Weight Window Isosurface Geometries for Monte Carlo Radiation Transport Variance Reduction

by

Kalin R. Kiesling

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy
(Nuclear Engineering and Engineering Physics)

at the

UNIVERSITY OF WISCONSIN–MADISON

2022

Date of final oral examination: 2022-01-18

The dissertation is approved by the following members of the Final Oral Committee:

Paul P.H. Wilson, Professor, Engineering Physics
Douglass Henderson, Professor, Engineering Physics
Benjamin Lindley, Assistant Professor, Engineering Physics
Dan Negrut, Professor, Mechanical Engineering
Thomas Evans, Distinguished R & D Staff, Oak Ridge National Lab
Gregory Davidson, Senior R & D Staff, Oak Ridge National Lab

I dedicate this to my daughter Riley who simultaneously made this my most difficult yet most worthwhile accomplishment ever.

Acknowledgments

"The good, the bad and the ugly"

I have a lot of people, groups, and entities to acknowledge as having contributed to the completion of my PhD. Most people tend to only acknowledge the "good" that helped them succeed, but I want to also acknowledge the "bad" and the "ugly" that all contributed to both the successes and challenges faced over the years. Without the bad and the ugly, in addition to all the good, I would not be the scholar nor person that I am at the completion of this degree.

The Good

First, I want to thank specific members (past and present) of the Computational Nuclear Engineering Research Group (CNERG) for providing invaluable technical support throughout the years: Patrick Shriwise for his invaluable expertise on MOAB and DAGMC; Andrew Davis and Baptiste Mouginot for their scientific guidance; and Yuehan Qin for her contributions to the mesh simplification and refinement analysis of this dissertation. Thank you also to the computational resources and assistance provided by the Center for High Throughput Computing at UW-Madison.

I want to thank the various funding sources over the years that allowed me to pursue this research: the Nuclear Regulatory Commission Graduate Fellowship (NRC-HQ-84-14-G-0030); the Department of Energy Office of Fusion Energy Sciences (DE-SC0017122); and the Wisconsin Distinguished Graduate Fellowship. In addition to research funding, I also want to acknowledge the additional monetary support I received that allowed me to live a decent life while in grad school. The American Nuclear Society provided me with various supplementary scholarships over the years to aide in general life expenses. I want to give a special thanks to the Office of Child Care and Family Resources at UW-Madison that provided us with childcare tuition assistance, allowing us to afford childcare for our daughter and focus on our studies (primarily before the pandemic). Without adequate funding, I would not have been able to focus on research and professional growth.

Last, and most certainly not least, I want to acknowledge the people and groups in my life that contributed to the moral support required to stay somewhat sane during a PhD. I cannot give enough thanks and love to my husband Brian Cornille who helped me through many challenging technical situations and stood by my side during all of the "bad" and the "ugly," in addition to celebrating all the "good" with me too. Thank you to my family, including my parents, siblings, and in-laws, who provided invaluable life support over the years and encouraged me throughout all the challenges. Thank you to all the women in nuclear who have come before me and have shown me what success in this field looks like. I want to specifically thank my advisor Paul Wilson for not only his (expected) technical and scientific guidance, but for his moral support and understanding when I went through my fair share of life challenges, all of which could have easily caused me to drop out of the

program had I not been given the grace I received. Lastly, thank you, from the bottom of my heart, to the other mamas in the Academic Mamas 2019 Facebook group who provided me with laughs, tears, and moral support as I navigated becoming a mom as a grad student and parenting during a global pandemic. I know for a fact that I would never have completed this degree without their support.

The Bad

Now on to the "bad," which is to say, all those who made this degree unnecessarily difficult or shut doors on me. I find it important to acknowledge such entities in writing because too many students have gone through grad school facing similar "bad" and it gets swept under the rug.

The most important thing I want to acknowledge is all the government officials and individuals in this country that did not, and continue to not, take a global pandemic seriously, which has led to me (and my husband) working for nearly the entire pandemic with a baby-turned-toddler at home in our full time care. I knew being a student-mother would challenging, but it was never supposed to be *this* challenging. We lost our regular childcare; we lost the ability to work with a focused mind; we lost the ability to take a very young child into public spaces without fear she may contract a dangerous virus; we lost the ability to do anything except to try to raise a child the best way possible given the circumstances while trying to fit in a few minutes or hours of research during nap time or late at night. The lack of societal support for families and individual disregard for public health during this time has led to my degree completion taking an additional year longer, many mental health crises, and raising a daughter who has hardly seen the outside of her home. However, if there is one silver lining in this "bad", it is the realization that even if individuals are good people, no career or organization will ever truly care for the well-being of you and your family. So thank you to all those who contributed to this pandemic and allowed me to set my priorities in life straight by focusing on my family.

I also want to thank the US Navy for giving me a medical rejection from a desk job due to minor scoliosis. Without that ridiculous and soul-crushing (at the time) rejection, I would not have even considered pursing a PhD. Thank you, US Navy, for changing my career path and making me realize I was a perfect fit for a PhD program instead.

The Ugly

And now I want to acknowledge all the "ugly" of my journey, meaning everything that might be uncomfortable to think about and acknowledge, but is an important factor in my achievement.

First, I want to acknowledge what most find uncomfortable: my privilege. I am lucky enough to come from a financially stable family such that I could complete an undergraduate and graduate degree without loans or having to work a second job, which is a large contributing factor to my success over the years. Without having to work a job, I was able to focus on my studies and get good grades. I was able to spend additional time contributing

volunteer hours to a student organization and professional organization that boosted my network and professional development. With financial stability, I was able to front the cost, or even pay out of pocket in some cases, to attend professional conferences, which is where I greatly expanding my professional network and put my name on the map. Without the burden of student loans and having a financial safety net, I was able to live well enough in grad school to never have to worry about making rent or having enough money for food, a privilege many grad students are not afforded.

With this financial security, I was also in a position to be able to start a family in grad school, which leads me to the other privilege I want to acknowledge: parental leave and parental support. I am acknowledging these two things in this section, because while they are "good" in that I was able to have such support, it is "ugly" that I have to list this as a privilege. Through the College of Engineering graduate student parental leave policy, I was afforded 12 weeks, and my husband 6 weeks, of fully paid maternity leave. This is 12 weeks more than the majority of graduate students and workers are given in this country. I would not have been able to return to my studies after giving birth without the time to recover and learn to raise an infant, and yet, it still was not enough time. Additionally, UW-Madison has provided wonderful support to student-parents (outside the pandemic) through financial aide and workshops. I am lucky to have had this organizational support during this time.

Lastly, I want to make one last acknowledgement to the therapy I received all through grad school. This is an "ugly" acknowledgement not only because this topic is often taboo, but also because I find it egregious that one of the top recommendations to students to make sure they get through a PhD is to enter therapy. It should not be the norm to have to get therapy just to get through a degree. And yet, I know that I would have never finished a degree if it weren't for the therapy I received for a variety of graduate school related reasons over the last 3/4 of a decade. Thank you to therapy for literally ensuring my survival during this time.

Contents

List of Tables

List of Figures

Nomenclature

ADVANTG AutomateD VAriaNce reducTion Generator

CADIS Consistent Adjoint-Driven Importance Sampling

CADIS-Ω Angle-informed CADIS

CWWM Cartesian WW mesh

DAG-MCNP DAGMC coupled with MCNP

DAGMC Direct Accelerated Geometry Monte Carlo

FOM figure of merit

FW-CADIS Forward-Weighted CADIS

GT-CADIS Groupwise-Transmutation CADIS

MAGIC Method of Automatic Generation of Importances by Calculation

MC Monte Carlo

MCNP Monte Carlo N-Particle transport code

MOAB Mesh Oriented datABase

MS-CADIS Multi-Step CADIS

OBB oriented bounding box

PARTISN PARallel TIme-dependent SN

PDF probability distribution function

PyNE Python for Nuclear Engineering

VOV variance of the variance

VR variance reduction

WW weight window

WWIG WW isosurface geometry

Abstract

In order to perform accurate Monte Carlo (MC) simulations, which is a stochastic method resulting in uncertainty, variance reduction (VR) techniques are often necessary to reduce the relative error for quantities of interest. The use of weight windows (WWs) is a common VR method in which the statistical weight of particles are changed based on various parameters in the simulation. WWs are most commonly represented as a Cartesian WW mesh (CWWM) where WWs are defined across all energies on each mesh voxel. For large, geometrically complex problems, these meshes often need to be developed with fine resolution over the entire spatial domain in order to capture necessary fine detail in some regions of the geometry. This can cause the memory footprint of these meshes to be extremely large and computationally prohibitive. Furthermore, CWWMs are not necessarily efficient in their implementation with respect to when particle weight is checked and updated.

This dissertation work presents a novel method for representing WWs aimed at addressing the computational limitations of CWWMs while also improving VR efficiency. In this method, the WWs are transformed into a faceted mesh geometry, known as a WW isosurface geometry (WWIG), where the surfaces are the isosurfaces derived from the WW values in a CWWM. The WWIGs can then be used during particle tracking with the Direct Accelerated Geometry Monte Carlo (DAGMC) toolkit, which allows for particle tracking on arbitrarily complex geometries.

In this work, an algorithm for using WWIGs for MC VR has been implemented in DAGMC coupled with Monte Carlo N-Particle transport code (MCNP) (DAG-MCNP) 6.2. Initial verification and demonstration experiments show that the WWIG method performs accurate and comparable VR to using CWWMs. Further analysis has been done to demonstrate how changing mesh geometric features of the WWIGs affects computational performance during MC radiation transport. Depending on parameters set for generating the WWIGs and the starting CWWM, the isosurfaces of the WWIGs can vary in mesh coarseness, surface roughness, and spacing. In this work, we explore how these different geometric features of the WWIGs affect the memory footprint and computational performance during variance reduction for Monte Carlo radiation transport. In the end, we see that using WWIGs for MC VR improves WW efficiency and is comparable in performance to using CWWMs.

Chapter 1

Introduction and Motivation

The design of nuclear systems generally requires the analysis of nuclear radiation using computational radiation transport methods. Typically, the Monte Carlo (MC) radiation transport method is a stochastic method used in the analysis of large complex nuclear systems, such as reactors and complex shielding setups. The MC method is generally used over deterministic methods because they are more readily parallelized, whereas deterministic methods can become computationally prohibitive as the size and complexity of a system increases. However, even when using MC methods for these large complex systems, performing accurate analysis can become time consuming and computationally prohibitive to reach the desired level of fidelity. To address this, analysts employ statistical computational methods, known as variance reduction (VR) techniques, during the MC transport process that aim to reduce the uncertainty of a simulation while also potentially reducing the computational resources (such as time or memory) required. However, as the ability and desire to design even larger and more detailed systems on paper progresses, we are still faced with computational limitations even with current VR techniques employed. This dissertation work aims to further improve VR efficiency and reduce the computational burden during complex MC radiation transport analysis.

One current and common method for VR, and the method this dissertation work is based on, is the use of weight windows (WWs) in which the statistical weight of a particle is altered based on its position in phase space and how important that position is considered to be to the end goal of the simulation. Most commonly WW values are defined on a Cartesian grid, here after referred to as a Cartesian WW mesh (CWWM), and can then be used in

MC transport analysis. Historically, these CWWMs, which can be generated in a number of ways, often require fine mesh detail across the full spatial domain of the transport geometry in order to capture areas of fine geometric detail, whether or not it is needed everywhere in the domain. As a result, these meshes can have a large memory footprint that can be computationally prohibitive when a certain level of fidelity is desired in a simulation. Furthermore, the implementation of using CWWMs in practice is not necessarily efficient when it comes to actually applying these WWs. When using these meshes for VR with the Monte Carlo N-Particle transport code (MCNP) [1], particle weights are checked, by default, at every collision, surface crossing, and mean free path traveled. However, at these checkpoints, the underlying WWs defined on the mesh may not have changed significantly enough to warrant initiating a variance reduction event. In this case, the WW checks can be inefficient and may not occur when it would be truly beneficial to check and update the particle weight.

The novel method presented in this dissertation uses WW isosurface geometries (WWIGs) generated from CWWMs, in MC simulations. By representing the WW meshes as a geometry of isosurfaces derived from the CWWM rather than a dense mesh, the necessary fine detail about the WWs is only stored at the isosurfaces rather than the entire spatial domain. In other words, information is only stored where the WW change is expected to matter or when the WW changes significantly to warrant a particle weight check. When used in simulations, the particle weight is only checked and updated at a WWIG surface crossing rather than every collision, surface crossing, and mean free path traveled. The Direct Accelerated Geometry Monte Carlo (DAGMC) toolkit, which is a geometry toolkit that can be coupled with MC codes to perform particle tracking on complex geometries with arbitrarily higher order surfaces, is what enables the use of the WWIGs. The goal of this method is to reduce memory footprint and improve VR efficiency while maintaining necessary levels of VR.

This dissertation will first go over background information on MC VR and the DAGMC toolkit in Chapter 2. Chapter 3 presents the novel research on the generation of WWIGs,

how they are used in radiation transport analysis, and a preliminary analysis of common geometry and mesh characteristics. Chapter 4 provides an experiment on the verification of the WWIG method implementation and a demonstration of their use. Chapter 5 covers additional background on faceted meshes, methods for measuring mesh characteristics, and mesh simplification methods, all of which have been implemented for modifying and improving WWIGs. In Chapter 6, three different sets of experiments are shown to understand how variations in different WWIG features affect computational performance compared to that of the traditional CWWM method. Finally, Chapters 7 and 8 contain an overall summary and conclusions of this work as well as possible future work for this method.

Chapter 2

Background

2.1 Monte Carlo Radiation Transport

In computational analysis of radiation transport in nuclear systems, it is common to use the Monte Carlo (MC) method in which the transport of particles is simulated as a series of stochastic events. This stochastic method treats space and energy continuously making it suitable for radiation transport analysis of large, complex systems. However, MC simulations of large complex systems usually still require variance reduction (VR) methods in order to adequately reduce the relative error within the limitations of computational resources. This chapter will give an overview of the MC method, VR techniques, and how weight window (WW)s are currently used for VR.

The MC method is a stochastic method for solving the transport equation in which the transport of a neutron particle through matter is defined by probabilistic events, or random walks. Each random event is sampled sequentially from underlying probability distribution function (PDF)s that represent the likelihood of each event occurring. The PDFs are defined by the physical parameters (such as position, energy, direction, and collision outcomes) for the problem at hand [1, 2, 3, 4]. Each particle is tracked from its source through a series of random events in phase space until it is terminated (e.g., it is absorbed, escapes, etc.). This path, or series of events, is known as the particle's history. The quantities of interest from each particle's history can be scored, or tallied, in phase space. Each history (i) results in a score (x_i). After N histories, there is a PDF of a set of scores $\{x_i\}$, where the mean is given by $\bar{x}$ (Equation (2.1)) and R is the associated relative error. The statistical relative

error R for each tally is defined by Equation (2.2) where $\sigma_{\bar{x}}$ is the standard deviation given in Equation (2.3). For well-behaved tallies, R is proportional to $1/\sqrt{N}$ [1]. This means that with an increasing number of histories, the relative error decreases.

$$\bar{x} = \frac{\sum x_i}{N} \tag{2.1}$$

$$R = \frac{\sigma_{\bar{x}}}{\bar{x}} \tag{2.2}$$

$$\begin{aligned} \sigma_x^2 &= \frac{\sum (x_i - \bar{x})^2}{N-1} \\ \sigma_{\bar{x}}^2 &= \frac{\sigma_x^2}{N} \end{aligned} \tag{2.3}$$

Another quantity of interest used to measure computational performance is the figure of merit (FOM), defined by Equation (2.4) where t_{proc} is the processor time required for the simulation [1]. It is desirable to have a high FOM, meaning there is low relative error and low processor time. FOM can be increased by decreasing R.

$$FOM = \frac{1}{R^2 t_{proc}} \tag{2.4}$$

One method for decreasing the relative error is to increase the number of histories. However, t_{proc} is proportional to N ($t_{proc} = C_t N$), so this will not necessarily increase the FOM alone. A more common practice is to reduce the constant of proportionality C_R between R^2 and $1/N$ ($R^2 = C_R/N$) through the use of VR methods [1].

2.1.1 Variance Reduction

Variance reduction (VR) methods aim to increase the number of particles whose scores are close to the tally mean (i.e. scores having high importance) in order to decrease the relative error, thus lowering the variance $\sigma_{\bar{x}}^2$ without necessarily decreasing the FOM for the same number of histories. Similar to the relative error, the processor time is proportional to the

number of histories. Equation (2.4) can be rewritten as a function of the proportionality constants shown in Equation (2.5). With most VR methods, the time per history will also increase, therefore increasing C_t. Effective VR methods will decrease C_R at a rate faster than the rate of increase of C_t

$$FOM = \frac{1}{C_R C_t} \tag{2.5}$$

In all VR methods, the idea is to preferentially sample and modify histories so that they become more representative of the tally mean. While there are many different classes of VR methods, one important to this work is known as population control [1]. In population control VR methods, the number of particles in phase space is controlled to increase the population in regions of interest (important regions) and decrease the population in regions of less importance. Particles' statistical weights are adjusted based on parameters defined over phase space, and therefore splitting and stochastic termination occurs based on changes in phase space [5].

In analog transport, the PDFs used for sampling are unbiased, meaning the PDFs represent the natural probabilities for events [1]. With VR methods, the PDFs can be biased to increase the influence the particles of interest by splitting them to pursue independent futures and stochastically terminating them to decrease the influence of particles of less interest. A particle that is deemed more important to the desired tally will be split into q particles, each now having a lower weight that is $1/q$th of their previous weight. Each particle produced from this splitting is considered statistically independent. For particles deemed unimportant, the particles are stochastically terminated, meaning they are terminated with some probability p. If a particle survives, it is given a higher weight. Particles of high importance have low statistical weight [1], but more of them exist as statistically independent histories contributing to the tally. This increase in the number of histories that contribute to the tally decreases C_R and the relative error, but does not bias the result of the tally because total weight is conserved. Because splitting causes more time to be spent on

tracking, C_t will increase. However, stochastic termination will lower C_t because less time is spent on unimportant particles. As previously stated, efficient VR methods will decrease C_R at at rate faster than the increase of C_t.

It is important to note that in Monte Carlo N-Particle transport code (MCNP) [1, 6], implicit capture, which is one type of VR, is turned on by default for neutrons, even in what we generally consider to be analog runs. With implicit capture, when a particle undergoes a collision, it is not terminated through absorption, but rather the statistical weight of the particle is reduced by the probability of absorption. Because implicit capture is turned on by default in MCNP [1, 6], all simulations in this work that do not use WWs (described in Section 2.1.2) will be referred to as analog from this point forward.

2.1.2 Weight Windows

A specific population control method for VR is the use of weight windows (WWs). WWs are defined by an upper and lower weight bound (w_U and w_L, respectively) and a survival weight (w_S, where $w_L < w_S < w_U$) for all of phase space. Typically, w_L is defined as a function of phase space, and w_U and w_S are defined as constant multiples of w_L ($w_U = C_U w_L$ and $w_S = C_S w_L$, where $C_U > C_S > 1$). In MCNP [1, 6], WWs can be defined spatially for each geometric cell or per voxel in a mesh spanning the region of interest. Typically, the spatial WWs are defined across discrete energy groups in the problem. At each collision, surface crossing, and mean free path traveled (in the case of using a WW mesh) in MCNP, the weight of the particle w is checked against the WW for the region of phase space where the particle is currently located [1, 6]. When particle weight w is checked, one of three things can happen to the particle:

1. If $w > w_U$, it splits into q particles each having a lower weight (w/q), where $q = w/w_U$ rounded up to the nearest integer.

2. If $w < w_L$, it is terminated with some probability p ($p = w/w_S$). If the particle

survives, it is given the higher survival weight w_S.

3. If $w_L < w < w_U$, it's left unchanged.

The work presented in this dissertation is based on the use of WW meshes, rather than geometric cell WW definitions, for generating WW isosurface geometry (WWIG)s. The Cartesian WW mesh (CWWM) can be generated in any method for this purpose. A few noteworthy methods for automatically generating CWWMs are briefly described here.

The Consistent Adjoint-Driven Importance Sampling (CADIS) method is a deterministic method to generate WW meshes automatically from the deterministic adjoint solution [7]. The CADIS method is implemented in the AutomateD VAriaNce reducTion Generator (ADVANTG) code [8], as well as in the Python for Nuclear Engineering (PyNE) toolkit [9, 10] that relies on the PARallel TIme-dependent SN (PARTISN) code [11] for the adjoint solution. Variations of the CADIS-based method include Forward-Weighted CADIS (FW-CADIS) [12], Multi-Step CADIS (MS-CADIS) [13], Groupwise-Transmutation CADIS (GT-CADIS) [14], and Angle-informed CADIS (CADIS-Ω) [15].

MCNP has a native method for generating WW meshes, known as the WW generator, in which the the importances are stochastically determined based on the history scores in each mesh voxel [1, 6]. The weights are then assigned to be inversely proportional to the importances. It is common to have to iterate on the generated WW meshes in this method, meaning that a new and better WW mesh can be generated by rerunning the simulation using the previously generated WW mesh.

Another WW mesh generation method is the Method of Automatic Generation of Importances by Calculation (MAGIC) [16]. This method is also an iterative stochastic method based on the initial flux distribution determined by a MC analog run. After the run, a mesh-based flux tally is normalized such that the highest flux value is 0.5. These new values are then used as the CWWM and can be used to generate a new flux tally in a subsequent run with better results. This is done iteratively until the desired level of fidelity in the results is achieved.

When using WW meshes, it is important to note that there is not a single correct WW mesh to use in each problem. WW meshes are meant to aid the simulation through VR, so some meshes may be better or more efficient than others. One must consider the trade-off between the effort to generate the mesh, the memory footprint, and the expected benefit when determining how to generate the WW mesh. Because each voxel of a CWWM contains information for each energy group, a with a mesh sized $N_x \times N_y \times N_z$ with G energy groups would require that $GN_xN_yN_z$ data points be stored. Depending on how the CWWM is generated, the number of energy groups G can range from one to more than 200. Additionally for very physically large and complex systems, the resolution for N_i must often be coarser than some of regions of the geometry. For example a $100\,\mathrm{m} \times 100\,\mathrm{m} \times 100\,\mathrm{m}$ transport model may not be able to have mesh voxels smaller than $1\,\mathrm{m}^3$ due to memory limitations, but in this same model there maybe variation in materials, and therefore variation in transport properties, much more frequently than every meter. A CWWM with a coarse mesh relative to the variation in transport properties can potentially lead to inadequate VR, which is an issue as the systems used in simulations used by analysts grow in both physical size and detail.

Ideally, an infinitely fine mesh would be used to capture all the fine detail in a geometrically complex problem. However, due to memory limitations, it is often not feasible to create extremely fine WW meshes and so some detail may be lost. Analysts must generate CWWMs that are fine enough to capture the detail in relevant parts of the system, yet coarse enough to not exceed computational resources. Take, for example, the simple transport geometry on the left side of Figure 2.1 where the yellow rectangles represent a highly attenuating material. In order to accurately capture the drastic change in WW values in each of the yellow rectangles, one might need to employ a very fine mesh across the entire domain as is shown on the right. However, in the regions outside the highly attenuating areas, the WW gradient is expected to be much less intense and so the fine mesh in those regions may be an unnecessary use of computational resources.

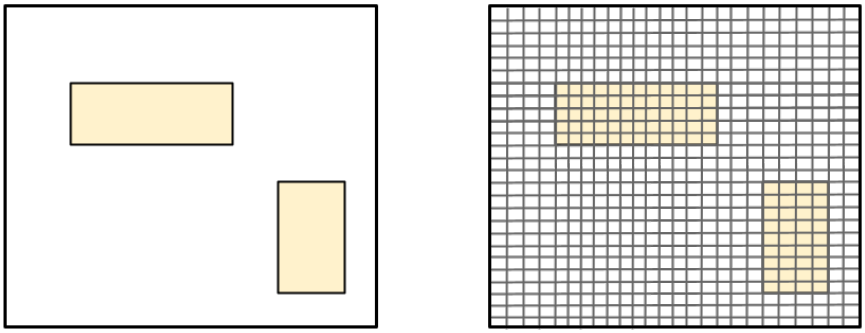


Figure 2.1: An example of a CWWM for the transport geometry (left) where the mesh (right) is very fine across the entire spatial domain.

Previous work by A. Ibrahim [17, 18] sought to alleviate the memory constraint of the WW meshes without losing fine detail in geometrically complex problems. The algorithm in this work strategically coarsens WW meshes in a way that does not compromise fine detail nor decrease the efficiency of the MC simulation. This is done by preserving the fine mesh resolution in regions of highly varying WWs and coarsening the mesh in regions with little WW variation. An example of how the CWWM for the transport geometry in Figure 2.1 is shown in Figure 2.2. By coarsening the mesh, the memory footprint of the mesh is decreased. These modified WW meshes were shown to be used in MC simulations with little to no decrease in the efficiency of the simulation. However, this coarsening method is not universally applicable to all geometries requiring high fidelity in only select regions of phase space. In Figure 2.2, we see that because the meshes are coarsened independently in each direction, there still may be regions of low weight gradient that have an unnecessarily fine mesh (upper right and lower left quadrants). One could imagine in a large complex system with many fine components throughout the entire domain, this coarsening may be much less effective.

