 = 1.33867e-01 ( iter:  12, eps: 1.006e-04 )
  M_2 = 1.33874e-01 ( iter:  13, eps: 4.674e-05 )
  M_2 Converged.
```




**(a)** Separated-moment Solution, $n = 32$

**(b)** Combined-moment Solution, $n = 32$

**Figure F.1:** Converged Separated- and Combined-moment Solutions for $\widehat{\Psi}_1$ and $\widehat{\Psi}_2$