The WW representation using WWIGs aims to address this by capturing fine detail in regions with high weight variation with the isosurfaces. Isosurfaces are not restricted to the Cartesian grid form of a CWWM but instead follow the arbitrarily complex forms of the changing WW values across the spatial domain. This means that the use of isosurfaces can

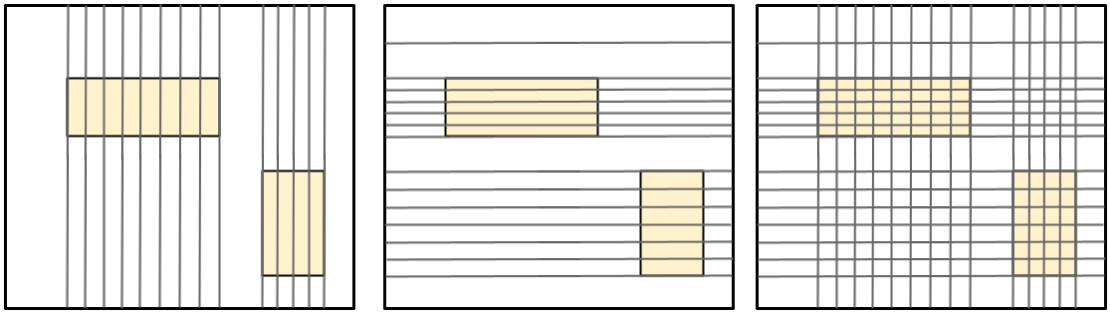


Figure 2.2: An example of a CWWM for the same transport geometry where the mesh is strategically coarsened in both directions (left and center).

capture the fine detail *only* in the regions of the high weight gradients. See Section 3.3 for more discussion.

2.2 Direct Accelerated Geometry Monte Carlo

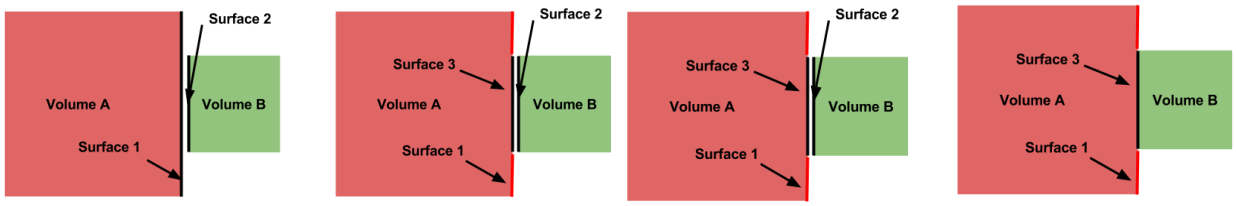
The Direct Accelerated Geometry Monte Carlo (DAGMC) toolkit [19] is a geometry toolkit that performs the necessary particle tracking steps during MC simulations directly on CAD-based tessellated geometries. DAGMC offers a method for ray tracing on arbitrarily complex geometries and higher order surfaces than what is typically allowed in native MC codes, thus eliminating the need to generate the geometry definition in the native MC format. These capabilities of the DAGMC toolkit are what enables the WWIG method presented in this dissertation. This chapter describes the process for creating DAGMC geometries and the DAGMC particle tracking algorithm.

2.2.1 Geometry Construction

A DAGMC geometry is a faceted (or tessellated) model of a CAD geometry represented using Mesh Oriented datABase (MOAB) [20]. The facets are stored in a hierarchical structure by MOAB so that each triangle facet belongs to a surface, and each surface belongs to at least one volume. When two volumes in the CAD model have coincident surfaces, the coincident region is redefined as a single surface belonging to each volume through operations known as

imprinting and *merging* [21, 22] prior to the faceting process. A more detailed description of the process for creating the faceted DAGMC geometries from a CAD model is as follows:

1. The user ensures that no volumes overlap or enclose others (no two volumes can occupy the same physical space).
2. Coincident surfaces are imprinted and merged (process shown in Figure 2.3).
3. Surfaces are faceted into triangular facets.
4. The facets are sealed to eliminate possible misalignment of vertices (also known as making the model *watertight* [23, 24]).



(a) Imprinting process *(b) Merging process*

Figure 2.3: Depiction of imprinting and merging two coincident surfaces (a gap is shown between surfaces 1 and 2 for clarity) into a single surface belonging to each volume A and B. (Images source: [21])

DAGMC requires that geometries used for MC transport meet the following requirements:

- All volumes are made of closed surfaces (there are no intentional spaces or gaps in the surfaces).
- No volumes occupy the same physical space (they do not overlap).
- Coincident surfaces between two adjacent volumes have been merged.
- Surfaces are faceted into triangles.
- The faceting of surfaces does not introduce numerical gaps (addressed with making the model watertight).

- Topological relationships are explicitly represented.

At the end of this geometry construction process, one should have a DAGMC geometry that meets these requirements and is suitable for robust MC particle transport.

2.2.2 Particle Tracking

The DAGMC toolkit can be coupled with various MC physics codes to perform the necessary particle tracking steps on the faceted geometry model, replacing the need for a text-based transport geometry definition native to the physics code. Currently, DAGMC has been coupled with MCNP 5 & 6 [1, 6], OpenMC [25, 26], Shift [27], Fluka [28], Geant4 [29], and Tripoli [30]. The exact implementation for each physics code can differ, but each makes use of the same basic principle of ray tracing to track particle position and movement through the geometry while the physics code continues to handle the underlying physics.

Ray tracing is when a ray is fired from the particle's current position in the direction being traveled to determine the next surface in the geometry that is intersected [19]. The surface that is intersected by the ray is determined by finding the facet that the ray intersects within an hierarchical tree of bounding boxes for the geometry [31]. DAGMC constructs sets of oriented bounding boxes (OBBs) around the facets that construct a volume (see Figure 2.4). The ray fired by DAGMC transverses the tree by checking if it intersects with each OBB at the first level. If it intersects with an OBB, it moves on to check the children OBBs (in the next level). This continues until it reaches the last level in the tree where the OBB contains only a few facets, which can then be used to calculate the exact point of intersection within a facet.

When the surface is determined based on the intersecting facet, the distance to the point of intersection and the volume on the other side of that surface are known. This information is then used in the physics code to determine the appropriate course of action for particle transport (e.g. transport to the surface and update properties for the next volume, perform collision physics, etc.) [19].

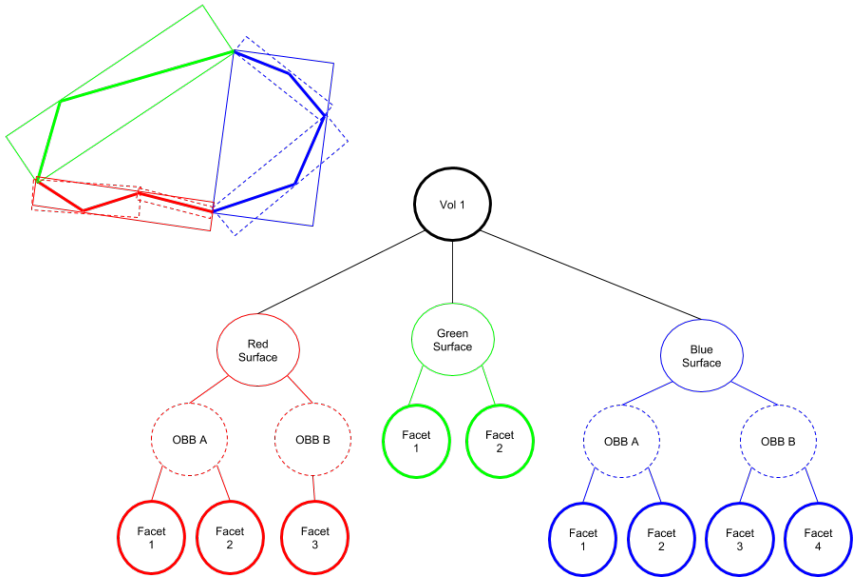


Figure 2.4: Each surface (red, green, and blue) of the volume (left) is made of facets (represented in 2D as thick lines). OBBs (depicted as thin solid and dashed lines) encase a set of facets in each surface. The hierarchical bounding box tree that is constructed for the volume is shown on the right. (Image source: [21])

Below is the current DAGMC particle tracking algorithm[1] (also presented visually in Figure 2.5) as it is implemented in MCNP 6.2 (known as DAGMC coupled with MCNP (DAG-MCNP)) [1, 6, 19] when WW meshes are used for VR:

1. Calculate the mean free path distance d_w using MCNP.

2. Sample the distance to collision d_c along the particle's trajectory with MCNP.

3. Find the distance to the next surface on the DAGMC geometry d_s along $\vec{r}$ using DAGMC's ray tracing.

4. The minimum distance D ($D = min(d_w, d_s, d_c)$) indicates the next event:

   a) if $D = d_w$: Transport particle distance d_w.

   b) if $D = d_s$: Transport particle to the next transport geometry surface using DAGMC.

[1]This algorithm is presented in a simplified manner to not include the use of DXTRAN spheres, time cutoffs, and energy cutoffs. Forced collisions have also been ignored.

c) if $D = d_c$: Transport particle distance d_c to collision site and perform appropriate collision physics using MCNP.

5. Look up WW for the current location in the WW mesh and check particle weight (splitting or terminating as appropriate). Restart the particle tracking for the updated surviving particle(s).

6. Repeat from the first step until all histories have be terminated in some capacity.

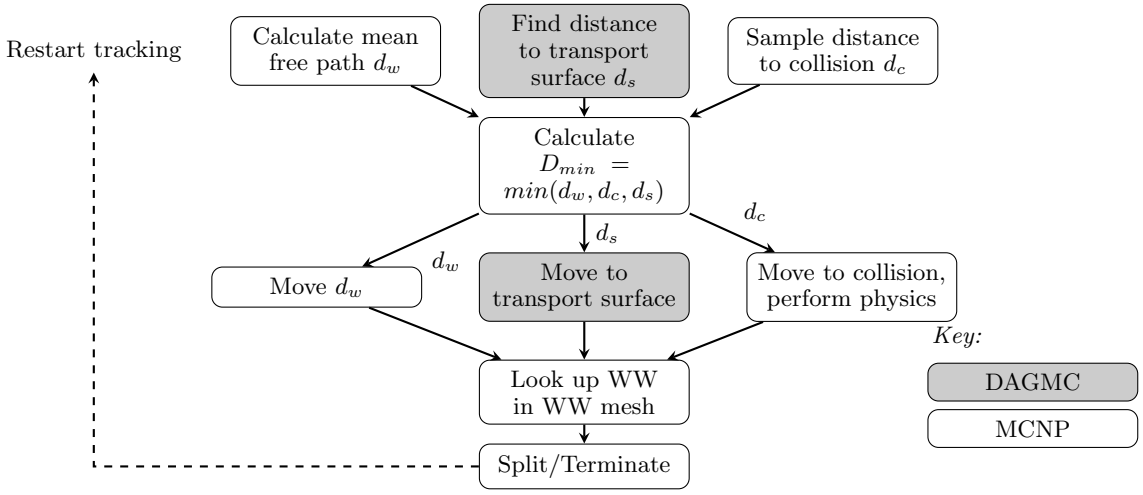


Figure 2.5: Visual depiction of the particle tracking algorithm when using DAG-MCNP and a CWWM for VR.

2.2.3 Performance

In the past, CAD-based ray-tracing for MC simulations had been significantly more time consuming than native MC simulations. There has been a trade off between having the capability to perform simulations on geometries with higher order surfaces and the computational time required to perform the particle tracking in MC codes. However, recent developments to ray tracing accelerations in DAGMC have shown that, especially for complex geometries, DAG-MCNP can be competitive with native MCNP in simulation run time [32, 33, 34]. This performance increase in DAGMC enables for the first time the capability for particle tracking on multiple complex faceted geometries as is presented in this dissertation.

Chapter 3

Weight Window Isosurface Geometries

WW isosurface geometries (WWIGs) are DAGMC-compliant faceted geometries where the surfaces of the volumes represent isosurfaces of the lower WW bounds in a CWWM. They can then be used in place of CWWMs during MC particle transport with DAG-MCNP 6.2. This chapter presents the detailed generation method, the particle tracking algorithm, and an analysis of the different geometric features of WWIGs.

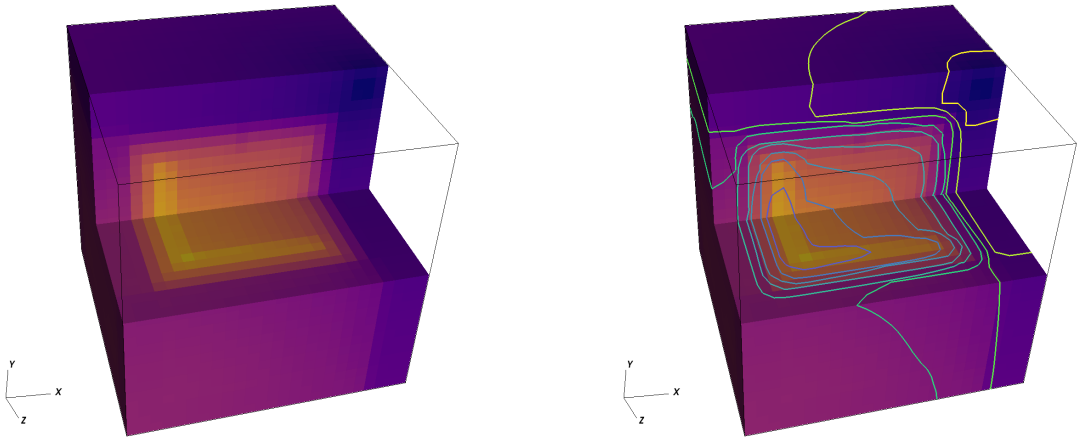
3.1 Generation Method

The WWIGs are generated from isosurfaces derived from the CWWMs. These CWWMs can be generated in a variety of ways, as described in Section 2.1.2. To generate WWIGs from the CWWM, VisIt (a visualization and data analysis tool) [35] and MOAB [20] are used in an automated Python tool called IsogeomGenerator [36] that requires only initial information about which WW values to use to generate the isosurfaces. This process generates a DAGMC-compliant geometry with each volume corresponding to the region of space between two consecutive isosurface values. A description of this process is as follows [37]:

1. *Select WW isosurface values:* The user specifies the WW values to use for the isosurface values (see Section 3.1.1 for additional details) and an optional normalization factor.[1]

[1]When using a CWWM with MCNP, one has the option to supply a normalization factor as input and every WW lower bound in the mesh is then multiplied by that factor [1, 6]. In the case of WWIGs, the normalization factor is applied during generation by multiplying it by the user-specified isosurface level

An example visual of such values are shown in Figure 3.1 where the isosurface contours are seen overlaid on the CWWM.



(a) An example 3-D CWWM. *(b) A CWWM with selected level values shown.*

Figure 3.1: An example CWWM with WW isosurface values shown that will be used to generate the WWIG surfaces. A cut-out shows the interior of the mesh.

2. *Export isovolume surface:* VisIt is used to generate a closed isovolume defined as the space between two consecutive WW values and, in some cases, the boundary of the CWWM domain where the isosurfaces intersect the boundary. The bounding surfaces of these isovolumes get exported as an STL file, a format in which the mesh is defined by triangular facets, the connectivity of all vertices is known, and all surfaces are closed. An example of a single surface mesh that might be generated in this process is shown in Figure 3.2.

3. *Volume separation:* Each isosurface from the previous step has the potential to be a isosurface that is actually a collection of disjoint closed surfaces (example shown in Figure 3.3). MOAB is used to collect all sets of connected vertices, and the triangles they form, of each isosurface and then redefines each disjoint set as a new surface belonging to the same volume.

values. This is important for CWWMs generated using ADVANTG which returns a normalization factor [8] that must be applied to the surface values.

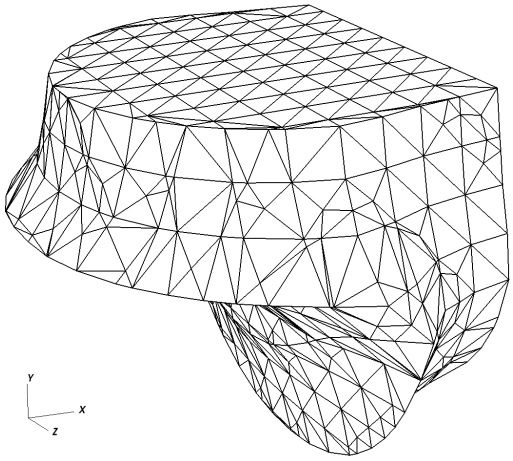


Figure 3.2: Example surface mesh generated by VisIt of the volume between two isosurfaces.

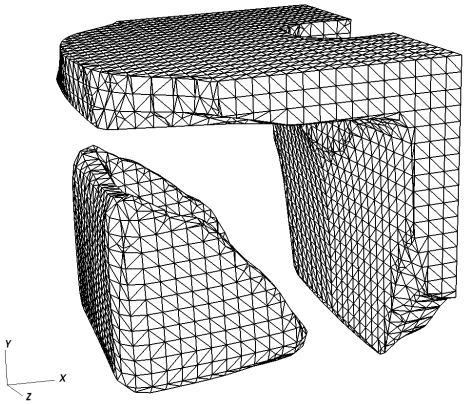


Figure 3.3: A single isosurface volume that created multiple closed surfaces that need to be redefined as separate surfaces.

4. *Surface separation:* To further ease the mesh merging process, these surfaces are further divided into separate, but connected, sets of triangles and vertices that are considered "interior" or "exterior" to the geometry. Sets of triangles whose centroids are coincident with the external bounding surfaces of the original CWWM are considered to be on the exterior, while all other triangles are on the interior. Only surfaces made of interior triangles need to be considered in the merging process in the following step. An example of a surface that would be further separated into interior and exterior surfaces is shown in Figure 3.4.

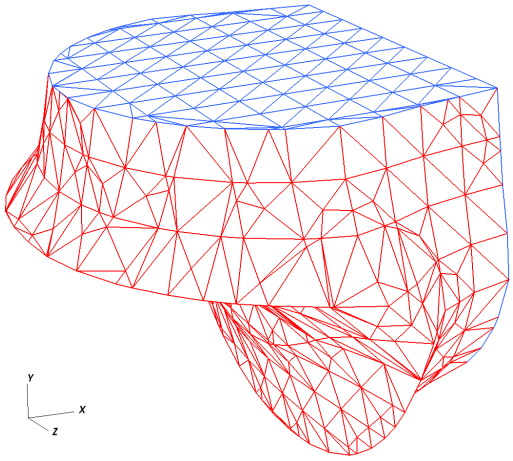


Figure 3.4: An example of an isosurface that has a portion of its surface coincident with the bounding surfaces. The blue surface indicates where the mesh is coincident with the external bounding surfaces and would be labeled as "exterior" while the red portion is considered "interior."

5. *Mesh-based merging:* Unlike the DAGMC geometry generation process for a transport geometry described in Section 2.2.1, the imprint and merging step in this process occurs *after* after the surfaces have been meshed into triangles, which adds a layer of complexity. To do this, MOAB is used to identify vertices in one isosurface that are coincident with other vertices in other isosurfaces. As soon as the position of one vertex is found to match that of a vertex in another surface, those two surfaces are considered to be coincident and one replaces the other. Due to the nature of isosurfaces and the knowledge that only surfaces on the interior of the geometry will be merged, only the vertices of the interior surfaces of two isovolumes that share a WW value need to be checked against each other. An example of two consecutive isovolumes that share the same interior surface is shown in Figure 3.5. This step relies on consistency from VisIt in the interpolation process for the creation of the original isosurfaces such that vertices do exactly align between two adjacent volumes. In some cases, vertices that are not perfectly aligned can cause issues in this step. In general, this is not an issue but it does pose a limitation on how close isosurfaces can be to each other as described in Section 3.1.1.

6. *Export DAGMC geometry:* The WW value used to make each of the surfaces are then

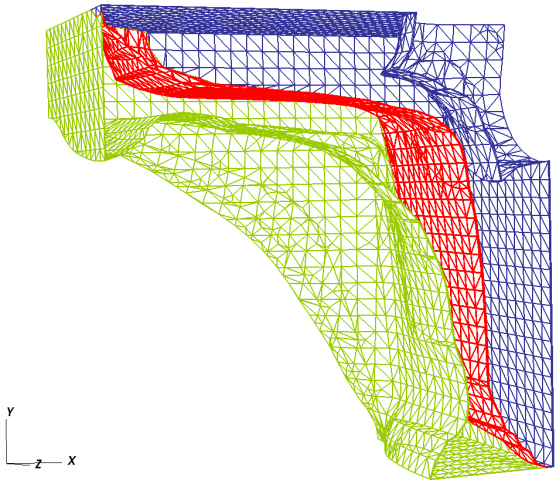


Figure 3.5: Two adjacent isosurface volumes (green and blue) that have a coincident internal surface (shown as red) that is in need of mesh-based merging. The red surface defined on each separate volume is redefined as the single red surface. A cut out shows the internal surface.

set as data on the surface and then the geometry is exported as a complete DAGMC-compliant geometry.

7. *Repeat:* The process is repeated for each energy group in the initial CWWM file. Figure 3.6 shows example WWIGs produced for multiple energy groups of one CWWM file.

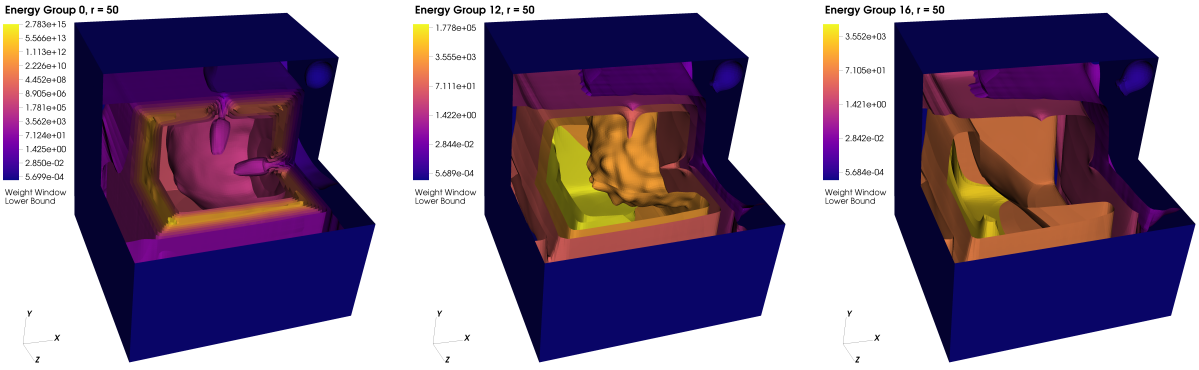


Figure 3.6: Example WWIGs produced for a select set of energy groups from a multi-energy group CWWM file.

After the generation of the WWIG(s), the user is left with multiple geometries that

occupy the same physical space: a transport geometry where the different volumes and surfaces represent different materials or physical components; and at least one WWIG where the volumes and surfaces represent different WW values. The WWIGs generated by this method can then be used for particle tracking with DAGMC (described in Section 3.2), as they satisfy the geometry requirements specified in Section 2.2.1.

3.1.1 Selection of Isosurface Values

The automatic generation process presented in the previous section requires the user to supply information about which values to use for the isosurfaces, which can be done in a number of ways. Below describes three specific methods for defining the isosurface values that have been implemented in the IsogeomGenerator tool [36]:

- User-specified values $\{S_0, S_1, S_2, \ldots, S_N\}$.
- User-specified number of surfaces N to be linearly or logarithmically evenly spaced between the minimum and maximum w_L values for an energy group.
- Surfaces are separated by a user-specified ratio r as defined by Equation (3.1).

$$\begin{aligned} S_0 &= min\,(w_L) \\ S_1 &= S_0 r \\ S_2 &= S_1 r = S_0 r^2 \\ &\cdots \\ S_i &= S_0 r^i \end{aligned} \tag{3.1}$$

Of the three methods, the selection of surfaces based on a separation ratio r is expected to be the most widely used because r can be related to the WW upper bound constant C_u used in transport. Because the particles only undergo splitting when crossing a WWIG surface, r can be chosen such that this splitting efficiency is optimized. It is expected that the most efficient spacing is $r = C_u$ because the number of splits at each crossing are expected to

then be equal to C_u. If the ratio spacing is higher than C_u, not enough splitting may occur, meaning there may not be adequate VR. In the case that the ratio spacing is lower, the efficiency may be decreased because too many particle weight checks are occurring when no or little splitting is occurring while the amount of time spent on surface ray tracing increases.

While the commonly used value for C_u is 5 (default in MCNP [1, 6]), there may be limitations as to how low r can be due to the gradient of WW values in the supplied CWWM, making $r = 5$ not necessarily feasible. When the CWWM has regions of very high gradients, such as in shielding regions, the isosurfaces generated by VisIt [35] can be very close together in physical space, if not overlapping, causing complications in the WWIG generation process. Because the generation method relies on both VisIt to interpolate across mesh voxels in a consistent manner between adjacent isovolumes and the gradients of values present in the CWWM, we can qualitatively derive a minimum supported ratio r_{min} as a function of the gradients and mesh voxel size. The derivation of r_{min} is as follows.

We know by definition, that the ratio of WW lower bound values w_L between two adjacent mesh voxels $\vec{x}$ and $\vec{x} + \Delta\vec{x}$ is given by Equation (3.2), where $\Delta\vec{x}$ is the vector between adjacent mesh voxels. In the case of a CWWM, $\Delta\vec{x}$ is defined by the edge lengths of the local mesh voxel in Equation (3.3) and whose magnitude is the length of the voxel diagonal, i.e. largest distance between known points of the mesh voxel. Ratios, also by definition, can be translated to a logarithmic scale such that $r = 10^a$, allowing us to write Equation (3.4). By taking the logarithm of Equation (3.4), we get Equation (3.5). Dividing both sides of Equation (3.5) by $\Delta\vec{x}$ (Equation (3.6)) transforms the right hand side into the definition of the gradient, allowing a to be more simply defined by Equation (3.7). The minimum supported ratio value r_{min} is therefore defined by the maximum value of a in the CWWM, given by Equation (3.8).

$$r = \left| \frac{w_L\left(\vec{x} + \Delta\vec{x}\right)}{w_L\left(\vec{x}\right)} \right| \tag{3.2}$$

$$\Delta\vec{x} = e_x\hat{x} + e_y\hat{y} + e_z\hat{z} \tag{3.3}$$

$$r = 10^a = \left| \frac{w_L\left(\vec{x} + \Delta\vec{x}\right)}{w_L\left(\vec{x}\right)} \right| \tag{3.4}$$

$$a = \left| \log\left(\frac{w_L\left(\vec{x} + \Delta\vec{x}\right)}{w_L\left(\vec{x}\right)} \right) \right| = \left| \log\left(w_L\left(\vec{x} + \Delta\vec{x}\right)\right) - \log\left(w_L\left(\vec{x}\right)\right) \right| \tag{3.5}$$

$$\frac{a}{|\Delta\vec{x}|} = \left| \frac{\log\left(w_L\left(\vec{x} + \Delta\vec{x}\right)\right) - \log\left(w_L\left(\vec{x}\right)\right)}{\Delta\vec{x}} \right| \tag{3.6}$$

$$a = \left| \nabla \log\left(w_L\left(\vec{x}\right)\right) \cdot \Delta\vec{x} \right| \tag{3.7}$$

$$\begin{aligned} r_{min} &= 10^a \\ a &= \max\left(\left| \nabla \log\left(w_L\left(\vec{x}\right)\right) \cdot \Delta\vec{x} \right| \right) \end{aligned} \tag{3.8}$$

3.2 Particle Tracking

This section describes the algorithm for using WWIGs during DAG-MCNP 6.2 [19, 6] transport in place of CWWMs. It is important to note that the development and assessment of the WWIG method in this work is specifically in the context of using MCNP. Other MC codes may implement the use of WWs and CWWMs in a different manner, making it possible that the WWIG method is less applicable. Previous work by E. Gonzalez and G. Davidson [38] compared various implementations of CWWMs and how they affect performance with Shift [27]. They found that a finer CWWM always performs better than a coarse mesh and that applying WWs pre-collision, as opposed to post-collision, also improves performance.

In this new method, particles are tracked simultaneously on both the transport model and the WWIGs. Tracking on the WWIGs uses the same ray tracing method already used for particle tracking on DAGMC transport models. When a particle crosses one of the WW isosurfaces, the particle weights are checked against the WWs defined for that surface. At

this point in time, splitting or stochastic termination can occur. The detailed algorithm[2] as it is implemented for DAG-MCNP 6.2 is as follows and is depicted visually in Figure 3.7 [39]:

1. Find the distance to the next surface along $\vec{r}$ on the WWIG (d_w) for the current energy group using DAGMC's ray-fire.

2. Find the distance to the next surface along $\vec{r}$ on the DAGMC transport geometry (d_s) using DAGMC's ray-fire.

3. Sample the distance to collision (d_c) along trajectory $\vec{r}$ using MCNP.

4. The minimum distance D ($D = min(d_w, d_s, d_c)$) indicates the next event:

    a) if $D = d_w$: Transport particle to the next surface on the WWIG, **update the particle weights (splitting or terminating as appropriate) according to the WW defined on that surface**, and restart particle tracking for the updated particle(s).

    b) if $D = d_s$: Transport particle to the next surface on the transport geometry and continue particle tracking for the particle in the new cell.

    c) if $D = d_c$: Transport particle d_c to collision site and perform appropriate collision physics. Restart particle tracking if particle is surviving.

It is possible to use WWIGs in combination with a native MCNP transport geometry rather, than a DAGMC transport geometry. In this case, the algorithm is identical to the one presented above, except that steps 2 and 4(b) would use MCNP for particle tracking on the transport geometry while DAGMC is still used for the WWIGs.

The difference between this new algorithm and the original DAG-MCNP algorithm described in Section 2.2.2 is that the particle weight is checked and updated now only at a

[2]This algorithm is presented in a simplified manner to not include the use of DXTRAN spheres, time cutoffs, and energy cutoffs. Forced collisions have been ignored.

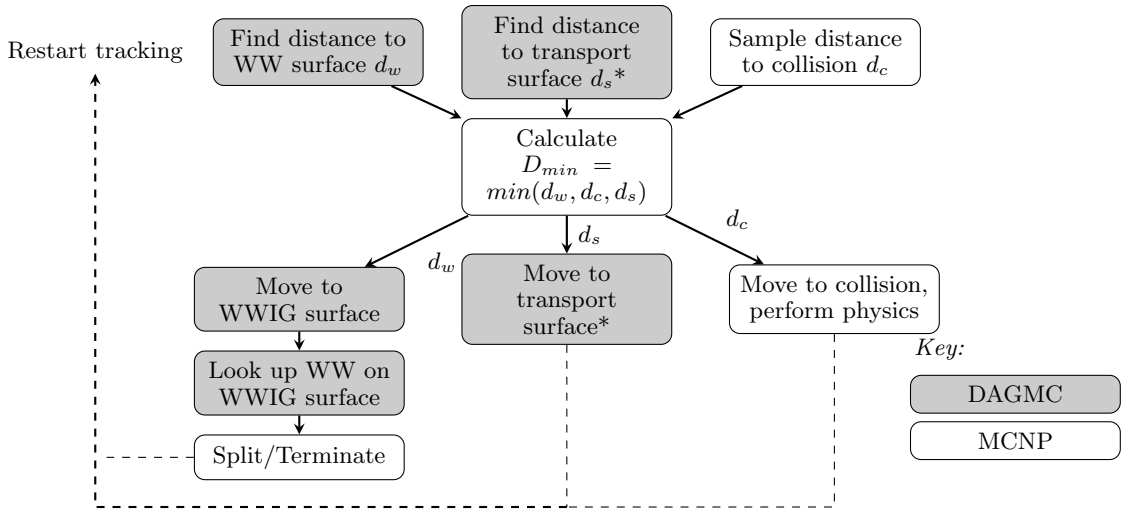


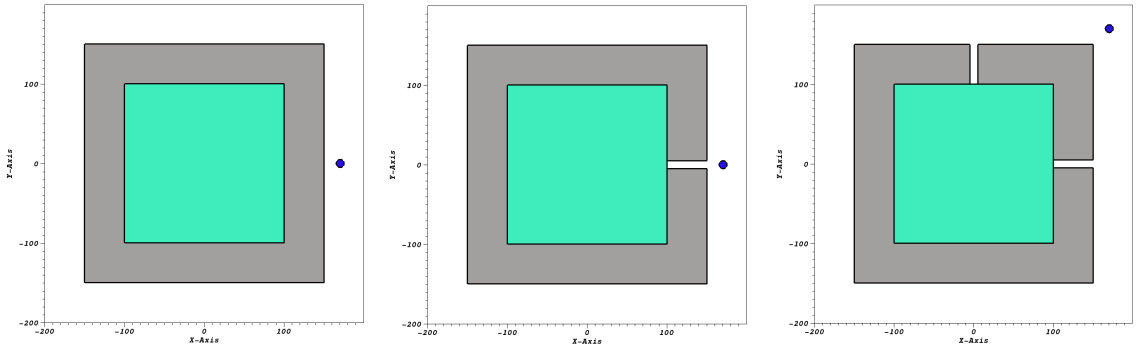
Figure 3.7: Visual depiction of the particle tracking algorithm when using DAG-MCNP and WWIGs for VR. An asterisk (*) indicates which steps would use MCNP, rather than DAGMC, in the case of using a native MCNP transport geometry.

WWIG surface crossing, rather than every mean free path, collision, and transport geometry surface crossing. The method for looking up the WW differs now as well to make use of the WWIGs, rather than a WW mesh, by looking up the WW on the WWIG surface rather than in the WW mesh. The rest of the particle tracking process and physics remains the same.

3.3 Geometric Features

This section will show some sample WWIGs to highlight some of the geometric mesh features. Three simple problems were developed to demonstrate geometric features that are present in the WWIGs. Each transport geometry is a variation of a point detector problem with a 14 MeV volumetric neutron source (coincident with a $2\,\text{m} \times 2\,\text{m} \times 2\,\text{m}$ helium box) surrounded by a stainless steel wall (SS316) 0.5 m thick (see Figure 3.8). A point detector is located on the outside of the box in all cases. For transport geometries 2 and 3, narrow streaming channels were introduced in the box (Figure 3.8b and Figure 3.8c).

For each problem, CWWMs were generated using both MCNP's WW generator (a single



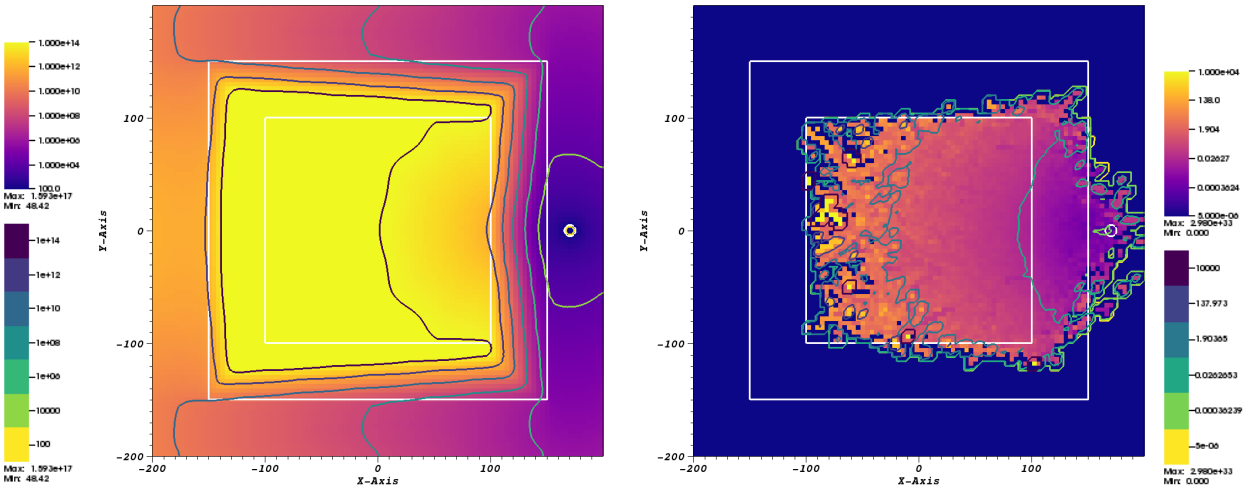
(a) Transport Geometry 1 *(b) Transport Geometry 2* *(c) Transport Geometry 3*

Figure 3.8: 2-D cross sections at $z = 0$ of the different transport geometries used: 14 MeV neutron source (green) surrounded by a 0.5 m stainless steel box (gray) with a detector outside (blue).

iteration) [1, 6] and using the CADIS method [7] in ADVANTG [8]. MCNP's native generator is a stochastic method while CADIS is deterministic. Slices of a single energy group of the generated CWWMs overlaid with the isosurfaces can be seen for each transport geometry in Figure 3.9. For each CWWM generated by CADIS, seven isosurfaces levels were chosen to be logarithmically spaced from 10^2 to 10^{14}. For each of the CWWMs generated by MCNP, six logarithmically spaced levels ranging from 5×10^{-6} to 10^4 were used. The 3-D generated WWIGs for a single energy group for the third geometry (Figure 3.8c) can be seen in Figure 3.10. These example WWIGs exhibit a number of geometric and mesh features that may impact performance when used in particle transport. The following sections will describe each feature in more detail.

3.3.1 Isosurface Spacing

For all transport geometries with streaming paths (Figure 3.8b and Figure 3.8c), the CADIS-generated geometries had the ability to capture the fine detail of the WWs in the streaming paths and the weight gradients in highly attenuating regions better than MCNP's WW generator, resulting in more WW isosurfaces in these regions. This is evident in Figure 3.9c and Figure 3.9e (CADIS results) where there is a much higher number of isosurfaces present in the steel wall and around the streaming paths compared to Figure 3.9d and Figure 3.9f

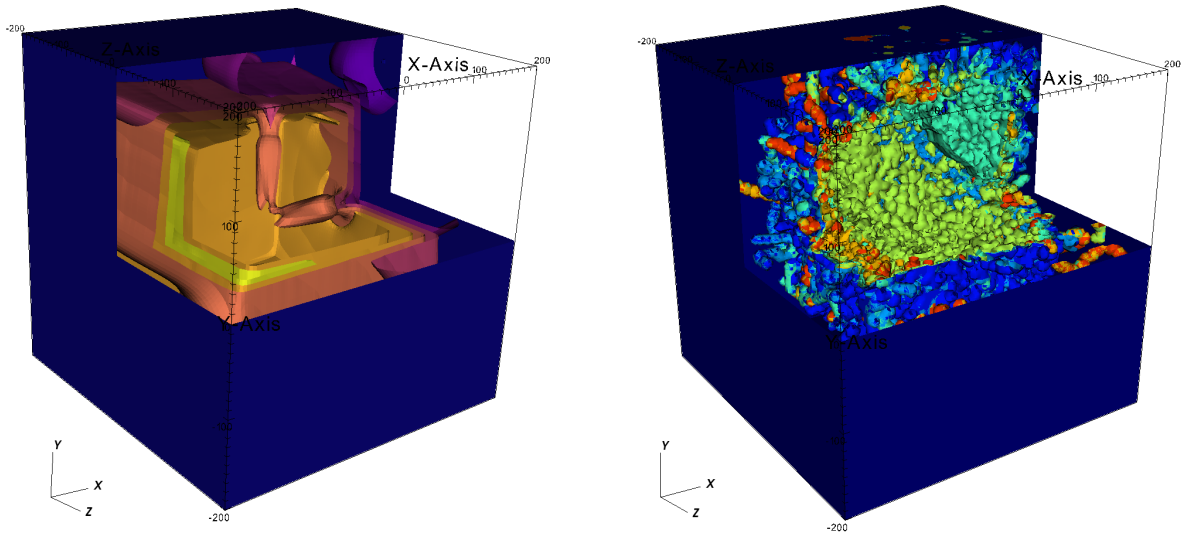


(a) Transport Geometry 1, CADIS *(b) Transport Geometry 1, MCNP*

(c) Transport Geometry 2, CADIS *(d) Transport Geometry 2, MCNP*

(e) Transport Geometry 3, CADIS *(f) Transport Geometry 3, MCNP*

Figure 3.9: 2-D cross sections at $z = 0$ for the generated WW isosurfaces superimposed on the original CWWM for each model. An outline of the transport geometry is shown in white.



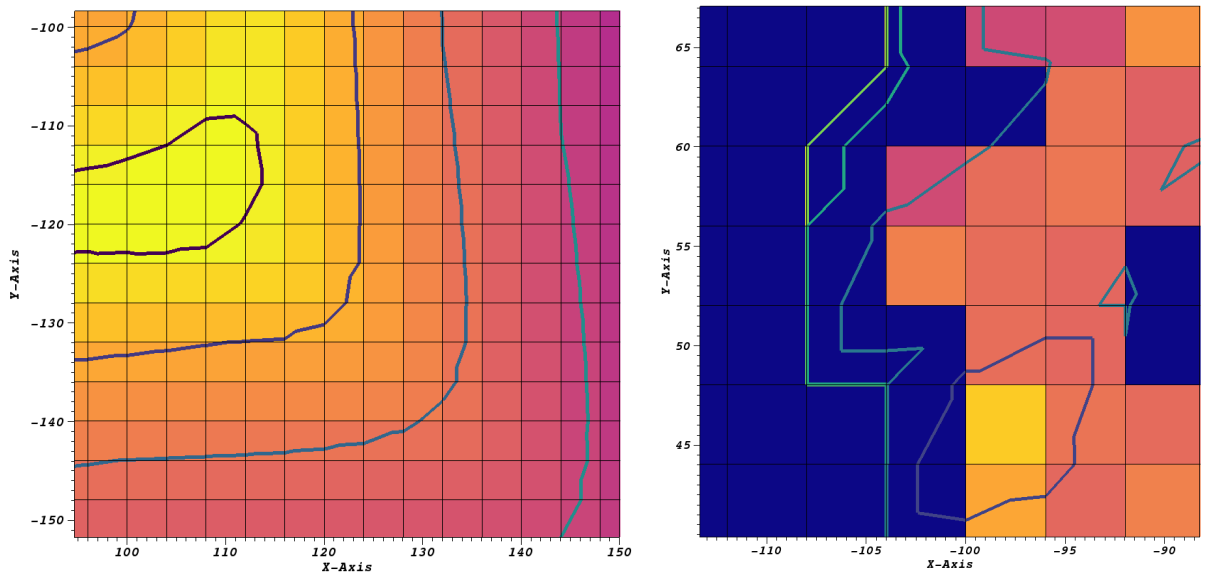
(a) CADIS *(b) MCNP*

Figure 3.10: Example 3-D models of the WWIG generated for transport geometry 3.

(MCNP results).

3.3.2 Weight Gradient Continuity

Due to the nature of how CADIS and the MCNP WW generator operate, there is significant difference in the continuity of the WW gradients in the initial CWWMs that can have significant impact on the quality of the resulting WWIGs. As seen in a close up of the original CWWM from CADIS in Figure 3.11a, adjacent mesh voxels in the original CWWM have a relatively smooth gradient of weight values making it easier to select isosurface values that are close enough to capture detail, but separated more than a single mesh voxel. This allows for the selected WW isosurfaces to be non-overlapping as described by the limitations of Equation (3.8) in Section 3.1.1. Conversely, in the close up of the CWWM produced by MCNP shown in Figure 3.11b, the stochastic nature of the CWWM production causes adjacent voxels to have very abrupt differences in weight values. This causes many isosurfaces to occupy the same space between the adjacent mesh voxels.



(a) CADIS *(b) MCNP*

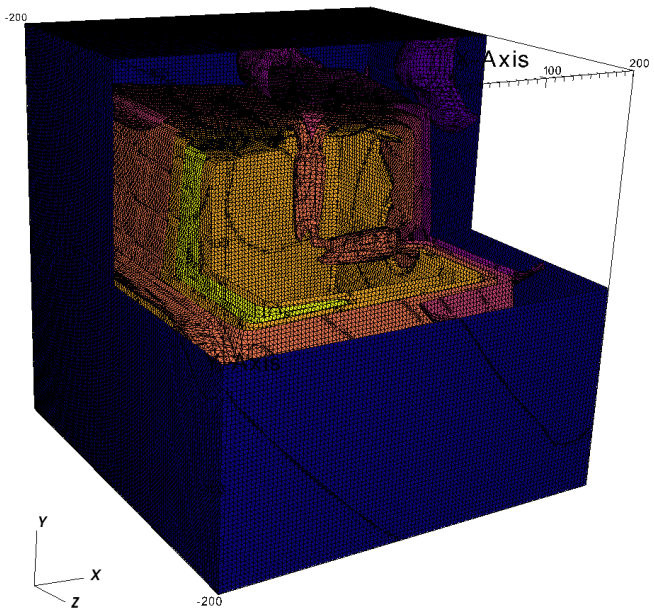
Figure 3.11: A close up of the WW values in the mesh voxels with the resulting isosurfaces superimposed.

3.3.3 Isosurface Roughness

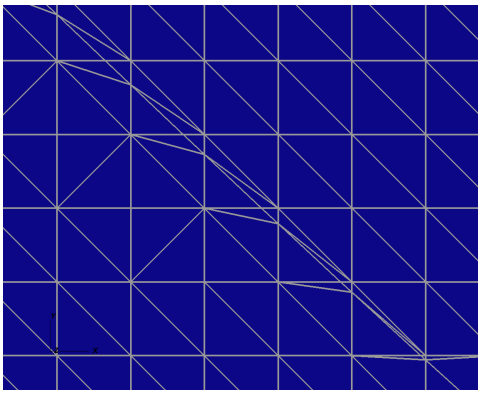
Another difference between the CADIS and MCNP WWIGs is the roughness of the resulting isosurfaces. In the case of the CADIS-generated WWIGs (Figure 3.10a), the surfaces are all relatively smooth, where as the stochastic nature of the MCNP-generated WWIGs create very rough, noisy surfaces (Figure 3.10b). Isosurfaces like those in the CADIS WWIGs are much more desirable for particle tracking and will likely create fewer unnecessary weight checks (see Figure 5.1 in Section 5.2). In the case of the MCNP-generated WWIG, a particle traveling in a straight line (with no weight change due to any other events such as collisions) may cross an isosurface many times due to the roughness, causing many unnecessary weight checks that do not result in weight changes. Rough isosurfaces that are likely to occur with any stochastic CWWM generation method can be addressed through surface smoothing described in Section 5.2.

3.3.4 Isosurface Coarseness

Because the surfaces of the WWIGs are generated from the original CWWM, which has fairly fine resolution, the resulting isosurfaces are also made of very fine triangular facets. Each face of one mesh voxel is split into at least two triangular facets by VisIt during the geometry generation process. An overlay of the surface meshes on the WWIG can be seen in Figure 3.12a which shows the presence of these small facets throughout the entire geometry. Additionally, where two isosurfaces meet on the outside of the geometry, even finer triangles are formed along the seams (a close up is seen in Figure 3.12b).



(a) WWIG with faceted triangular mesh made visible.



(b) Close view of facets where two isosurfaces meet.

Figure 3.12: A view of the triangular facets that make up the surface meshes where two isosurfaces meet on the outside of the WWIG.

This fine mesh resolution of the isosurfaces can potentially cause slowdown with DAGMC particle tracking because it will take longer to transverse the constructed OBB tree when determining the point of intersection. Additionally, the fine resolution in the meshes can unnecessarily increase the memory footprint of the WWIG. However, both of these concerns

can be addressed by applying decimation described in Section 5.1.

3.4 Summary of Weight Window Representation Differences

By using the WWIG method, WWs are represented in a fundamentally different way than the traditional CWWMs. With the CWWMs, all WWs are defined per mesh voxel in the mesh, meaning these values are defined on a per volume basis. With the WWIG, WW values are instead only defined on the surfaces of the volumes, rather than the volumes themselves. With CWWMs, fine resolution of the mesh is generally required across the whole spatial domain to capture the fine detail where it matters, whereas WWIGs only require fine resolution at the isosurfaces. When used with DAG-MCNP, particle weight is checked against the CWWM at points in time that are generally irrelevant to how the underlying WW values are changing in the mesh. In the case of the WWIGs, particle weight is checked at a WWIG surface crossing, and therefore corresponds to the changing WW values in phase space.

Chapter 4

Verification and Demonstration of WWIG Particle Tracking

This chapter demonstrates the WWIG tracking algorithm in two capacities. The first is verification of the method to ensure proper implementation of the tracking algorithm and the second is a proof of concept.

4.1 Verification

The purpose of this verification experiment is to test that the particle tracking method was implemented correctly. To do this, checks were added to the DAG-MCNP code to confirm the correct application of the WWs. Every WW check in a simulation should only occur at a surface crossing of the WWIG corresponding to the energy group for the particle's current energy.

4.1.1 Problem Setup

Two near identical transport geometries sized $20\,\text{cm} \times 5\,\text{cm} \times 5\,\text{cm}$ were used to verify this. Both use a mono-directional $\vec{v} = (1, 0, 0)$ neutron source at $x = 0.5\,\text{cm}$ with a varying source energy evenly distributed from $0.001\,\text{MeV}$ to $1.5\,\text{MeV}$ (see Figure 4.1). A point detector is located at $x = 19.5\,\text{cm}$. The first geometry is left as a void, simply to test that the WWIG corresponding to the energy group for the initial energy of the particle is chosen correctly. The second is filled with a lightly scattering media (water with density $0.02\,\text{g/cm}^3$) to check

that the WWIG is correctly updated for the change in energy groups upon collisions, but that it does not apply a particle weight check. Every simulation result presented here used 10^5 histories.

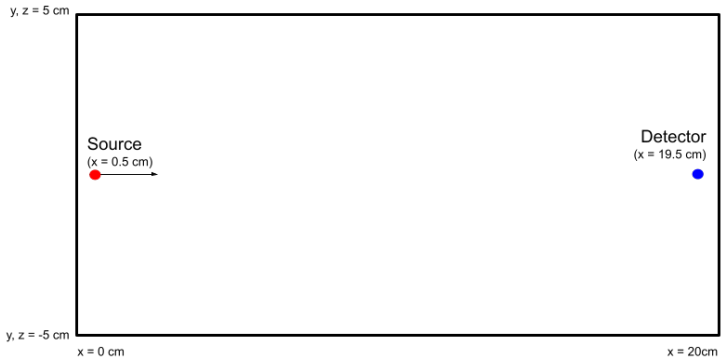
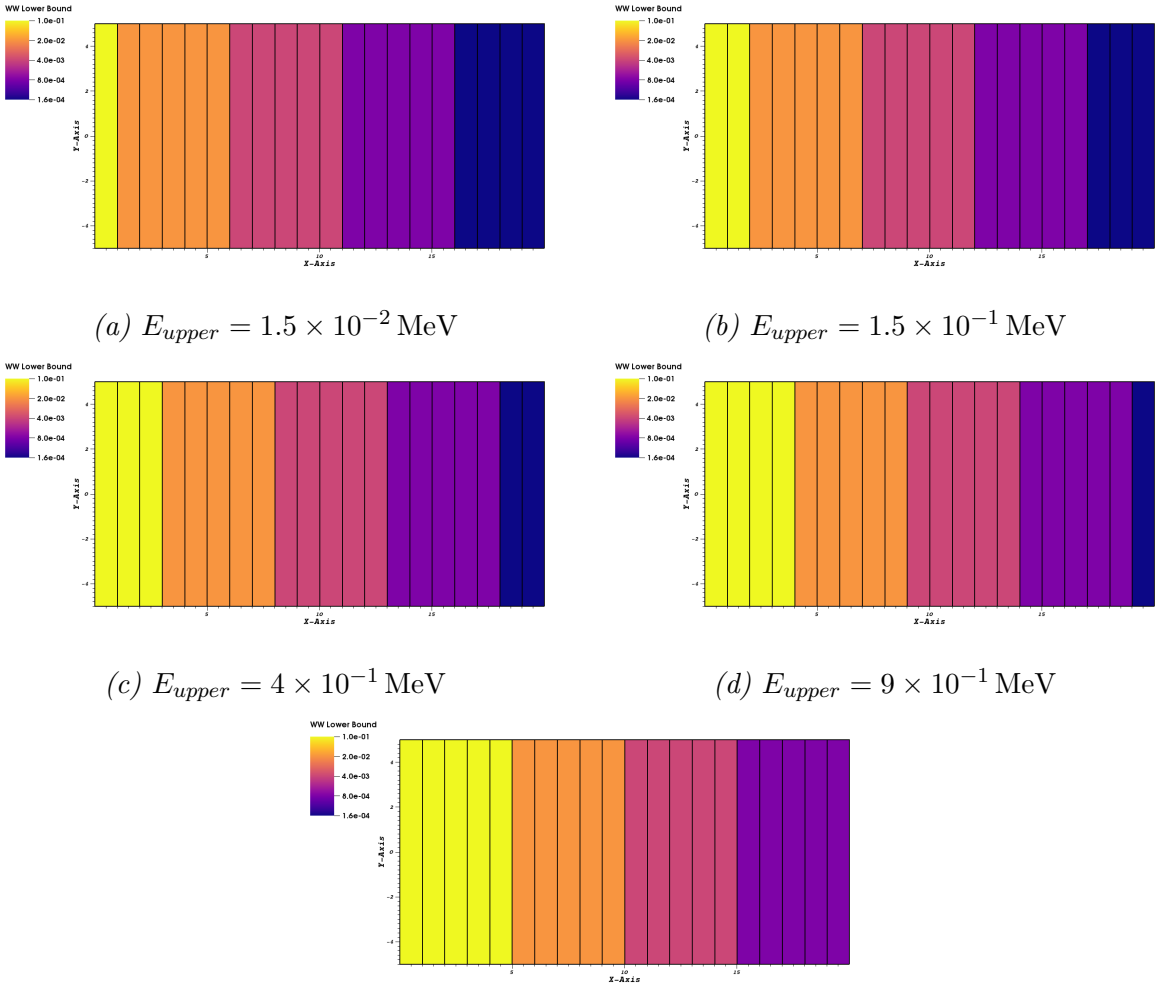


Figure 4.1: Transport geometry used for the verification experiment

For each of the above transport geometries, the same set of WWs were used. The energy groups for the WWs have upper bounds $E_{upper} = 1.5\times10^{-2}$, 1.5×10^{-1}, 4×10^{-1}, 9×10^{-1} and 1.5 MeV. The CWWM was designed such that there is one mesh voxel every centimeter in the x-direction (for a mesh resolution of $20 \times 1 \times 1$ mesh voxels). The WW values used on the CWWM were 1×10^{-1}, 2×10^{-2}, 4×10^{-3}, 8×10^{-4} and 1.6×10^{-4}. The same set of WW values were used for each energy group but offset in the x-direction by one mesh voxel with each increasing energy group (see Figure 4.2). This offset was done so that the resulting isosurfaces in the WWIGs would not be in the same physical location for each energy group. The WWIGs for each energy group were then created using the same previously stated WW values for each of the isosurface values. The resulting WWIGs can be seen in Figure 4.3.

4.1.2 Results and Analysis

In order to check that the particle weights are updated accordingly with each WWIG surface crossing and that the particle is tracked on the correct WWIG given its energy, the code was instrumented to collect data to be analyzed in post-processing. For every event, the particle must have the correct weight before and after the event. Collisions and transport



(a) $E_{upper} = 1.5 \times 10^{-2}\,\text{MeV}$

(b) $E_{upper} = 1.5 \times 10^{-1}\,\text{MeV}$

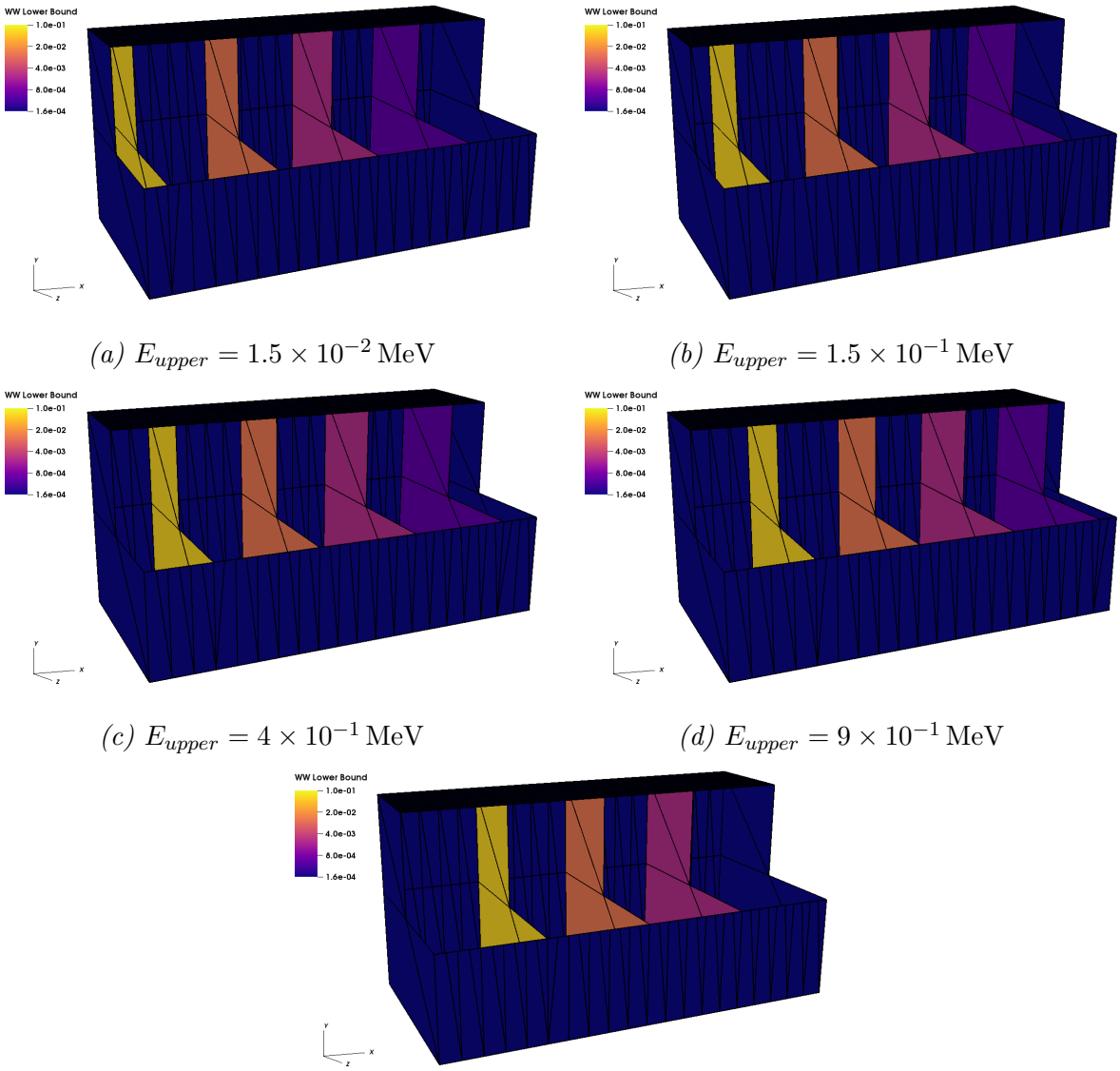
(c) $E_{upper} = 4 \times 10^{-1}\,\text{MeV}$

(d) $E_{upper} = 9 \times 10^{-1}\,\text{MeV}$

(e) $E_{upper} = 1.5\,\text{MeV}$

Figure 4.2: A 2-D slice at $z = 0$ of the WWs for each energy group in the CWWM.

surface crossings must not invoke a weight change through WWs, though collisions may still cause weight changes due to implicit capture. A collision can also prompt a change in energy group and therefore a change of WWIG. At WWIG surface crossings, a particle can split or stochastically terminate. To specifically check that the correct WWIGs are being used according to the particle's energy, for every WWIG ray tracing event, it was verified that energy of the particle is within the energy bounds of the current WWIG being used. Analysis using the first transport geometry with the void material confirmed that the appropriate WWIG was being used given the initial particle energy. Analysis of the simulation with the



(a) $E_{upper} = 1.5 \times 10^{-2}\,\text{MeV}$

(b) $E_{upper} = 1.5 \times 10^{-1}\,\text{MeV}$

(c) $E_{upper} = 4 \times 10^{-1}\,\text{MeV}$

(d) $E_{upper} = 9 \times 10^{-1}\,\text{MeV}$

(e) $E_{upper} = 1.5\,\text{MeV}$

Figure 4.3: The generated WWIGs used for each energy group. A cutout shows the interior of the geometries.

second transport geometry confirmed that the WWIG was appropriately updated with each change in energy due to collisions. In each case, particles correctly underwent splitting and terminating at the WWIG surfaces, and only at those WWIG surface crossings.

In the case of the second geometry with the lightly scattering medium, we can also compare the WWIG detector tally results to that of the analog and CWWM (shown in Table 4.1). We are interested in knowing that the results from the WWIG simulation match

the analog and CWWM results for each energy group and for the total tally. In Figure 4.4 we can see that the WWIG results agree well in all energy groups except the lowest, where the reported relative error is high for both the analog and CWWM results. For the total neutron flux, the ratio of the WWIG to analog results is 1.0132 ± 0.0120, and 1.0047 ± 0.0146 for the ratio of the WWIG to CWWM results, both of which are considered to be in strong agreement. Because these WWs were not specifically designed for VR of each energy group, which can explain the differences in Figure 4.4, we can accept the agreement in the total neutron flux as indication that the WWIG method yields correct results.

Table 4.1: Detector tally results for each simulation mode for the lightly scattering verification problem setup.

E_{upper} [MeV]	***Analog***		***CWWM***		***WWIG***	
	Flux [1/cm^2]	**Error**	**Flux** [1/cm^2]	**Error**	**Flux** [1/cm^2]	**Error**
1.5×10^{-2}	$4.192\,35 \times 10^{-6}$	0.1128	$7.386\,33 \times 10^{-6}$	0.3354	$4.847\,62 \times 10^{-6}$	0.0471
1.5×10^{-1}	$2.274\,96 \times 10^{-5}$	0.0489	$2.343\,74 \times 10^{-5}$	0.0264	$2.489\,53 \times 10^{-5}$	0.0180
4.0×10^{-1}	$3.908\,09 \times 10^{-5}$	0.0460	$3.673\,48 \times 10^{-5}$	0.0188	$3.780\,83 \times 10^{-5}$	0.0104
9.0×10^{-1}	$6.816\,79 \times 10^{-5}$	0.0126	$6.907\,40 \times 10^{-5}$	0.0213	$6.973\,55 \times 10^{-5}$	0.0065
1.5	$8.014\,88 \times 10^{-5}$	0.0090	$7.950\,49 \times 10^{-5}$	0.0051	$7.987\,54 \times 10^{-5}$	0.0050
Total	$2.143\,40 \times 10^{-4}$	0.0112	$2.161\,37 \times 10^{-4}$	0.0140	$2.171\,62 \times 10^{-4}$	0.0038

4.2 Demonstration

This experiment demonstrates the VR capabilities of the WWIG method compared to the traditional CWWM method.

4.2.1 Problem Setup

For this demonstration, the transport model is a simplified geometry that shares features with a fusion energy system. Helium is evenly distributed in a $1\,\text{m} \times 1\,\text{m} \times 1\,\text{m}$ box whose presence is coincidental to the existence of a 14 MeV isotropic volumetric neutron source.

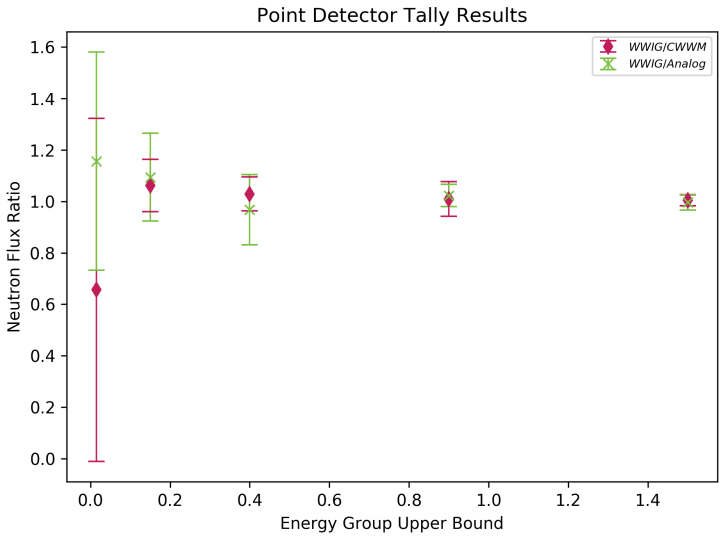


Figure 4.4: Ratio of the WWIG results compared to both the CWWM and analog results of the point detector tally neutron flux for each energy group. Error bars are $\pm 3\sigma$ for the ratio.

The helium is encased in a 0.5 m stainless steel wall (SS316). Two streaming paths, each 5 cm × 5 cm wide, are introduced on the x- and y-axes and the device is surrounded by air. A point detector is located outside the reactor between the two streaming paths as shown in Figure 4.5. All material compositions are defined by the PNNL Material Compendium [40] and interpreted by PyNE [9, 10][1].

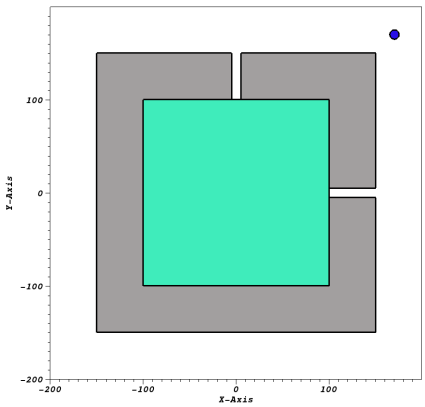
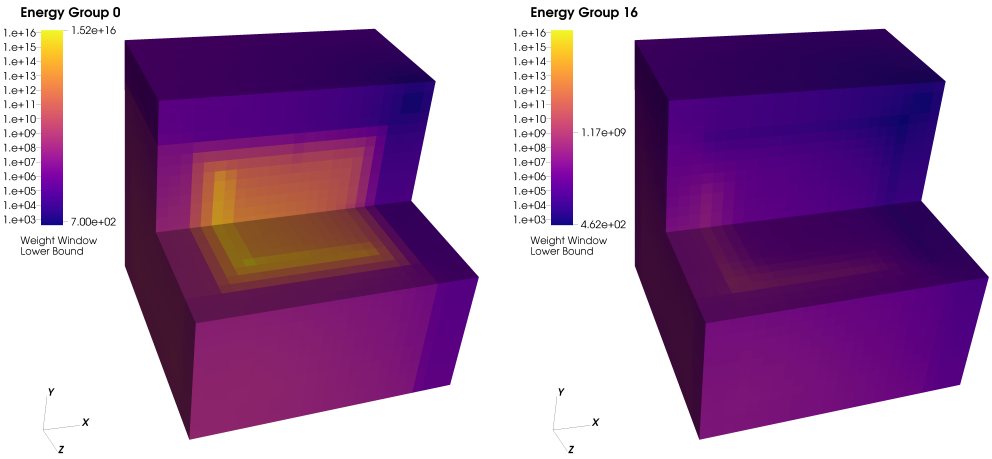


Figure 4.5: A slice at $z = 0$ of the fusion reactor. The helium source (green)is surrounded by a stainless steel wall (gray). A point detector is shown in blue in the upper right quadrant.

The CADIS [7] method in ADVANTG [8] optimized for the point detector was used to

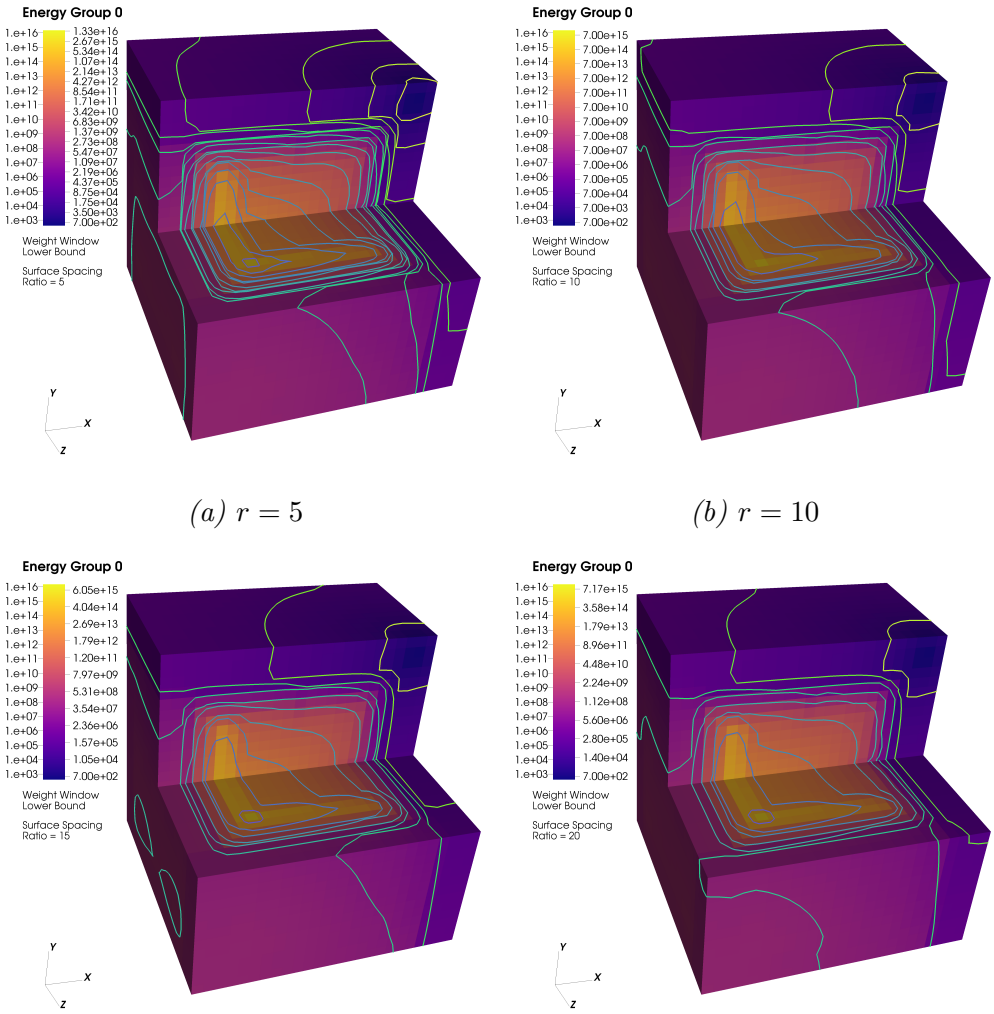
[1]This setup is the same transport geometry used in Figure 3.8c

created a CWWM with $25 \times 25 \times 25$ mesh voxels spanning the full spatial domain. Twenty-seven neutron energy groups were used from the ADVANTG 27n19g library [8]. Figure 4.6 shows cutouts of the CWWM for two example energy groups (E_0 and E_{16}). For images of all energy groups in the CWWM, see Appendix A. Four sets of WWIG geometries were produced from this mesh using different ratio spacings r (Equation (3.1)) for the isosurfaces: 5, 10, 15 and 20. The WW isosurfaces were multiplied by the normalization constant $6.370\,272\,997 \times 10^{-6}$ provided by ADVANTG. The location of the isosurfaces used to generate each of the WWIGs are shown for the two sample energy groups in Figure 4.7 and Figure 4.8. The resulting WWIGs are shown in Figure 4.9 and Figure 4.10. For images of all WWIGs for each energy group at every spacing ratio, see Appendix B. It is important to point out that the WWIGs are generated directly from the data of the CADIS-generated CWWM and that the source biasing scheme is still used in the WWIG simulation. Therefore, we can consider the use of the WWIGs in place of the CADIS-generated CWWM to be consistent with the source biasing schemes required by the CADIS method [7].



(a) E_0 *(b)* E_{16}

Figure 4.6: The resulting CWWM from CADIS for two example different energy groups. The values labeled on the right of the color bar indicate the minimum and maximum WW values for that energy group. A cutout shows the values on the interior.



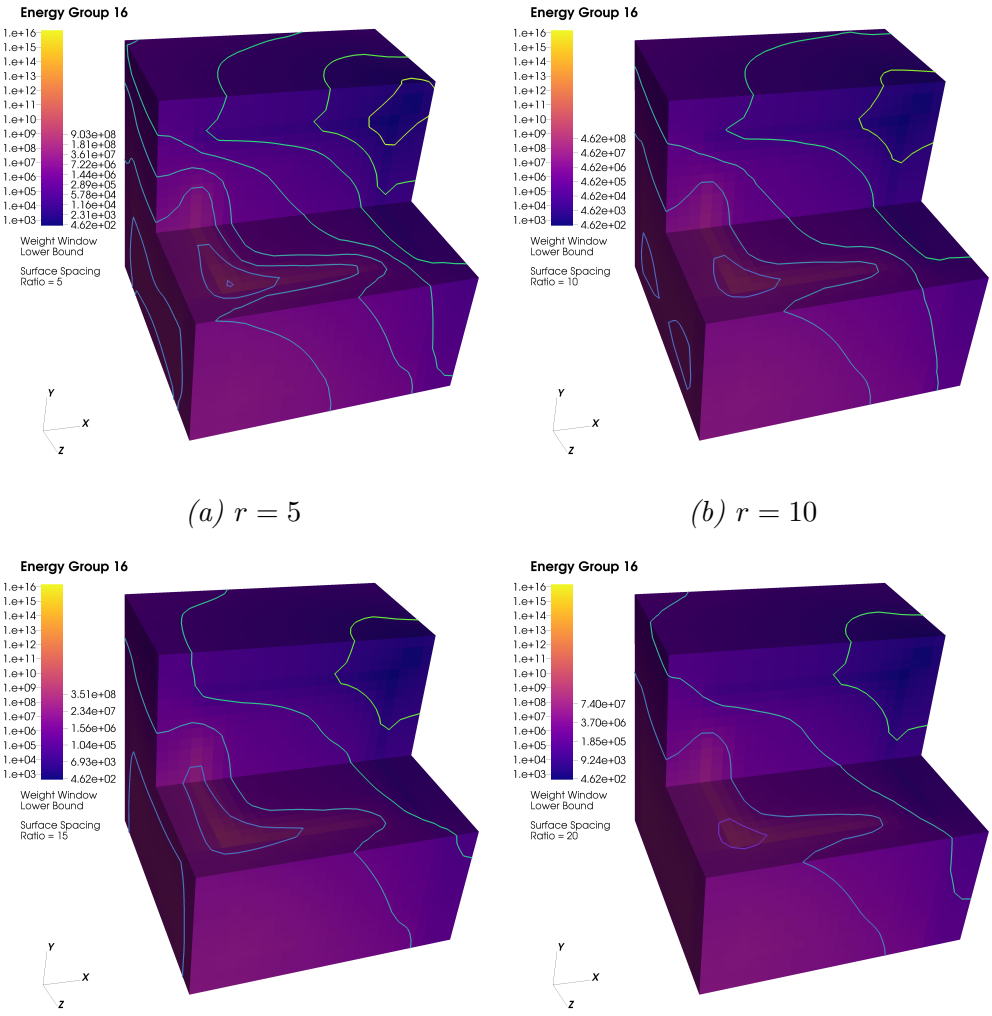
(a) $r = 5$ *(b)* $r = 10$

(c) $r = 15$ *(d)* $r = 20$

Figure 4.7: The CWWM for energy group E_0 with the overlaid isocontours, whose values are indicated on the right side of the color bar, for four different ratio r spacings.

4.2.2 Results and Analysis

Seven modes were simulated: two in analog, one with the CWWM using the standard WW method implemented in MCNP, and four with the different sets of WWIG geometries (one for each r value) using the new WWIG method in MCNP. The first analog run used 10^7 histories and is defined as the reference result. All other simulations used 10^6 histories. The total neutron flux at the detector for each simulation is shown in Table 4.2. Figure 4.11 shows specifically the ratio of the tally results for each mode (E) to the reference result (F)



(a) $r = 5$ *(b)* $r = 10$

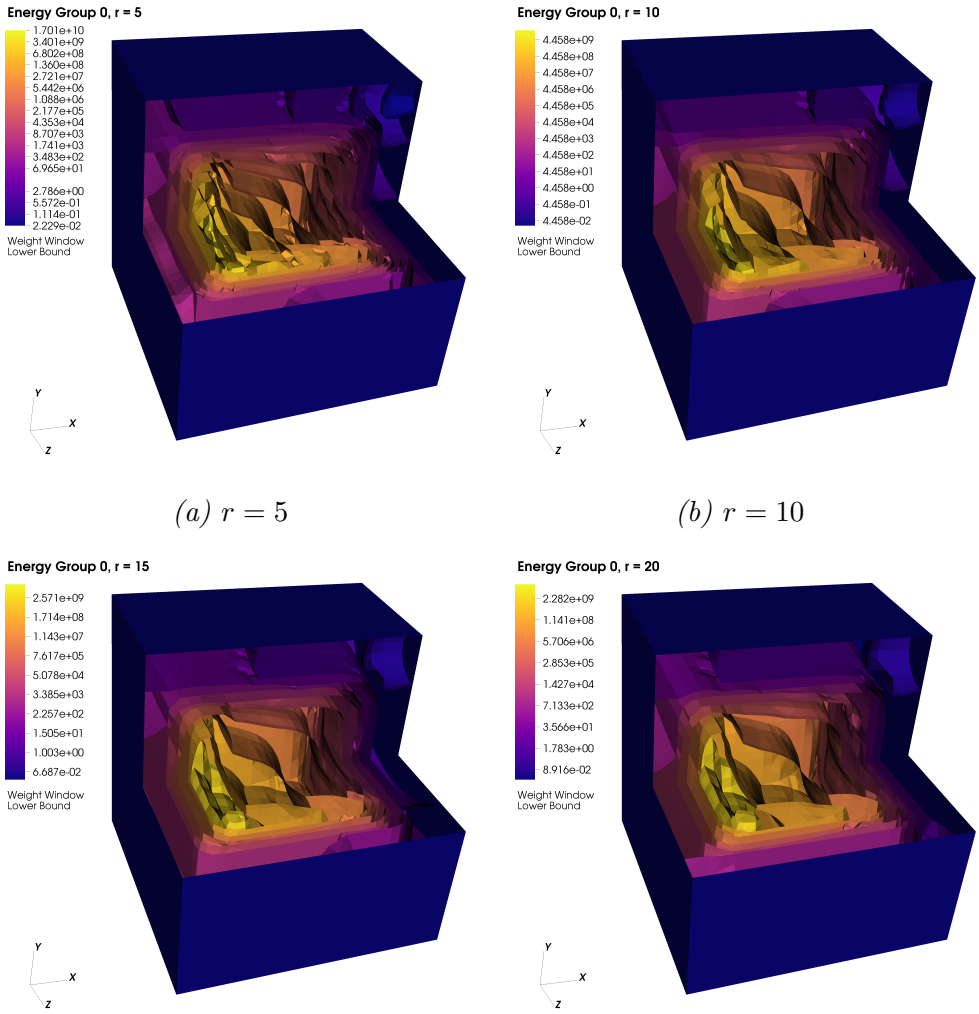
(c) $r = 15$ *(d)* $r = 20$

Figure 4.8: The CWWM for energy group E_{16} *with the overlaid isocontours, whose values are indicated on the right side of the color bar, for four different ratio* r *spacings.*

and whether the values agree within 1σ or 2σ of said ratio. We define the σ for the ratio (E/F) through error propagation given by Equation (4.1).

$$\sigma = \sqrt{\left(\frac{\sigma_E}{\bar{x}_F}\right)^2 + \left(\frac{\bar{x}_E \sigma_F}{\bar{x}_F^2}\right)^2} \tag{4.1}$$

From Figure 4.11 we can see that the simulation results for both the traditional CWWM method as well as the 3 of the 4 WWIG simulations agree with the reference result within 1σ, while the WWIG run with surfaces spaced by a ratio of $r = 10$ agrees within 2σ. Furthermore, from Figure 4.12 we can see that the relative error for each of the WWIG



(a) $r = 5$ *(b)* $r = 10$

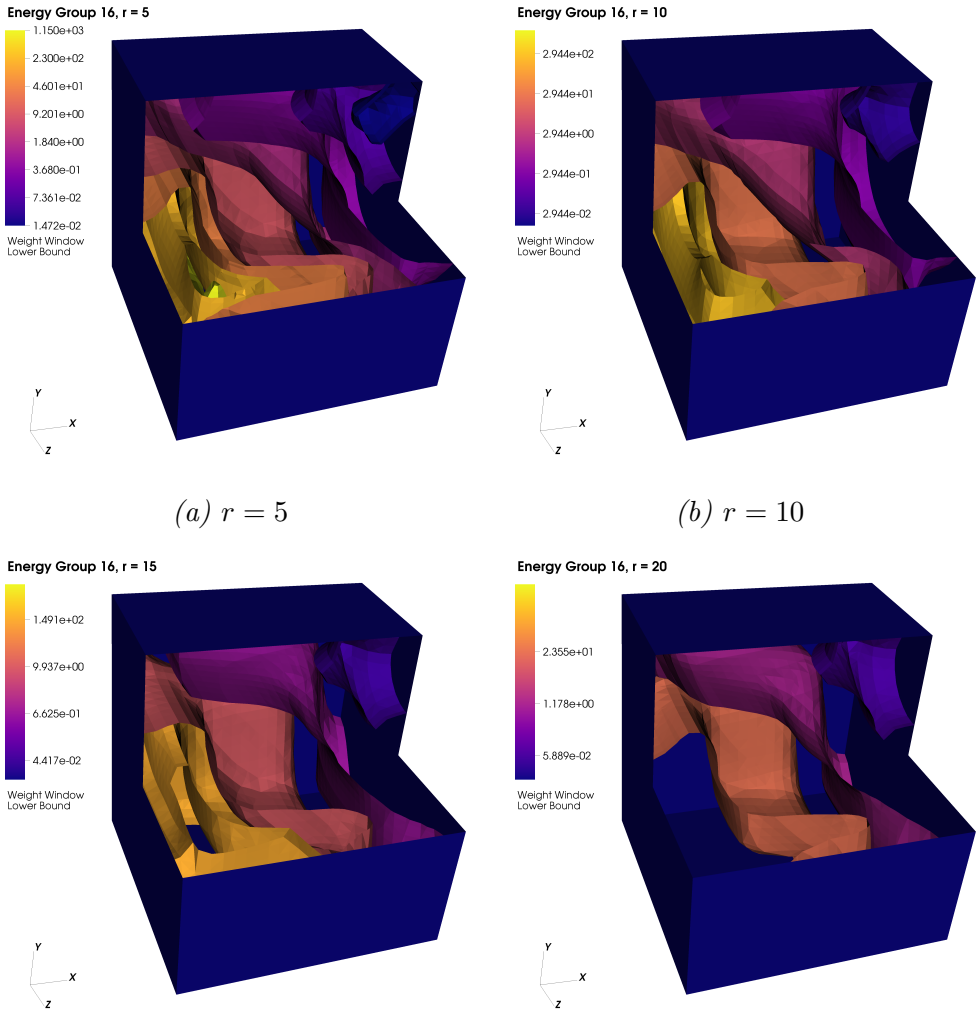
(c) $r = 15$ *(d)* $r = 20$

Figure 4.9: The generated WWIGs for energy group E_0 with each ratio spacing r. The isosurface values are labeled on the color bar.

runs is close to that of the CWWM and much lower than the analog results for the same number of histories. This agreement and low relative error indicates that accurate VR was performed.

4.3 Summary and Conclusions

From the verification results, we can see that the WWIG particle tracking method was correctly implemented in DAG-MCNP 6.2. The results of the demonstration experiment



(a) $r = 5$ *(b)* $r = 10$

(c) $r = 15$ *(d)* $r = 20$

Figure 4.10: The generated WWIGs for energy group E_{16} *with each ratio spacing* r*. The isosurface values are labeled on the color bar.*

show that this is a viable method for using WWIGs in place of CWWMs for VR in MC simulations. The WWIG method produces accurate results and does lower the variance when compared to the corresponding analog simulation, though not necessarily as effectively as the corresponding CWWM in this particular demonstration experiment. The following experiments in Chapter 6 will analyze performance of the WWIGs in more depth.

Table 4.2: Detector tally results for each simulation mode.

Mode		**Neutron Flux** $[1/\text{cm}^2]$	**Relative Error**
Reference		5.5894×10^{-8}	0.0076
Analog		5.5739×10^{-8}	0.0240
CWWM		5.5896×10^{-8}	0.0100
WWIG	$r = 5$	5.6022×10^{-8}	0.0093
	$r = 10$	5.5119×10^{-8}	0.0107
	$r = 15$	5.5265×10^{-8}	0.0121
	$r = 20$	5.5659×10^{-8}	0.0126

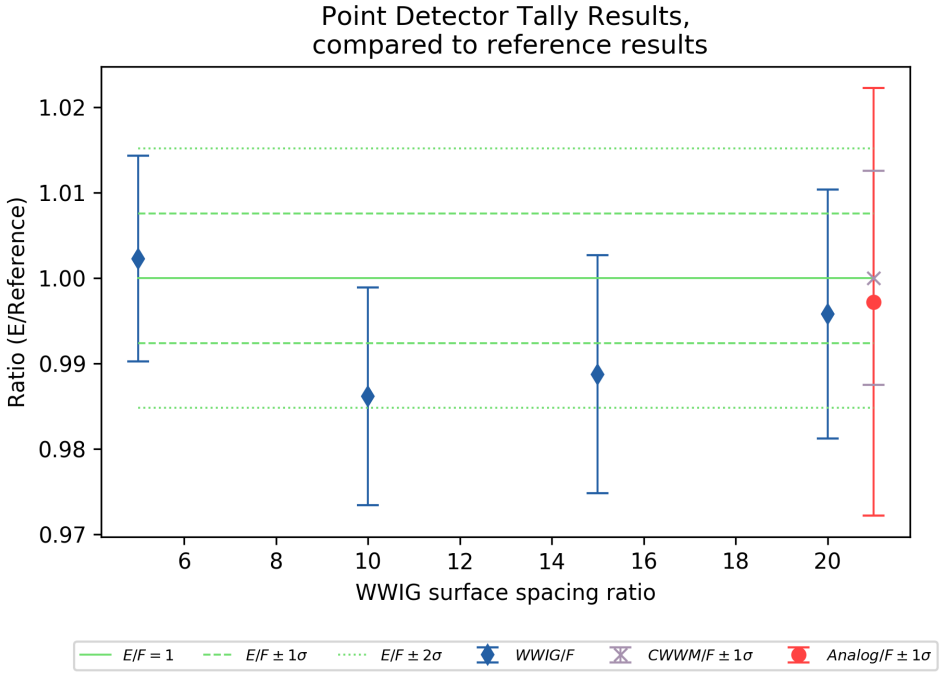


Figure 4.11: The ratio of each simulation (E) compared to the reference result. $E/F \pm 1\sigma$ for each comparison is shown, as well as $E/F \pm 2\sigma$ for each of the WWIG results.

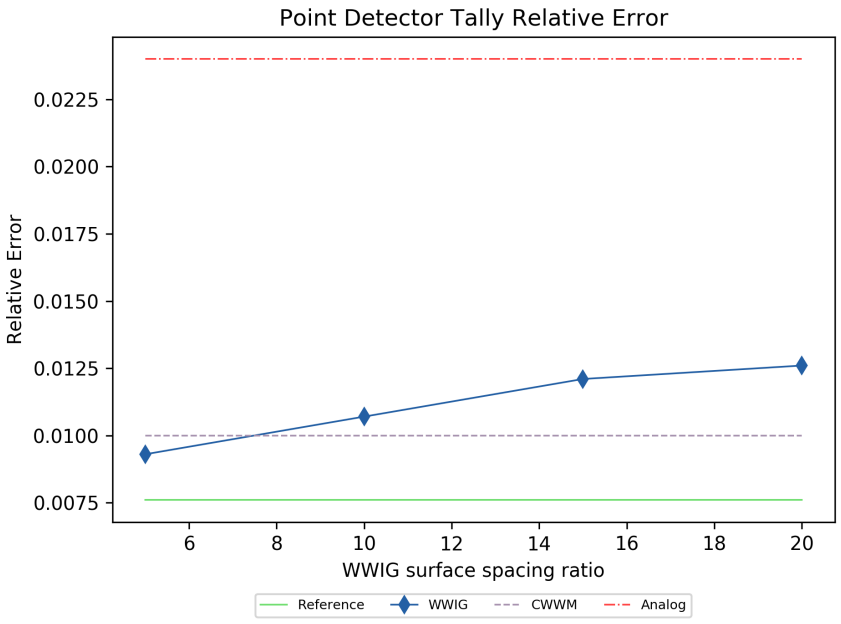


Figure 4.12: Relative errors for each simulation mode.

Chapter 5

Mesh Refinement and Simplification

DAGMC geometries, and therefore the WWIGs in this dissertation, are represented as triangular meshes. Triangular mesh geometries have been used widely in many different fields since their introduction in the late 20th century, and as such there have been a wide array of implementations for achieving various mesh refinement and simplification goals [41, 42, 43, 44]. We use two forms of mesh refinement, decimation and smoothing, to address the aforementioned potential issues with the WWIGs in Section 3.3.

5.1 Decimation

Decimation, sometimes known as mesh coarsening, is the removal and redefinition of facets on surfaces such that the facet density of the surface decreases. This is of particular interest in the field of computer graphics to increase rendering speeds of complex graphics. We are similarly interested in the speedup achieved through decimation in the context of the particle tracking algorithm as well as reduction of the memory footprint. High facet density could mean better resolution of fine or key mesh features in a geometry. However, if a surface has an unnecessarily high facet density, the ray tracing algorithm used by DAGMC can become unnecessarily slow due to a larger OBB tree to search with each ray tracing call. As such, we are interested in coarsening the surfaces of the WWIGs that have high facet density (such as large flat, smooth surfaces) but without losing features that have important impact on VR performance.

5.1.1 Decimation Algorithm

Many different algorithms for decimation and mesh coarsening have been developed over time [41, 45, 46, 47, 48]. Taking advantage of implementations already designed for our data structures, we use the progressive decimation algorithm implemented in VTK [41], a data visualization toolkit for computer graphics, called `vtkDecimatePro` [45, 49] in the mesh refinement tool in IsogeomGenerator [36]. In this implementation based on the algorithm by W. Schroeder et al. [45], multiple passes are made over all the vertices in the geometry until the mesh has been decimated by a desired amount. At each pass, if a vertex meets the criteria for removal, then it is deleted and all the connected triangles are deleted. A vertex meets the criteria for deletion if its position is within a specified distance to the average plane for the surface. This forms a hole in the surface that is then refilled with local triangulation. In the VTK implementation [41, 49], restrictions can be set to ensure the preservation of topology (no creation of holes in the mesh) and to not allow boundary vertex deletion. The latter is important in our case because it ensures that the vertices on the outer edges of all surfaces are preserved, and therefore no gaps are introduced into the geometry between adjoining surfaces (ie, it remains watertight).

5.1.2 Measuring Coarseness

Quantifying surface coarseness is relatively straightforward in that it is the facet density ρ (facets per unit area) for a surface given by Equation (5.1), where N_f is the number of facets on the surface and A_i is the area of the i^{th} facet. The global average coarseness $\bar{\rho}$ for an entire geometry with S surfaces is the average of the coarseness for each surface ρ_j weighted by its surface area A_j given by Equation (5.2).

$$\rho = \frac{N_f}{\sum_{i \in N_f} A_i} \tag{5.1}$$

$$\bar{\rho} = \frac{\sum_{j \in S} \rho_j \cdot A_j}{\sum_{j \in S} A_j} \tag{5.2}$$

We have implemented this method of measuring mesh coarseness on a full geometry or a subset of a meshed geometry in the DAGMC Stats Python package [50], a package designed for measuring and quantifying mesh features in DAGMC-compliant geometries.

5.2 Surface Smoothing

Surface smoothing is another mesh simplification technique that has been of interest to many fields, such as computer graphics and medical imaging, for a long time in order to remove visual artifacts that can impede the look or how we interpret images. In the case of our WWIGs, we are not so focused on the visual perception, but rather the performance increase associated with having smooth surfaces. In the particle tracking algorithm described later in Section 3.2, particle weight is checked at every WWIG surface crossing, so when we have unnecessarily rough or noisy surfaces, those weight checks may occur more frequently than necessary. An extreme example of this can be seen in Figure 5.1 where a particle crossing a rough surface (blue) would result in many more WW checks (red x's) than the corresponding smooth surface (black). As such, we are interested in being able to smooth noisy surfaces that might be present in CWWMs.

5.2.1 Smoothing Algorithm

Surface smoothing being a widely applicable mesh refinement also has been studied extensively with many algorithms developed and implemented [41, 43, 51, 52, 53, 54, 55, 56, 57, 58]. In the IsogeomGenerator tool [36], we once again take advantage of an algorithm already implemented in the VTK library designed for our data structures called `vtkWindowedSincPolyDataFilter`, which is based on the algorithm designed by G. Taubin et al. [41, 51, 59]. This smoothing algorithm iterates on mesh relaxation methods to ad-

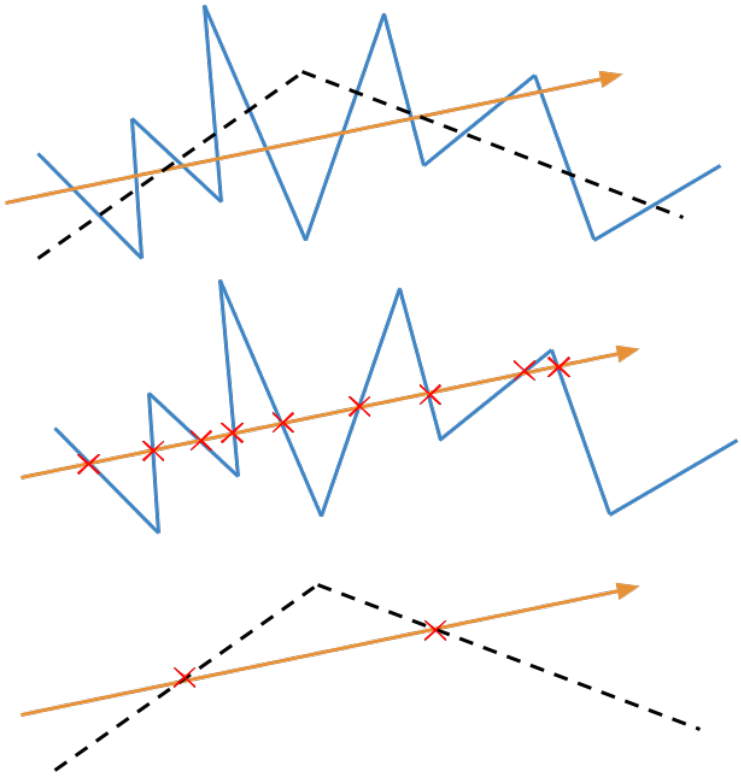


Figure 5.1: An example of a particle crossing a rough surface. The blue, jagged lines represent the facets of a rough surface while the black dashed line is what the facets would be after smoothing. When a particle (trajectory shown in orange) crosses the surface, each time a WW check is performed (shown as red x's).

just vertex locations using higher order Chebyshev polynomials for approximation with each iteration. More iterations yields more intense smoothing. Similar to the decimation implementation, we take advantage again of two restrictions available in VTK that prevent the smoothing of boundary vertices as to not introduce gaps between surfaces and that prevent manifold smoothing which preserves surface topology (no interior holes created).

5.2.2 Measuring Roughness

There are many different methods for measuring roughness of meshed surfaces [60, 61, 62]. For the purpose of this work, we use the method developed by K. Wang et al. [62] as it adequately quantified differences in surface roughness values for our purposes. In this method we calculate a global average of the local roughness values for an entire meshed geometry or

subset of the geometry to be $\overline{LR}$ given by Equation (5.3) [62], where LR_i (Equation (5.4)) is the calculated local roughness at vertex v_i, $N_i^{(F)}$ and $N_i^{(V)}$ are the sets of neighboring facets and vertices, respectively, to vertex v_i, and s_i is 1/3 of the total area of $N_i^{(F)}$. The definition for the α and β angles are shown in Figure 5.2.

$$\overline{LR} = \frac{\sum_i LR_i s_i}{\sum_i s_i} \tag{5.3}$$

$$LR_i = \left| GC_i + \frac{\sum_{j \in N_i^{(V)}} D_{i,j} GC_j}{D_{i,i}} \right| \tag{5.4}$$

$$GC_i = \left| 2\pi - \sum_{j \in N_i^{(F)}} \alpha_j \right| \tag{5.5}$$

$$\begin{cases} D_{i,j} = \frac{\cot \beta_{ij} + \cot \beta'_{ij}}{2} & \text{for } j \in N_i^{(V)} \\ D_{i,i} = -\sum_j D_{i,j} & \text{for } j = i \end{cases} \tag{5.6}$$

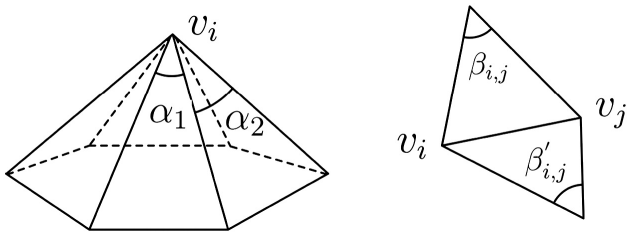


Figure 5.2: Definition of α and β angles in relation to connected vertices v_i and v_j on a mesh. (Figure from K. Wang et al. [62])

This method for measuring mesh surface roughness on a full or subset of a geometry has also been implemented in the DAGMC Stats package [50].

Chapter 6

WWIG Performance Analysis

This chapter analyzes how different geometric features of WWIGs described in Section 3.3 affect the MC VR quality and performance. Specifically, we are interested in understanding how isosurface spacing, mesh coarseness, and surface roughness affect WW check efficiency, the FOM, and memory footprint. This chapter will describe how to we measure efficiency and performance and then present a set of three experiments for analyzing each of the geometric features described.

6.1 Measuring Performance

This section provides the details for how performance of MC VR with WWs was measured in the following experiments.

6.1.1 Figure of Merit

The figure of merit (FOM), described in Section 2.1, is used as the most common measure of performance and is one we use in this analysis as well. To measure FOM for a tally, MCNP [1, 6] uses the computational time t_{proc} (CPU time) and the tally relative error R in Equation (2.4). However there can be uncertainty in this measurement because there is uncertainty in the relative error known as the variance of the variance (VOV). MCNP defines the VOV as the relative variance of the relative error [1]. Propagating this uncertainty we can get the standard deviation for the relative error σ_R in Equation (6.1), where σ_v^2 is the VOV. From this we can calculate the standard deviation of the FOM σ_{fom}, again with

uncertainty propagation, in Equation (6.2). In this case, we only consider the uncertainty of the relative error and not the uncertainty in CPU time. While there is likely uncertainty in this measurement, it is not reported and therefore we will consider it to be zero.

$$\sigma_R = R\sqrt{\sigma_v^2} \tag{6.1}$$

$$\sigma_{fom} = \frac{2\sqrt{\sigma_v^2}}{R^2 t_{cpu}} \tag{6.2}$$

6.1.2 Weight Window Efficiency

One performance measurement we are particularly interested in when using WWIGs rather than the CWWMs is the WW efficiency. One goal with using WWIGs is to improve WW efficiency by only checking particle weight when the WW is expected to have an effect (either with splitting or stochastic termination), as opposed to times when it is convenient in the code which is how the use of CWWMs is implemented with MCNP [1, 6]. When using WWIGs, the isosurfaces indicate when the WW values in phase space have changed significantly and will likely warrant a weight change. Therefore only checking and applying WWs when a particle crosses a surface of the WWIG is expected to improve WW efficiency compared to using a CWWM. It is important to note that there are many possible ways to assess the performance of WWs, but because the WWIG method focuses on optimizing *when* particle weights are checked this WW efficiency metric was chosen to best assess that ability.

Overall WW efficiency η_{ww} can be calculated using Equation (6.3) where N_{ww} is the total number of WW checks, N_{split} is the number of WW checks with initial particle weight $w > w_U$ (leading to splitting), and N_{term} is the number of WW checks with initial particle weight $w < w_L$ (leading to possible stochastic termination). A higher value for η_{ww} is desirable because it indicates that particle weight checks are happening when they have an effect on

the particle's weight (either by splitting or stochastic termination).

$$\eta_{ww} = \frac{N_{split} + N_{term}}{N_{ww}} \tag{6.3}$$

We are also interested in knowing what fraction of effective weight changes lead to splitting (f_{split}) or possible stochastic termination (f_{term}) because this can give insight into what may be causing variation in overall efficiency. These quantities are defined in Equations (6.4) and (6.5).

$$f_{split} = \frac{N_{split}}{N_{ww}} \tag{6.4}$$

$$f_{term} = \frac{N_{term}}{N_{ww}} \tag{6.5}$$

We can further analyze the efficiency of particle splitting specifically by calculating the fraction of splitting events that yield a number of new particles greater than ($f_{>C_U}$), less than ($f_{<C_U}$), and equal to ($f_{=C_U}$) the WW upper bound constant C_U. These quantities are given by Equations (6.6) to (6.8) where $S_{>C_U}$, $S_{<C_U}$, and $S_{=C_U}$ are the number of times that a splitting event led to a number of split particles greater than, less than, or equal to, respectively, C_U. A high value for $f_{>C_U}$ indicates that splitting may not be occurring frequently enough (under checking) because the particle weight is much greater than $C_u \times w_U$, yielding many particles with each splitting event. Conversely a high value for $f_{<C_U}$ indicates that possibly too frequent particle splitting is occurring (over checking), which can lower overall efficiency, because few particles are being produced with each splitting event. Ideally, $f_{=C_U}$, which we will refer to as the "splitting efficiency" moving forward, should be as close to one as possible which indicates perfect splitting efficiency (no over checking or under checking).

$$f_{>C_U} = \frac{S_{>C_U}}{N_{split}} \tag{6.6}$$

$$f_{<C_U} = \frac{S_{<C_U}}{N_{split}} \tag{6.7}$$

$$f_{=C_U} = \frac{S_{=C_U}}{N_{split}} \tag{6.8}$$

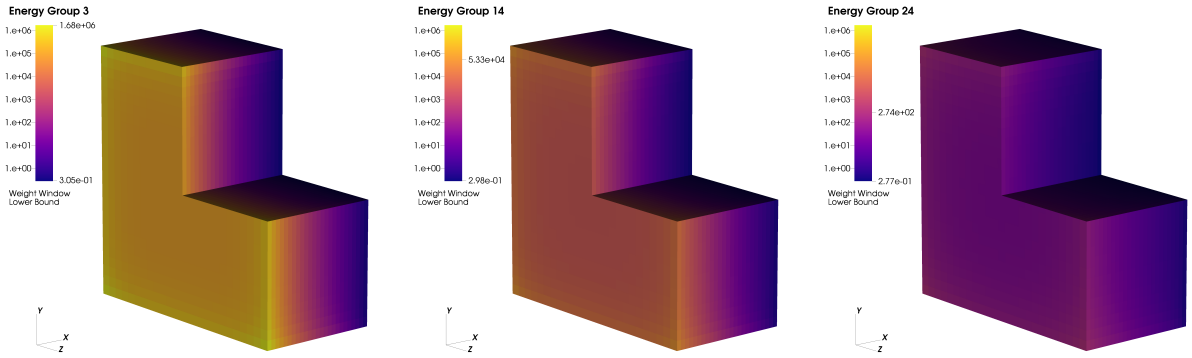
6.2 Surface Spacing Experiment

When using WWIGs, the user must select which WW lower bound values to use as the isosurfaces in the geometries as described in Section 3.1. This choice of isosurfaces is expected to affect performance, and therefore it is important to understand how the choice of surfaces can affect this. This experiment is designed to analyze how the choice of surface values, specifically the spacing of the isosurfaces when spaced by some ratio r, affects VR quality, WW efficiency, and overall performance. Various surface spacing ratios (given by Equation (3.1)) were used to generate various sets of WWIGs and used to compare to CWWM and analog simulations.

6.2.1 Problem setup

For this experiment, the transport geometry is a concrete slab (material defined by the PNNL Material Compendium [40]) sized $50\,\text{cm} \times 100\,\text{cm} \times 100\,\text{cm}$ with reflecting surfaces on the y and z planes. The $-x$ plane has a 14 MeV evenly distributed surface source emitted monodirectionally in the $+x$ direction. A surface tally is located on the $+x$ plane. The CADIS [7] implementation in ADVANTG [8] was used to generate a CWWM with resolution $25 \times 25 \times 25$ mesh voxels and optimized for the surface tally. The WW upper bound constant was set to $C_U = 7$ and survival weight constant set to $C_S = 4$ (as opposed to the default, and typically used, values of 5 and 3, respectively). These values were chosen in order to assess the overall and splitting efficiency for cases where $r < C_U$. Due to the limitations for the spacing ratio described by Equation (3.8), this particular CWWM did not allow for $r < 5$; therefore the upper bound and survival weight constants were adjusted accordingly. All 27 neutron energy groups of the ADVANTG 27n19g library [8] were used. The resulting

CWWM, which can be seen in Figure 6.1, has a relatively smooth and flat gradient in the x direction for the WW lower bound which allows us to generate various sets of WWIGs to effectively test the surface spacing. The values for the isosurfaces in the WWIGs were multiplied by the normalization factor of $8.738\,889\,025 \times 10^{-3}$ provided by the ADVANTG output (see Footnote 1 in Section 3.1). For images of all energy groups in the CWWM see Appendix C.



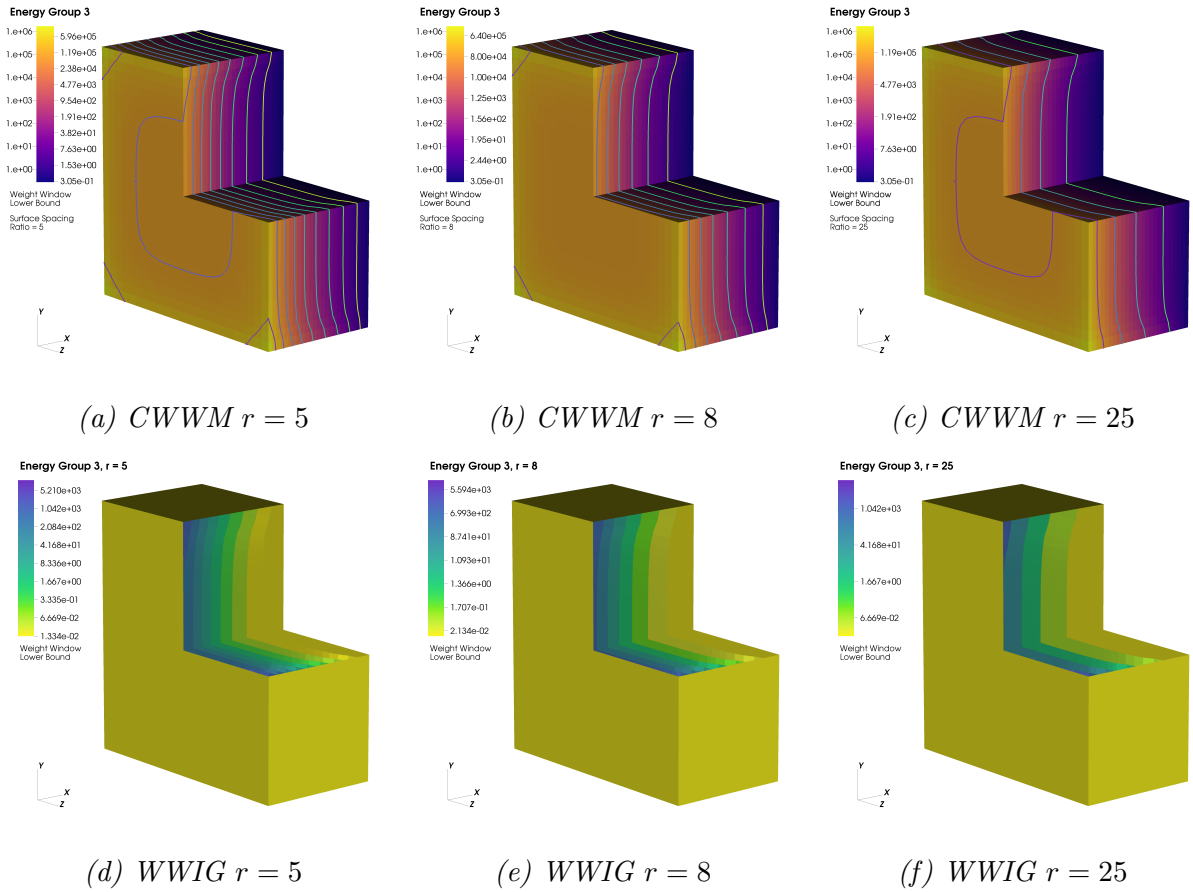
(a) E_3 *(b)* E_{14} *(c)* E_{24}

Figure 6.1: Three example energy groups of the CWWM used for all experiments. A cutout on the y and z plane at the origin shows the interior of the mesh. The color bar on each plot ranges from the global minimum and maximum WW values for all of the energy groups shown. The right side of the color bar indicates the minimum and maximum values for the individual energy groups.

Thirteen different sets of WWIG geometries were generated with different spacings between their isosurfaces using Equation (3.1) with surface spacing ratio r ranging from 5 to 25 ($r = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25\}$). Images of the WWIGs for energy group E_3 at a few select surface spacing ratio values can be seen in Figure 6.2. For images of all WWIGs for each energy group at every surface spacing ratio, see Appendix D.

6.2.2 Results and Analysis

Each of the following simulations used 10^5 histories, with the exception of the reference simulation which was an analog run with 10^6 histories. The results for total neutron flux for the surface tally can be found in Table 6.1 and plotted as a ratio compared to the reference



(a) CWWM $r = 5$ *(b) CWWM $r = 8$* *(c) CWWM $r = 25$*

(d) WWIG $r = 5$ *(e) WWIG $r = 8$* *(f) WWIG $r = 25$*

Figure 6.2: The top row shows the CWWM for energy group E_3 with the isosurfaces overlaid for various surface spacing ratios r. The right side of the legend indicates the WW values in the CWWM that were used for the isosurfaces. The bottom rows shows the corresponding WWIG geometries for the same energy group and surface spacings. The labels on the color bar on the bottom row indicate the isosurface values. A cutout on the y and z planes at the origin shows the interiors of the geometries.

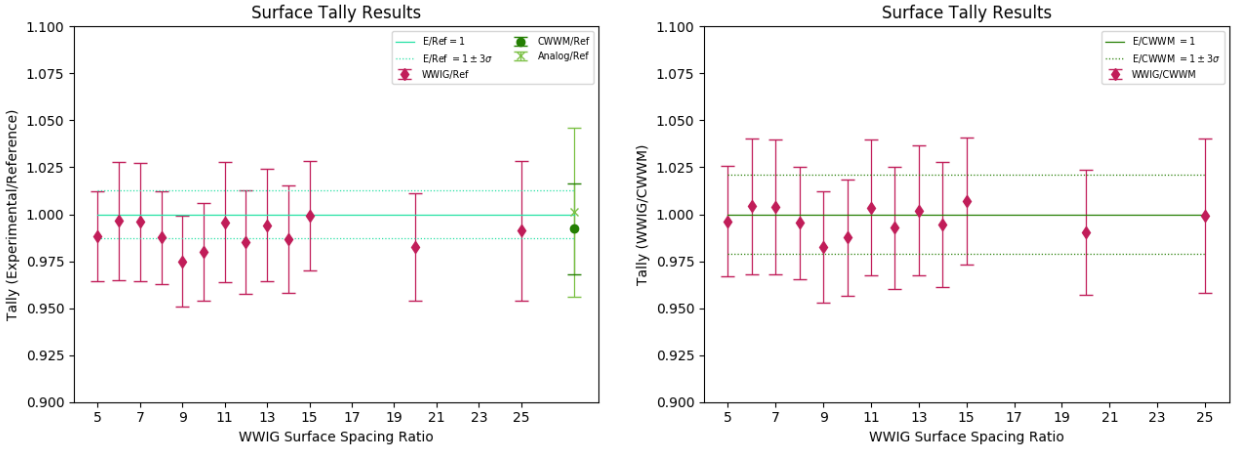
result and the CWWM result in Figure 6.3.

From Figure 6.3 we can see that there is fairly strong agreement with both the reference and CWWM results and that the WWIG results are evenly distributed around some mean value. In particular, we see in Figure 6.3b that the tally means are evenly distributed around the accepted results for the CWWM mode, indicating that there is no biasing of the tally mean by the act of using WWIGs.

In Figure 6.4 we see there is variability in the relative error R for each simulation, and in particular, variability of the VOV (indicated by the error bars). However, because there

Table 6.1: Surface tally results for different WWIG surface spacings.

Mode		Neutron Flux $[1/\text{cm}^2]$	Relative Error	VOV
Reference		1.4383×10^{-5}	0.0042	0.0338
Analog		1.4401×10^{-5}	0.0144	0.1412
CWWM		1.4271×10^{-5}	0.0070	0.0691
WWIG	$r = 5$	1.4217×10^{-5}	0.0069	0.0029
	$r = 6$	1.4331×10^{-5}	0.0097	0.0928
	$r = 7$	1.4325×10^{-5}	0.0097	0.0902
	$r = 8$	1.4206×10^{-5}	0.0072	0.0026
	$r = 9$	1.4023×10^{-5}	0.0072	0.0020
	$r = 10$	1.4094×10^{-5}	0.0078	0.0075
	$r = 11$	1.4322×10^{-5}	0.0098	0.0792
	$r = 12$	1.4170×10^{-5}	0.0084	0.0179
	$r = 13$	1.4300×10^{-5}	0.0091	0.0235
	$r = 14$	1.4192×10^{-5}	0.0087	0.0141
	$r = 15$	1.4371×10^{-5}	0.0088	0.0236
	$r = 20$	1.4134×10^{-5}	0.0088	0.0451
	$r = 25$	1.4259×10^{-5}	0.0118	0.1125



(a) Reference result comparison *(b) CWWM result comparison*

Figure 6.3: Total neutron flux for the surface tally plotted as a ratio compared the reference and CWWM results. Error bars are 3σ for the ratio of the results.

is no clear trend for this variability as a function of the surface spacing, we can assume that any variability of the tally means and error are due to general artifacts of VR methods and not specific to using WWIGs.

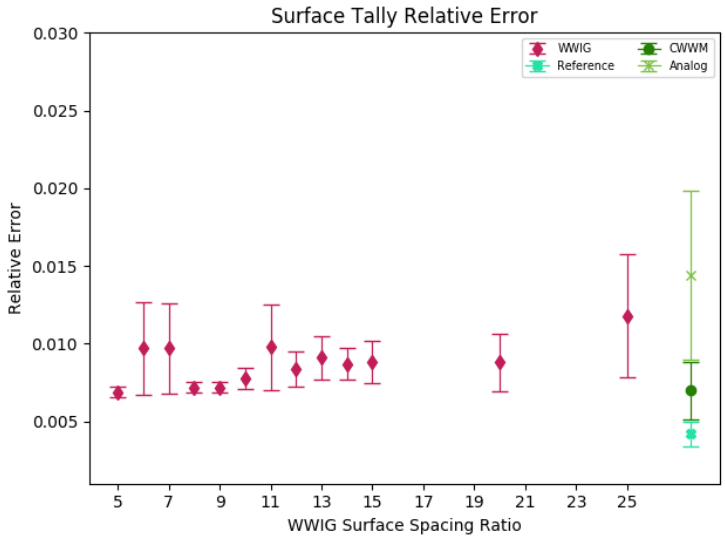
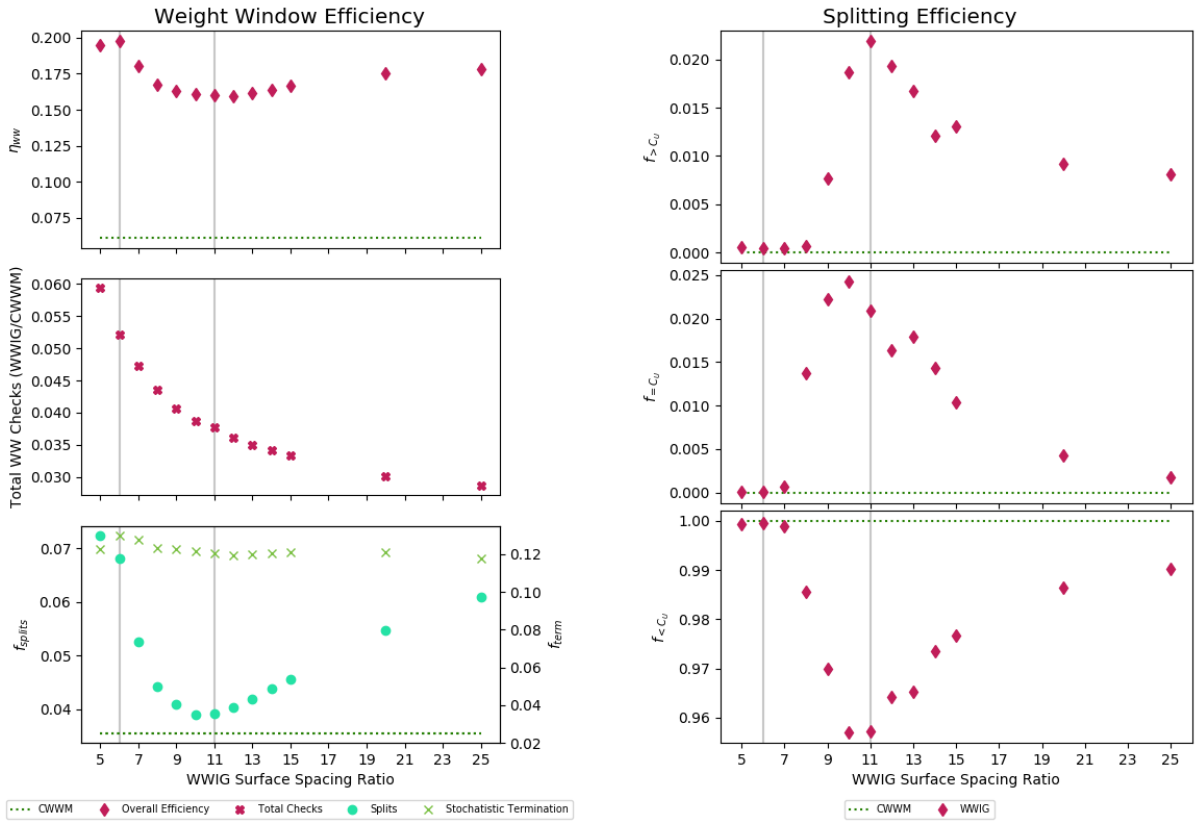


Figure 6.4: Relative error for each different surface spacing ratio. The error bars are given by Equation (6.2).

The spacing of the WWIG surfaces is expected to affect the WW efficiency and we can see this in Figure 6.5. In the middle plot of Figure 6.5a we can see that the total number of WW checks in each of the WWIG runs is much lower than the CWWM run and that it decreases as the surface spacing increases. This trend in the overall decrease is expected because there are fewer surfaces in the WWIGs as their surface spacing ratio increases, and therefore fewer opportunities to apply WWs. However, we do not see the same trend with the overall efficiency for the WWIGs in the top plot of Figure 6.5a.

Although there is a change in η_{ww} as a function of surface spacing, the promising feature from every WWIG result is that the overall WW efficiency η_{ww} is significantly higher than that of the CWWM run. For the CWWM run $\eta_{ww} = 6.094 \times 10^{-2}$ while η_{ww} for each of the WWIG results is roughly 30 times higher (seen in the top plot of Figure 6.5a). We also see fairly consistent VR across all WWIG results that is comparable, if not better in some cases, to that of the CWWM results. This is important in that it signifies a user can increase the

WWIG surface spacing ratio significantly before seeing a drastic change in the VR quality and performance.



(a) WW check efficiency and breakdown *(b) Splitting efficiency*

Figure 6.5: WW metrics as a function of the WWIG surface spacing. The CWWM metrics are also included. The vertical gray lines indicate the three separate regions discussed.

We are, however, interested in explaining the minor yet unexpected trend of the overall efficiency seen in Figure 6.5. Upon first thought, one might expect efficiency to steadily increase with the increase in r because if there are fewer opportunities to apply WWs when surfaces are further apart in physical space, then when they are applied the likelihood that splitting or stochastic termination is required will be higher. However, this is not the case and we hypothesize that collisions are effecting this metric. When a collisions occurs two things happen to surviving particles: implicit capture and a change in energy. With the former, the weight of the particle changes at each collision as described in Section 2.1.1. And with the latter, a significant change in particle energy can cause the particle to change to a new

energy group defined by the WWs, leading to the particle being tracked on a new WWIG for the new, usually lower, energy group. In the case of this configuration, the lower the energy group, the larger the gradient in the x-direction is for the WWs and therefore there are more surfaces in the WWIG geometries. When a particle switches to the lower energy group, the next WWIG surface is likely to be closer in physical space, with a very different WW lower bound value, on the new WWIG compared to the previous higher energy WWIG. This can lead to a higher portion of WW checks that have no effect, decreasing overall efficiency or leading to less efficient splitting.

Considering these two competing factors at play (effect due to purely surface spacing and effect due to collisions), we can divide Figure 6.5 into three regions (approximated by the vertical gray lines) to examine which effects are dominating under various circumstances. In the far left region for $r \leq 6$, the dominating effect contributing to the overall efficiency η_{ww} is the pure number of WW checks and splitting occurring. The highest portion events are splitting events in this region, contributing to a higher overall higher efficiency. The WW checks occur frequently enough here that a weight change is likely occurring more frequently due to the WWs rather than collisions. It is important to note here that all splitting events lead to fewer than C_U particles ($f_{<C_U} = 1$ in Figure 6.5b). This is because implicit capture lowers the particle weight during collisions, likely causing the number of particles resulting from splits to be lower than expected (particle weight is closer to, though still greater than, the upper WW value).

In the middle region ($6 < r < 11$), we see a peak in splitting efficiency as $f_{=C_U}$ is highest (Figure 6.5b), but we also see a peak in $f_{>C_U}$, indicating that under checking is occurring. The increase in $f_{>C_U}$ is due to the WWIG surfaces being too far apart in physical space, leading to a higher number of particles with weights well above the WW upper bound. However, the larger physical spacing between the WWIG surfaces also means that there are more collisions occurring, leading to decreased weight due to implicit capture. This implicit capture is likely bringing the particle weights closer to the WW upper bound constant which

leads to the high number of splits where the resulting number of particles is equal to C_U ($f_{=C_U}$). Initially it was expected that this peak in splitting efficiency would be the result purely of the WWIG surface spacing, but we see that it is more likely due to implicit capture. Because the overall WW efficiency decreases in this region (top plot of Figure 6.5a), we can say that the dominating factor is the change in weight and energy due to collisions, which still lowers overall WW efficiency.

Finally in the third region ($r > 11$), there is more balance between the surface spacing and the effects of collisions. We see an overall higher portion of WW events that lead to splitting, which leads to increased overall efficiency (Figure 6.5a). In Figure 6.5b we see that the peak splitting efficiency ($f_{=C_U}$) begins to lower again. This larger separation of the surfaces in physical space leads to more collisions between the application of WWs, just like before, so we see a decrease in the splitting efficiency $f_{=C_U}$. However, there are fewer overall WW checks meaning that overall efficiency still increases.

We are also interested in understanding how general computational performance metrics are affected by WWIG surface spacing. The first is the FOM and computational time (given by CPU time) which is shown in Figure 6.6. As expected, the computational time decreases as the WWIG surfaces become more separated. This is because there are fewer WW checks and fewer splitting events, leading to fewer overall particles being tracked. The trends of the drops and plateaus in the CPU time correspond directly to the total number of isosurfaces present across all energy groups in each set of WWIGs. However, we do not see the same drastic increase in FOM due to this lower computational time due to the variability of the relative error and the VOV, and therefore cannot conclude whether or not the FOM is truly affected by WWIG surface spacing.

It is important to note that the FOM for the CWWM simulation was 8246, much higher than that of the WWIG simulations. However, because the WWIG implementation has not been fully optimized from a computational stand point, we have determined that comparison of the WWIG FOM to that of the CWWM is an unfair comparison for the time being. We

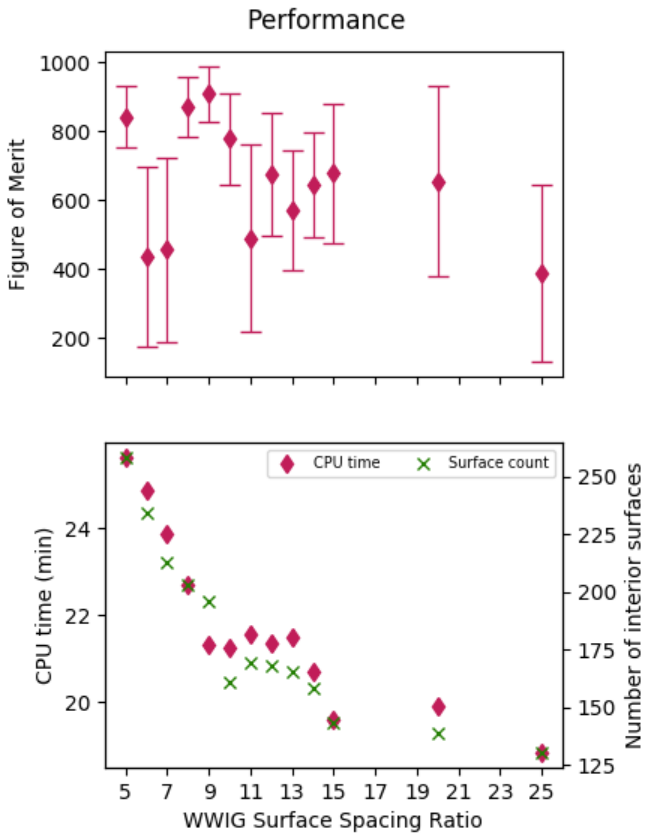


Figure 6.6: Computational performance as a function of WWIG surface spacing. The error in the FOM is given through error propagation of the relative error and VOV. The bottom plot also includes the total number of interior surfaces as a function of surface spacing.

instead are interested in the effect of the WWIG geometric features, and therefore will assess FOM only in the context of the WWIGs moving forward.

The other metric that one might be interested in is the memory footprint of the WWIG files. The total size of all energy group files can be seen in Figure 6.7. As expected, the memory footprint decreases as the WWIG surface spacing increases because there are fewer surfaces, and therefore fewer data points to store. However, it is still overall greater than the memory footprint for the CWWM simulation. While this is not ideal, it is expected because the current generation method and implementation of WWIGs requires that each energy group G be a separate geometry. Each geometry contains a number of surfaces S_g for the energy group, where information about the mesh connectivity c (where c is the vertex

information (x, y, and z) and their connectivity) but only a single WW value must be stored. Therefore the necessary amount of data D to be stored for the WWIGs is given by Equation (6.9). Conversely, CWWMss are nominally a single geometry with data values stored for each energy group on each mesh voxel ($GN_xN_yN_z$, as described in Section 2.1.2). However, it may not be necessary to store a separate WWIG or have separately defined surfaces for each WWIG energy group, which could lead to an overall lower memory footprint. See Chapter 8 for further discussion on this topic. Additionally, mesh coarsening can be employed to limit the size of c as described in the following experiment in Section 6.3.

$$D = \sum_{g \in G} \sum_{s \in S_g} (c_s + 1) \tag{6.9}$$

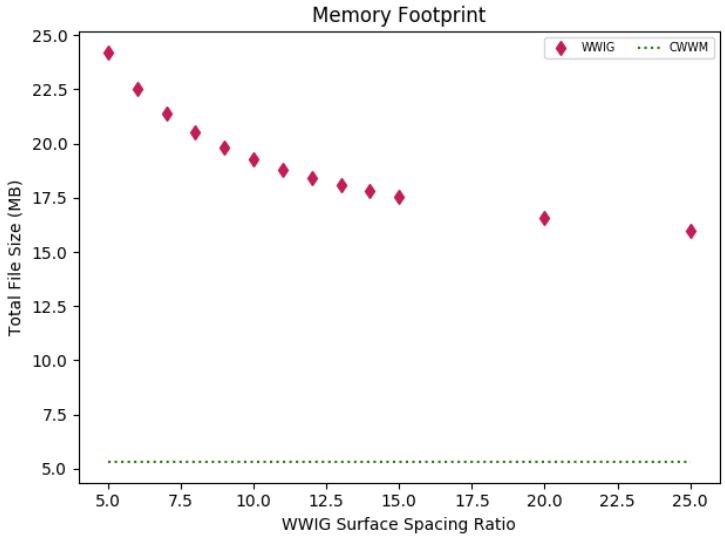


Figure 6.7: Total file size for all energy groups as a function of surface spacing.

6.3 Mesh Coarseness Experiment

The purpose of this experiment is to understand how the memory footprint of the WWIG files can be decreased without compromising performance. As described in Section 3.3.4, the surfaces of the WWIGs can be comprised of unnecessarily fine meshes. When meshes are

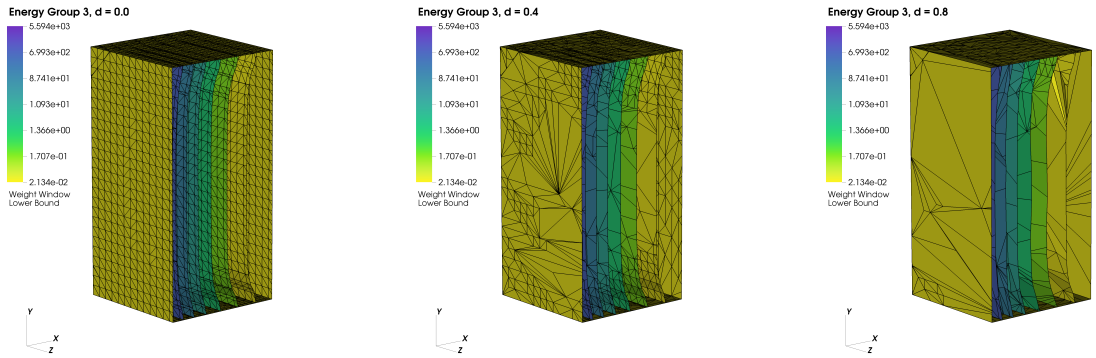
unnecessarily fine, data about the mesh connectivity is high as described in Equation (6.9), which is what leads to an increased memory footprint. Ideally, coarsening the meshes to reduce the connectivity information should have minimal effect on VR performance while still realizing the benefits of a reduced memory footprint.

6.3.1 Problem Setup

To demonstrate this, eight new sets of WWIGs were created. The set of WWIGs with surface spacing $r = 8$ from the previous surface spacing experiments in Section 6.2.1 were decimated in various amounts. The $r = 8$ set was chosen because there was good agreement with the reference results, low relative error, and some fraction of splitting was present for all three measurements presented in Figure 6.5b. The IsogeomGenerator tool [36] described in Section 5.1 was used to apply eight different decimation factors d ranging from 0.1 to 0.8. Figure 6.8 shows how the facets of the mesh change for various decimation factors for a single example energy group. For images of all WWIGs for each energy group at every decimation factor, see Appendix E. To calculate a representative single coarseness value for each set of WWIGs, the surface area-weighted global average mesh coarseness (Equation (5.2)) was calculated for each energy group and then all energy groups were equally averaged together. This average mesh coarseness for each decimation factor is seen in Figure 6.9 and the total memory footprint for the files is seen in Figure 6.10. In both cases we see a linear decrease in coarseness and file size, which is expected.

6.3.2 Results and Analysis

Each set of WWIGs was used again in a simulation with 10^5 histories and WW constants $C_U = 7$ and $C_S = 4$. The surface tally results can be seen in Table 6.2 and compared to both the reference and CWWM results in Figure 6.11. There is once again good agreement with both the reference results and CWWM results in most cases except for two sets of WWIGs (which correspond to the decimation factors of 0.6 and 0.7). It is unclear why these



(a) $d = 0.0$ *(b)* $d = 0.4$ *(c)* $d = 0.8$

Figure 6.8: The WWIG for energy group E_3 after various decimation factors d have been applied. The mesh facets are shown outlined in black and a cutout at $z = 0$ shows the interior mesh surfaces.

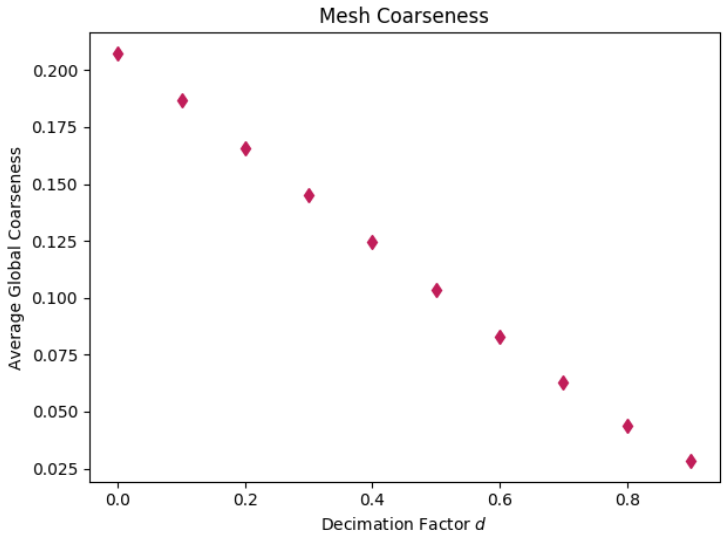


Figure 6.9: Average coarseness for all energy groups.

two sets of WWIGs resulted in such high relative error and VOV (seen in Figure 6.12). One possible explanation is that during the decimation process with VTK, the surface was altered too much in a key locations that caused the loss of necessary detail. Another possible explanation is that the general act of applying VR caused rare high weight events to occur that perturbed the results which is not uncommon with VR [2]. However, because the VOV is significantly high in these two cases, it is possible the results would converge with more histories. Furthermore, the agreement among all the other sets of WWIGs and the

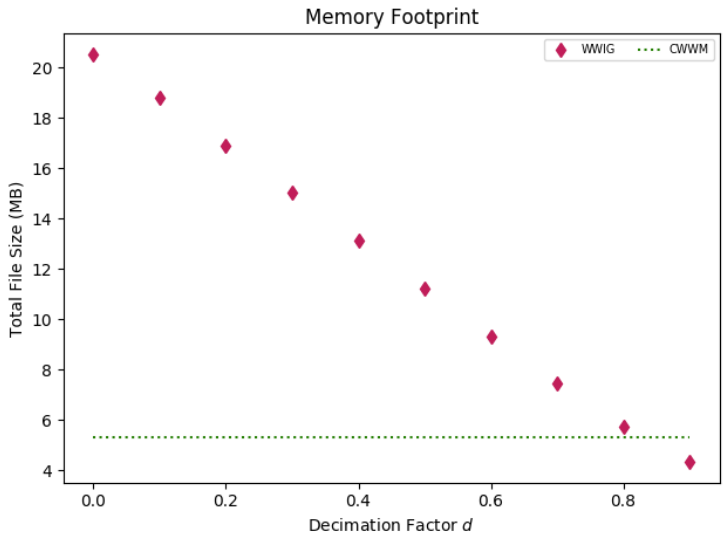
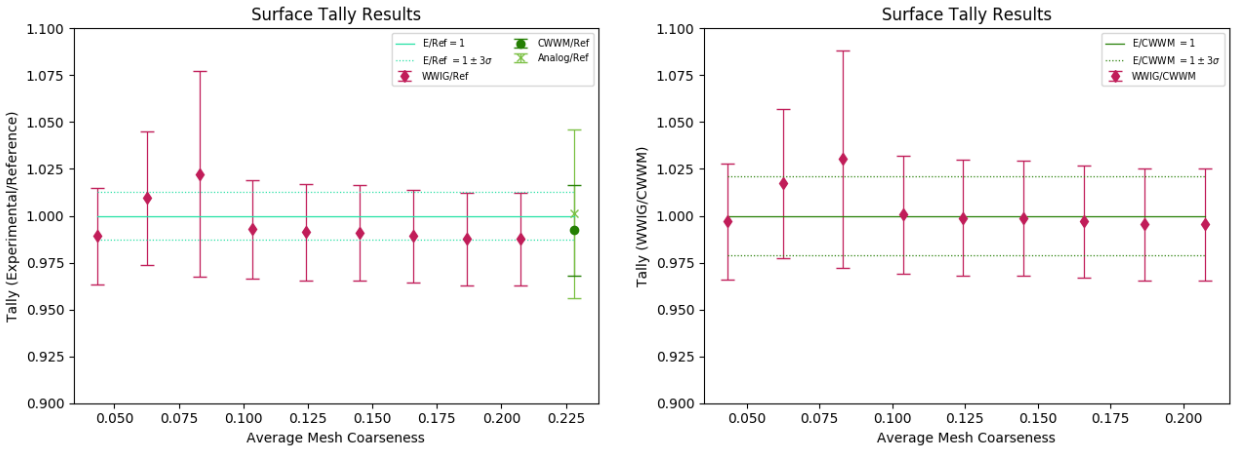


Figure 6.10: Total file size for all energy groups as a function of decimation factor.

lack of variation in those results indicates that it is possible to refine the mesh through decimation to achieve a lower memory footprint without significantly altering the outcome of the simulation. In problems where the isosurfaces follow fine, key details of the transport geometry, it may not be possible to apply as high amounts of decimation.

Table 6.2: Surface tally results for different decimation factors.

Mode		**Neutron Flux** $[1/\text{cm}^2]$	**Relative Error**	**VOV**
Reference		1.4383×10^{-5}	0.0042	0.0338
Analog		1.4401×10^{-5}	0.0144	0.1412
CWWM		1.4271×10^{-5}	0.0070	0.0691
WWIG	$d = 0.0$	1.4206×10^{-5}	0.0072	0.0026
	$d = 0.1$	1.4205×10^{-5}	0.0072	0.0026
	$d = 0.2$	1.4227×10^{-5}	0.0072	0.0027
	$d = 0.3$	1.4254×10^{-5}	0.0075	0.0061
	$d = 0.4$	1.4256×10^{-5}	0.0076	0.0075
	$d = 0.5$	1.4280×10^{-5}	0.0078	0.0119
	$d = 0.6$	1.4704×10^{-5}	0.0174	0.4586
	$d = 0.7$	1.4519×10^{-5}	0.0110	0.2274
	$d = 0.8$	1.4227×10^{-5}	0.0076	0.0062



(a) Reference result comparison *(b) CWWM result comparison*

Figure 6.11: Total neutron flux the surface tally plotted as a ratio compared the reference and CWWM results for various average mesh coarseness values. Error bars are 3σ for the ratio of the results.

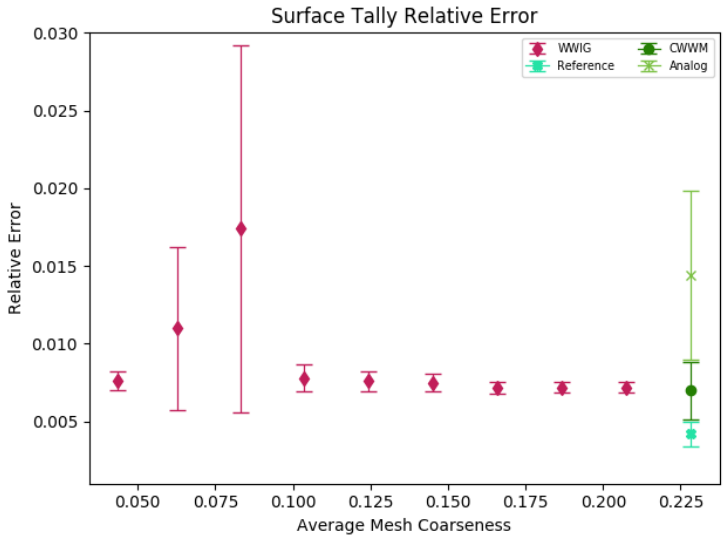
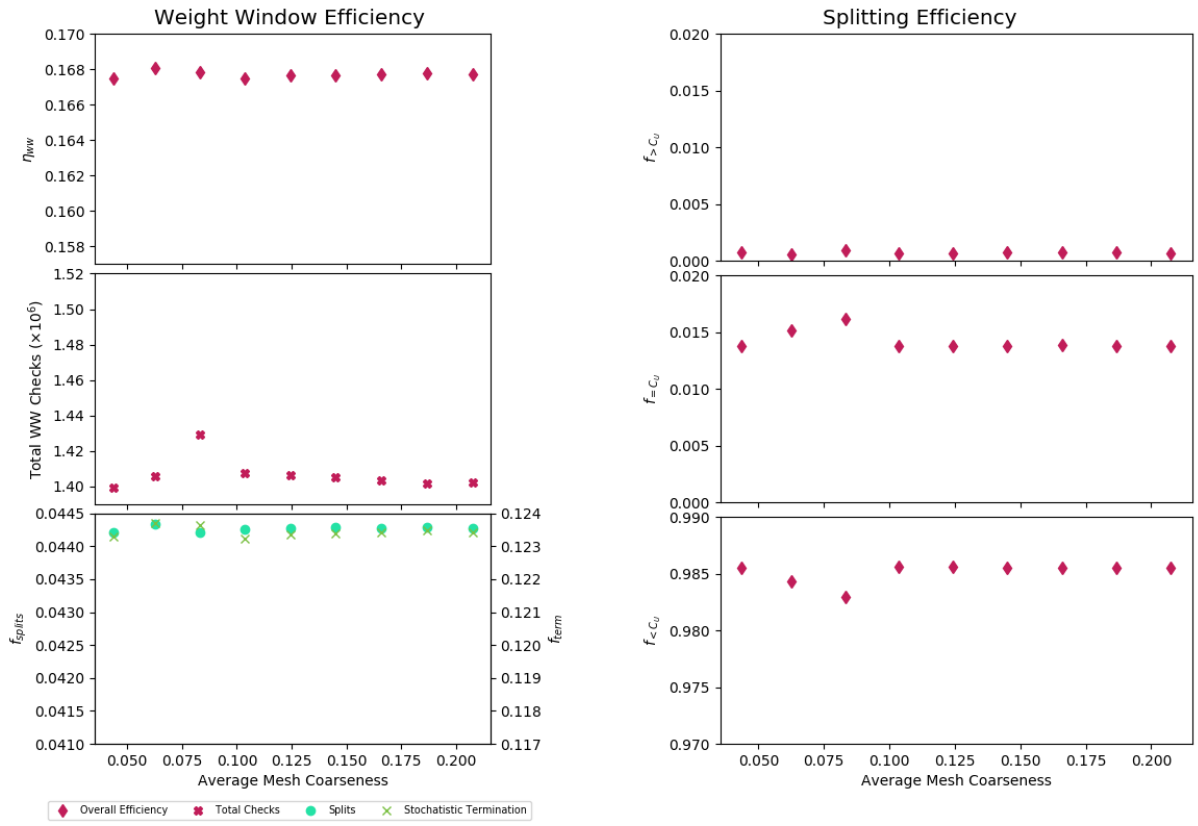


Figure 6.12: Relative error as a function of average mesh coarseness. The error bars are given by the VOV

In Figure 6.13 we see there is no major effect on the overall WW efficiency η_{ww} nor the amount of splitting or terminating as the result of mesh decimation. This lack of variation is expected as we are just altering the mesh resolution and not the geometry or surface locations.

Another key motivation, besides the memory footprint, for applying decimation to the



(a) WW check efficiency and breakdown *(b) Splitting efficiency*

Figure 6.13: WW metrics as a function of the mesh coarseness.

WWIGs is to improve ray tracing performance with DAGMC. Because DAGMC constructs an OBB tree for all facets in the geometries and must search the tree to find the facet of intersection, when a geometry has a higher number of facets, the time required to perform this task increases. We expect to see the CPU time decrease as the mesh coarseness decreases, and conversely the FOM should increase if the relative error is steady. However, we actually do not see in Figure 6.14 that such trend is the case. There appears to be no strong correlation between CPU time and the mesh coarseness, nor do we see any strong variation in the FOM as the result of this CPU time (any variation of the FOM is most likely due to the variation in relative error). In the case of the decimation for this specific test problem, the number of triangles are reduced by at most a factor of 10 (analogous to the measured coarseness because surface area is assumed to be unchanged). The depth of OBB trees are $\mathcal{O}(\log n)$, where n

is the number of triangles [31]. A factor of 10 difference in this case does not significantly impact the tree depth, and therefore does not significantly impact the time spent traversing the OBB tree during the particle tracking process either. However it should be noted that while decimation may not always increase performance, it also doesn't decrease performance. Therefore it is still a reasonable to apply decimation for the purpose of reducing the memory footprint.

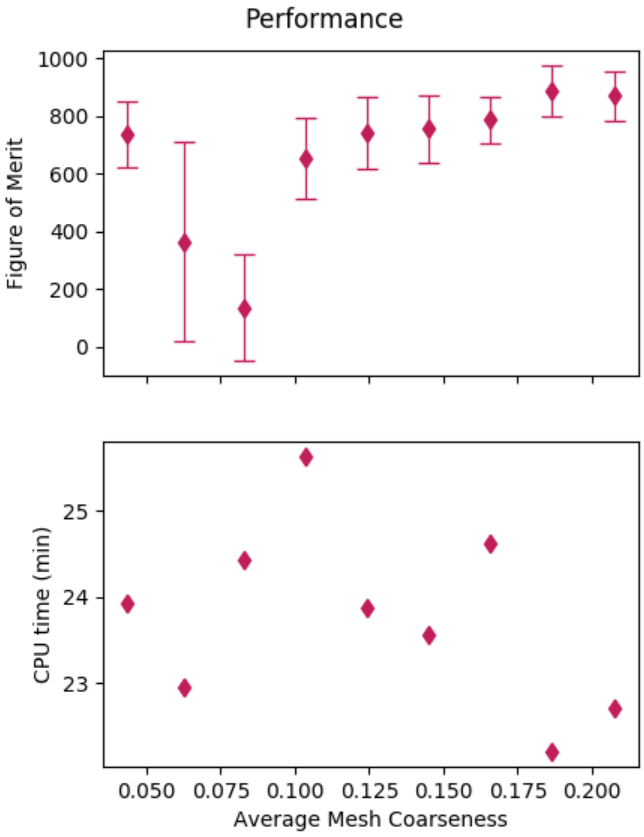


Figure 6.14: Computational performance as a function of mesh coarseness.

6.4 Surface Roughness Experiment

This experiment is meant to simulate how CWWMs with unnecessarily rough surfaces may yield less efficient WWIGs, such as is shown in Section 3.3.3. By smoothing these unneces-

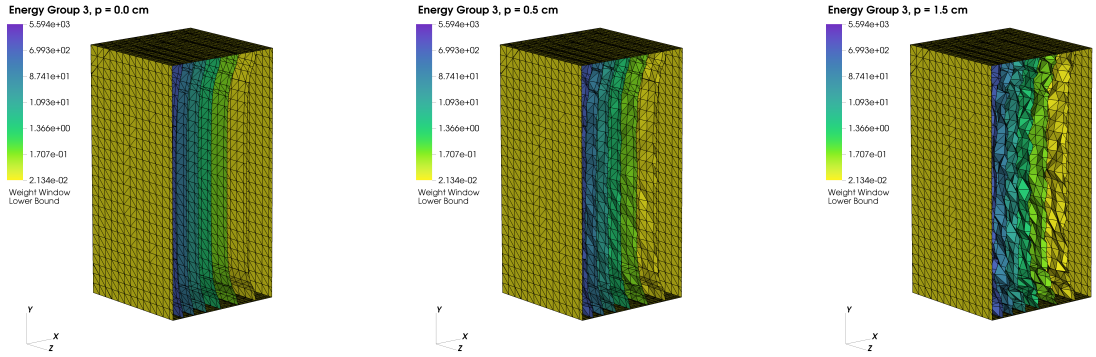
sarily rough surfaces, we expect to see an increased WW check efficiency and no penalty to performance.

6.4.1 Problem Setup

To measure the effect of surface roughness on performance, we start again with the same transport geometry and starting CWWM described in Section 6.2.1 and use the $r = 8$ surface spacing like was used in Section 6.3. To better control the surface locations and the roughness for testing, we start with the smooth, parallel surfaces from the previous setup and artificially perturb the vertices to simulate noise(as opposed to applying smoothing mesh refinement to already generated WWIGs with naturally rough surfaces). By creating the WWIGs in this artificial method, we can ensure that the only changing factor in each experiment is indeed due to only the surface roughness. The x coordinate of each of the vertices on the interior surfaces was perturbed artificially by some random amount between $-\delta$ and $+\delta$. Sixteen sets of WWIGs were generated using a δ value ranging from 0.0 cm to 1.5 cm. Example images of how these interior surfaces changed with the application of the perturbation can be seen in Figure 6.15. For images of all WWIGs for each energy group at every applied perturbation, see Appendix F. The global average roughness of only the interior surfaces[1] for each energy group geometry was calculated using Equation (5.3) and then averaged across all energy groups to get a single value for each set of WWIGs shown in Figure 6.16a. To ensure that this is artificial perturbation is representative of WWIGs that may be generated from CWWMs produced via stochastic methods, the smoothing algorithm described in Section 5.2.1 was then applied to the set of geometries with the largest perturbation ($\delta = \pm 1.5$ cm) in an attempt to recover the original smooth surfaces. The relaxation factor for smoothing was set to 0.1 and a varying number of iterations were used ranging from 2 to 10 iterations. From Figure 6.16b we see that we can begin to recover the original surface roughness values and

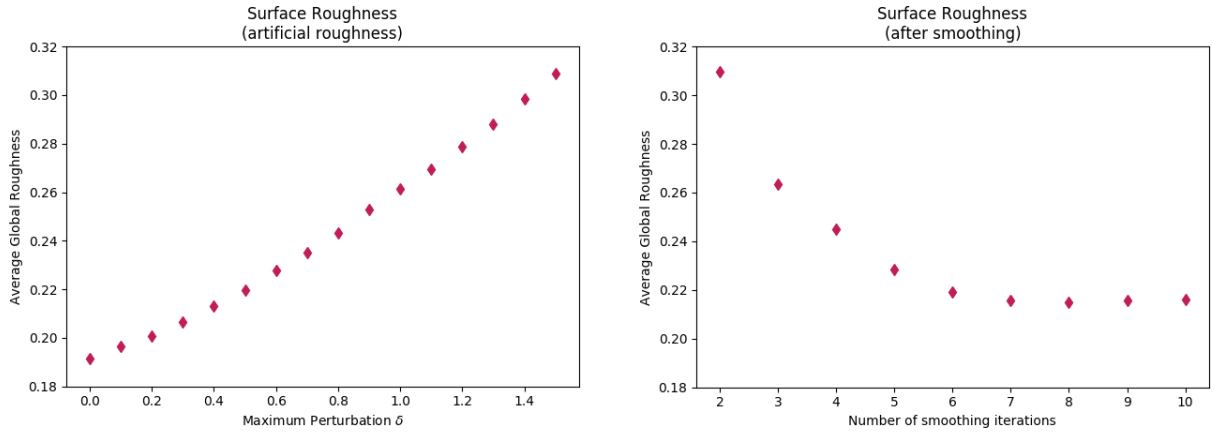
[1]In this experiment only interior surfaces were perturbed to create roughness and included in the overall roughness measurement because in a real scenario, the outer surfaces of a WWIG would still be smooth even if it were generated using stochastic methods. Therefore, we only measure the roughness of the interior surfaces.

therefore these artificially perturbed geometries can be considered an accurate representation of WWIGs from stochastically generated CWWMs.



(a) $\delta = 0.0\,\text{cm}$ *(b)* $\delta = 0.5\,\text{cm}$ *(c)* $\delta = 1.5\,\text{cm}$

Figure 6.15: The WWIG for energy group E_3 after various amounts of mesh perturbation δ have been applied to the interior surfaces. The mesh facets are shown outlined in black and a cutout at $z = 0$ shows the interior mesh.



(a) Roughness after applying varying amounts of random perturbations of the vertices.

(b) Roughness after applying smoothing to the set of geometries with $\delta = \pm 1.5\,\text{cm}$.

Figure 6.16: Average global surface roughness measured across all interior surfaces of all energy groups.

6.4.2 Results and Analysis

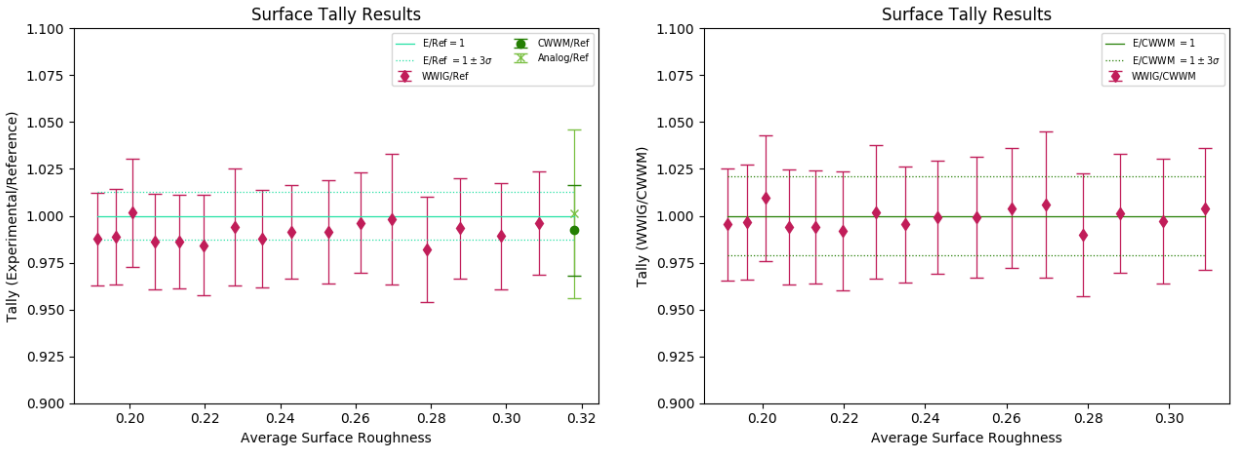
Similar to each of the last experiments, all sets of WWIGs were run with 10^5 histories and the results can be seen in Table 6.3 and Figure 6.17. Again we see that the results agree fairly

well with the CWWM and reference results. In Figure 6.18 we see that there is significantly higher VOV in most cases and more variation in the relative error which indicates that noisy surfaces can cause some difficulty converging on an answer. However, it should be pointed out that the VOV is still on the same order of magnitude and in most cases lower than that of the CWWM run.

Table 6.3: Surface tally results for different roughness perturbations

Mode		**Neutron Flux** $[1/\text{cm}^2]$	**Relative Error**	**VOV**
Reference		1.4383×10^{-5}	0.0042	0.0338
Analog		1.4401×10^{-5}	0.0144	0.1412
CWWM		1.4271×10^{-5}	0.0070	0.0691
WWIG	$\delta = 0.0$	1.4206×10^{-5}	0.0072	0.0026
	$\delta = 0.1$	1.4220×10^{-5}	0.0075	0.0064
	$\delta = 0.2$	1.4406×10^{-5}	0.0086	0.0342
	$\delta = 0.3$	1.4187×10^{-5}	0.0075	0.0093
	$\delta = 0.4$	1.4185×10^{-5}	0.0073	0.0041
	$\delta = 0.5$	1.4158×10^{-5}	0.0080	0.0320
	$\delta = 0.6$	1.4299×10^{-5}	0.0096	0.1239
	$\delta = 0.7$	1.4205×10^{-5}	0.0077	0.0221
	$\delta = 0.8$	1.4261×10^{-5}	0.0073	0.0039
	$\delta = 0.9$	1.4261×10^{-5}	0.0082	0.0158
	$\delta = 1.0$	1.4330×10^{-5}	0.0079	0.0074
	$\delta = 1.1$	1.4356×10^{-5}	0.0109	0.2272
	$\delta = 1.2$	1.4124×10^{-5}	0.0085	0.1065
	$\delta = 1.3$	1.4287×10^{-5}	0.0079	0.0138
	$\delta = 1.4$	1.4229×10^{-5}	0.0086	0.0900
	$\delta = 1.5$	1.4325×10^{-5}	0.0082	0.0191

Where we expect the surface roughness to have the most impact is on the WW efficiency, and consequently the CPU time and FOM. In Figure 6.19a there is a clear, though small, trend of decreased efficiency as the roughness increases. While we see this decrease in efficiency, the overall WW efficiency is still much higher than that of the CWWM mode as



(a) Reference result comparison *(b) CWWM result comparison*

Figure 6.17: Total neutron flux the surface tally plotted as a ratio compared the reference and CWWM results for various average surface roughness values. Error bars are 3σ for the ratio of the results.

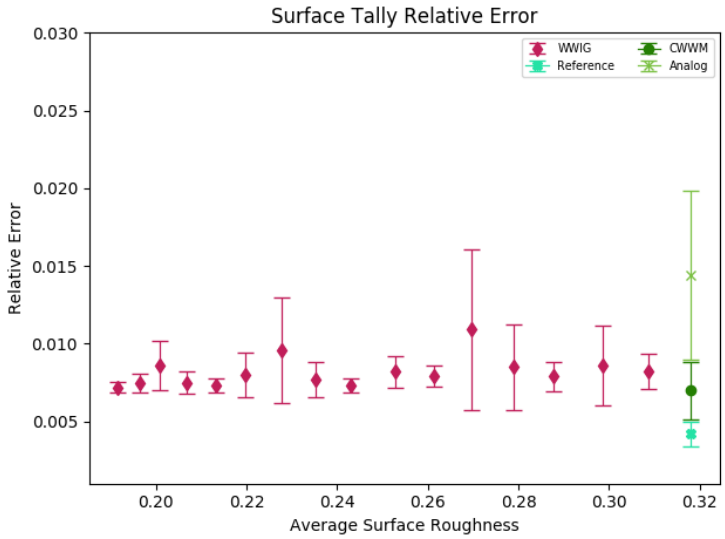
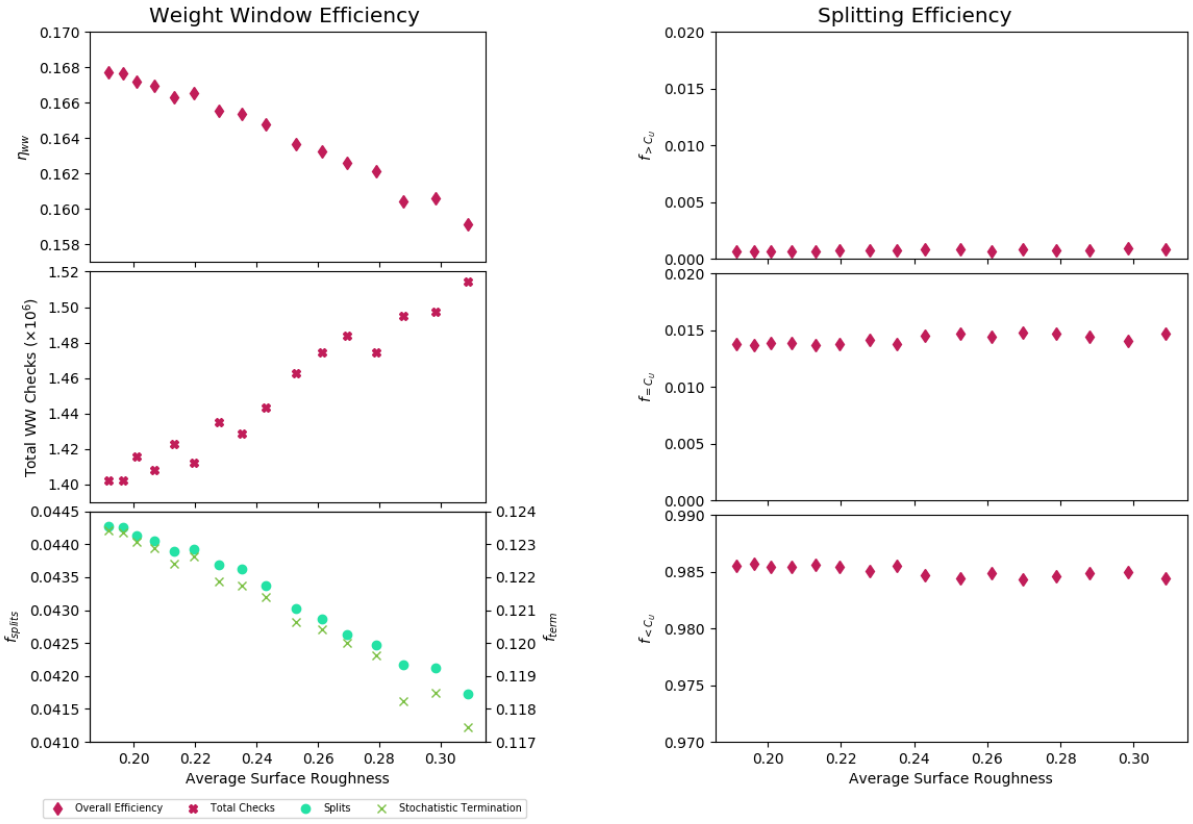


Figure 6.18: Relative error as a function of average mesh roughness. The error bars are given by the VOV

noted in Section 6.2.2. Nonetheless, this trend is still expected and worth discussing for further understanding. When a particle traverses a rough WWIG surface (as opposed to the coordinating smooth surface as indicated in Figure 5.1), it experiences more surface crossings, and therefore undergoes more WW checks. Because these checks are now more likely occurring before any additional collisions and the WW value for the surface is not changing,

the particle is less likely to experience additional splitting or terminating at the rough surface crossings. We can confirm this by the increase in the total number of WW checks and steady decrease in splitting and terminating shown in Figure 6.19a. In Figure 6.19b we see that the breakdown of splitting doesn't vary strongly as a function of roughness and remains relatively constant which is expected.



(a) WW check efficiency and breakdown *(b) Splitting efficiency*

Figure 6.19: WW metrics as a function of the surface roughness.

In Figure 6.20 we can see a slight increase, apart from an outlier for unknown reasons[2], in CPU time required as the surface roughness increases. This trend is expected due to the increased number of WW checks and surface crossings. However, there is no clear trend in the FOM again due to the variability of the relative error and VOV for each set of WWIGs.

[2]Based this high CPU time, we hypothesize that this is due to an unintentional long history. However it was not investigated further.

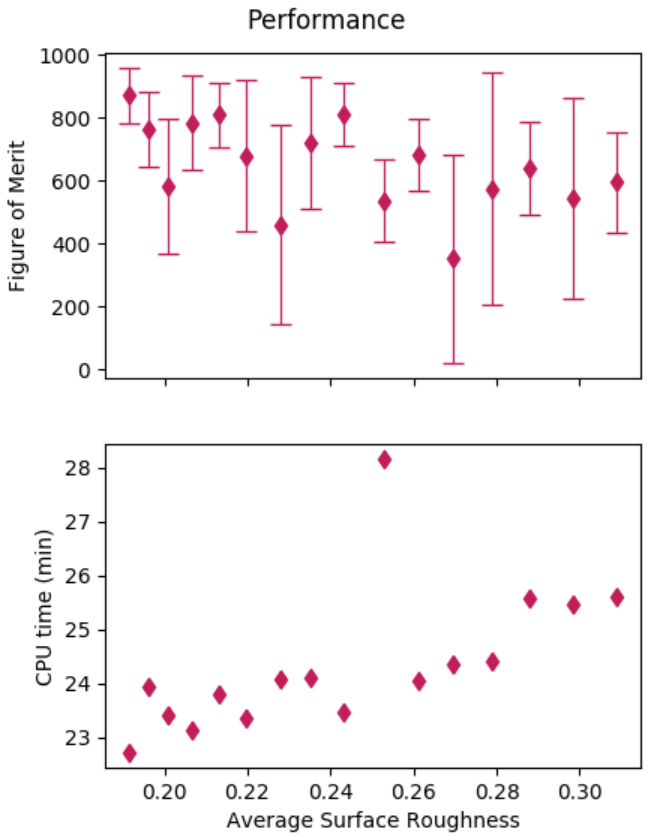


Figure 6.20: Computational performance as a function of average surface roughness.

6.5 Performance Analysis Conclusions

In this chapter we analyzed how various mesh features (surface spacing, mesh coarseness, and surface roughness) of WWIGs affect overall performance during MC VR using DAG-MCNP 6.2. Adjusting the spacing of the isosurfaces was not found to have a significant impact on the VR quality, though there is a peak WW efficiency for separation ratios just below the upper WW constant. However, this efficiency variability is minimal compared to the efficiency of the corresponding CWWM and does not appear to impact the overall FOM. We do see, however, a significant decrease in the computational time and memory footprint as the WW surface spacing increases, meaning that if one is most concerned with the memory footprint or CPU time, it would be advantageous to use a higher separation ratio than one that offers peak WW efficiency. Memory footprint can be further reduced without affecting necessarily

performance or VR quality by applying mesh decimation, allowing it to be competitive with the memory footprint of CWWMs. We also demonstrated that unnecessarily noisy or rough surfaces can decrease WW efficiency and increase the CPU time required, though it does not significantly impact the VR quality.

Chapter 7

Summary and Conclusions

In this dissertation a novel method for representing and using WWs as DAGMC-compliant mesh geometries during MC particle tracking was presented. The WWIGs, which can be automatically generated from existing CWWMs using the IsogeomGenerator tool [36], are complex geometries whose surfaces are derived from isosurfaces of WW values in the CWWM. The internal surfaces of the WWIGs are each defined by a single WW lower bound value, which are used as the WWs applied during MC particle tracking.

The particle tracking algorithm has been successfully implemented in DAG-MCNP 6.2 [19, 6, 39]. In this algorithm particle weight is checked only when a particle crosses a WWIG surface rather than at each collision, transport geometry crossing, and mean free path traveled, which is the current default implementation when using CWWMs in MCNP. This change in the particle tracking algorithm is meant to improve WW efficiency by only by only applying WWs when the particle is expected to be affected by a weight change. In the demonstration experiment, the WWIG method proved to accurately perform VR and is comparable, if not sometimes better, than the corresponding CWWM in terms of VR.

Beyond this, geometric properties of the WWIGs, including isosurface spacing, mesh coarseness, and surface roughness, were analyzed to understand how computational performance is affected. We found that WWIGs whose surface spacing ratio is close to the WW upper bound constant C_U performed best in terms of WW efficiency, though even significantly larger spacing still performed accurate and efficient VR with a significantly reduced memory footprint. With the decimation of WWIG surface meshes, we saw a desirable reduction in the memory footprint while maintaining accurate VR and no loss of WW efficiency.

And finally, in analyzing how surface roughness affects performance, we found that surface roughness does reduce WW efficiency and slightly affects the computational time. With all of these geometric features, any change in WW efficiency or performance as a result of changing the geometry or mesh characteristics was minor in comparison to the low WW efficiency of the corresponding CWWM.

In conclusion, WWIGs have been demonstrated to provide similar reduction in variance compared to CWWMs, though further study is warranted to determine the problem configurations that will yield an overall performance benefit, after the implementation has been computationally optimized. A hypothesis for fully realizing the benefits of using WWIGs is to start with a CWWM with significantly higher mesh resolution than would normally be used given available computational resources for typical MC simulations. Creating WWIGs from this finer mesh would likely capture fine geometric details in the isosurfaces that would not otherwise be seen with a lower resolution CWWM. Applying decimation to the resulting WWIGs in this case would mean there could be both the benefit of improved resolution for fine geometric detail as well as the benefit of a reduced memory footprint compared to the equivalent CWWM, regardless of surface spacing.

Chapter 8

Future Work

Although this work has already demonstrated how mesh refinement methods can improve MC VR performance with WWIGs, there is opportunity for even more improvement. Currently a single WWIG file is required for each WW energy group. However, the isosurfaces in each of the energy groups are generally very close in physical space and follow the same patterns across several energy groups. This would make it possible to instead select a few sets of isosurfaces that could represent all the energy groups in just a handful of WWIGs (reducing G in Equation (6.9)). Each surface of these representative WWIGs would then have the WW defined for each of the energy groups on every surface. Fewer files would mean a further reduction in memory footprint. This could make the WWIG method extremely competitive to using CWWMs, especially so if mesh refinement is used. Additionally, further investigation should be done with using WWIGs on more complex transport systems to understand the effect on performance and if mesh refinement techniques can be adequately applied.

Additional potential future work is related to the implementation. Because this method is novel, the details of the implementation are not necessarily setup to take advantage of all novel computing and coding techniques that could improve computational efficiency. The implementation could potentially be reworked to take advantage of methods that further reduce computational time, therefore increasing FOM. This method could also be implemented in other MC physics codes that are already compatible with DAGMC, such as OpenMC [25, 26] and Shift [27].

Bibliography

[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, version 5 ed. (2008).

[2] I. Lux and L. Koblinger, *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*, CRC Press (2018); 10.1201/9781351074834., URL https://www.taylorfrancis.com/books/9781351083287.

[3] T. M. Jenkins, W. R. Nelson, and A. Rindi, *Monte Carlo Transport of Electrons and Photons*, vol. 38 of *Ettore Majorana International Science Series*, Plenum Press, New York and London (1987).

[4] O. N. Vassiliev, *Monte Carlo Methods for Radiation Transport*, Biological and Medical Physics, Biomedical Engineering, Springer International Publishing, Cham (2017); 10.1007/978-3-319-44141-2., URL http://link.springer.com/10.1007/978-3-319-44141-2.

[5] A. Haghighat and J. C. Wagner, "Monte Carlo variance reduction with deterministic importance functions," *Progress in Nuclear Energy*, **42**, *1*, 25 (2003); 10.1016/S0149-1970(02)00002-1., URL http://www.sciencedirect.com/science/article/pii/S0149197002000021.

[6] J. Armstrong, F. B. Brown, J. S. Bull, L. Casswell, L. J. Cox, D. Dixon, R. A. Forster, J. T. Goorley, H. G. Hughes, J. Favorite, R. Martz, S. G. Mashnik, M. E. Rising, C. Solomon, A. Sood, J. E. Sweezy, C. J. Werner, A. Zukaitis, C. Anderson, J. S. Elson, J. W. Durkee, R. C. Johns, G. W. McKinney, G. E. McMath, J. S. Hendricks, D. B. Pelowitz, R. E. Prael, T. E. Booth, M. R. James, M. L. Fensin, T. A. Wilcox, and B. C. Kiedrowski, *MCNP User's Manual: Code Version 6.2*, Los Alamos National Laboratory, manual rev. 0 ed. (2017).

[7] J. C. Wagner and A. Haghighat, "Automated Variance Reduction of Monte Carlo Shielding Calculations Using the Discrete Ordinates Adjoint Function," *Nuclear Science and Engineering*, **128**, *2*, 186 (1998); 10.13182/NSE98-2., URL https://doi.org/10.13182/NSE98-2.

[8] S. Mosher, S. Johnson, A. Bevill, A. Ibrahim, C. Daily, T. Evans, W. J.C., J. Johnson, and R. Grove, *ADVANTG 3.0.3: AutomateD VAriaNce reducTion Generator*, Oak Ridge National Laboratory, Oak Ridge, TN, 3rd ed. (2015).

[9] C. Bates, E. Biondo, K. Huff, K. Kiesling, R. Carlsen, A. Davis, M. Gidden, T. Haines, J. Howland, B. Huff, K. Manalo, A. Opotowsky, R. Slaybaugh, E. Relson, P. Romano, P. Shriwise, J. D. Xia, P. Wilson, and J. Zachman,

"PyNE Progress Report," *Am. Nuc. Soc. Winter Meeting 2014*, vol. 111, Anaheim, CA, USA (2014).

[10] THE PYNE DEVELOPMENT TEAM, "PyNE: The Nuclear Engineering Toolkit," (2014)URL http://pyne.io.

[11] R. E. ALCOUFFE, R. S. BAKER, J. A. DAHL, E. J. DAVIS, T. G. SALLER, S. A. TURNER, R. C. WARD, and R. J. ZERR, *PARTISN: A Time-Dependent, Parallel Neutral Particle Transport Code System*, Los Alamos National Laboratory, version 8.29 ed. (2018).

[12] 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*, **176**, *1*, 37 (2014); 10.13182/NSE12-33., URL https://doi.org/10.13182/NSE12-33.

[13] A. M. IBRAHIM, D. E. PEPLOW, R. E. GROVE, J. L. PETERSON, and S. R. JOHNSON, "The Multi-Step CADIS Method for Shutdown Dose Rate Calculations and Uncertainty Propagation," *Nuclear Technology*, **192**, *3*, 286 (2015).

[14] E. D. BIONDO and P. P. WILSON, "Transmutation Approximations for the Application of Hybrid Monte Carlo/Deterministic Neutron Transport to Shutdown Dose Rate Analysis," *Nuclear Science and Engineering*, **187**, *2*, 27 (2017).

[15] M. MUNK, R. SLAYBAUGH, T. PANDYA, S. JOHNSON, and T. EVANS, "FW/CADIS-Ω: An angle-informed hybrid method for deep-penetration radiation transport," *Physics of Reactors 2016, PHYSOR 2016: Unifying Theory and Experiments in the 21st Century*, **2**, 1069 (2016).

[16] A. DAVIS and A. TURNER, "Comparison of global variance reduction techniques for Monte Carlo radiation transport simulations of ITER," *Fusion Engineering and Design*, **86**, *9*, 2698 (2011); 10.1016/j.fusengdes.2011.01.059., URL http://www.sciencedirect.com/science/article/pii/S0920379611000718.

[17] A. M. IBRAHIM, P. P. H. WILSON, M. E. SAWAN, S. W. MOSHER, D. E. PEPLOW, J. C. WAGNER, T. M. EVANS, and R. E. GROVE, "Automatic Mesh Adaptivity for Hybrid Monte Carlo/Deterministic Neutronics Modeling of Difficult Shielding Problems," *Nuclear Science and Engineering*, **181**, *1*, 48 (2015); dx.doi.org/10.13182/NSE14-94., URL http://epubs.ans.org/?a=37494.

[18] A. M. IBRAHIM, P. P. WILSON, M. E. SAWAN, S. W. MOSHER, D. E. PEPLOW, and R. E. GROVE, "Assessment of fusion facility dose rate map using mesh adaptivity enhancements of hybrid Monte Carlo/deterministic techniques," *Fusion Engineering and Design*, **89**, *910*, 1875 (2014); 10.1016/j.fusengdes.2014.02.046., URL http://www.sciencedirect.com/science/article/pii/S0920379614001434.

[19] T. J. TAUTGES, P. P. WILSON, J. A. KRAFTCHECK, B. M. SMITH, and D. L. HENDERSON, "Acceleration Techniques For Direct Use of CAD-Based Geometries In Monte

Carlo Radiation Transport," *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)*, American Nuclear Society, Saratoga Springs, NY (2009).

[20] V. S. Mahadevan, T. J. Tautges, I. Grindeanu, R. Jain, D. Wu, N. Ray, J. Hu, P. Wilson, P. Shriwise, and A. Scopatz, *MOAB: Mesh Oriented datABase*, 5th ed.

[21] "DAGMC: Direct Accelerated Geometry Monte Carlo," http://svalinn.github.io/DAGMC/URL `http://svalinn.github.io/DAGMC/`.

[22] R. Morris and C. McBride, *Trelis 16.5 User Documentation*, csimsoft (2018).

[23] B. M. Smith, "Robust Tracking and Advanced Geometry for Monte Carlo Radiation Transport," PhD Thesis, University of Wisconsin-Madison, Madison, Wisconsin (2011).

[24] B. Smith, T. Tautges, and P. Wilson, "Sealing Faceted Surfaces to Achieve Watertight CAD Models," *19th International Meshing Roundtable*, Springer, Chattanooga, TN (2010).

[25] P. K. Romano and B. Forget, "The OpenMC Monte Carlo particle transport code," *Annals of Nuclear Energy*, **51**, 274 (2013); 10.1016/j.anucene.2012.06.040., URL `https://www.sciencedirect.com/science/article/pii/S0306454912003283`.

[26] 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*, **82**, 90 (2015); 10.1016/j.anucene.2014.07.048., URL `http://www.sciencedirect.com/science/article/pii/S030645491400379X`.

[27] T. M. Pandya, S. R. Johnson, G. G. Davidson, T. M. Evans, and S. P. Hamilton, "Shift: A Massively Parallel Monte Carlo Radiation Transport Package," , Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States). Oak Ridge Leadership Computing Facility (OLCF); Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States). Consortium for Advanced Simulation of LWRs (CASL) (2015)URL `https://www.osti.gov/biblio/1185847`.

[28] G. Battistoni, T. Boehlen, F. Cerutti, P. W. Chin, L. S. Esposito, A. Fass, A. Ferrari, A. Lechner, A. Empl, A. Mairani, A. Mereghetti, P. G. Ortega, J. Ranft, S. Roesler, P. R. Sala, V. Vlachoudis, and G. Smirnov, "Overview of the FLUKA code," *Annals of Nuclear Energy*, **82**, 10 (2015); 10.1016/j.anucene.2014.11.007., URL `https://www.sciencedirect.com/science/article/pii/S0306454914005878`.

[29] S. Guatelli, D. Cutajar, B. Oborn, and A. B. Rosenfeld, "Introduction to the Geant4 Simulation toolkit," *AIP Conference Proceedings*, **1345**, *1*, 303 (2011); 10.1063/1.3576174., URL `https://aip.scitation.org/doi/abs/10.1063/1.3576174`.

[30] E. Brun, F. Damian, E. Dumonteil, F.-X. Hugot, Y.-K. Lee, F. Malvagi, A. Mazzolo, O. Petit, J.-C. Trama, T. Visonneau, and A. Zoia, "TRIPOLI-4R Version 8 User Guide," , France (2013)CEA-R–6316 INIS Reference Number: 44060240.

[31] Arvo, James and Cook, Robert L. and Glassner, Andrew and Haines, Eric and Hanrahan, Pat and Heckbert, Paul S., and Kirk, David, *An Introduction to Ray Tracing*, vol. v1.3, Academic Press, London (1989).

[32] P. C. Shriwise and P. P. Wilson, "Reduced Precision Ray Tracing Performance Enhancements in the DAGMC Toolkit," *2018 ANS Winter Meeting*, vol. 119, 608–611, American Nuclear Society, Orlando, FL (2018).

[33] P. C. Shriwise, A. Davis, L. J. Jacobson, and P. P. Wilson, "Particle tracking acceleration via signed distance fields in direct-accelerated geometry Monte Carlo," *Nuclear Engineering and Technology*, **49**, 1189 (2017).

[34] P. C. Shriwise, "Geometry Query Optimizations in CAD-based Tessellations for Monte Carlo Radiation Transport," PhD Thesis, University of Wisconsin-Madison, Madison, Wisconsin (2018).

[35] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil, "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data," *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, 357–372.

[36] K. R. Kiesling and B. Mouginot, "IsogeomGenerator," `https://github.com/svalinn/IsogeomGenerator.git` (2021).

[37] K. Kiesling and P. P. H. Wilson, "Generation of Weight Window Isosurface Geometries for Monte-Carlo Variance Reduction," vol. 119, 586–589, American Nuclear Society, Orlando, FL (2018).

[38] E. S. Gonzalez and G. G. Davidson, "Choosing Transport Events for Initiating Splitting and Rouletting," *Journal of Nuclear Engineering*, **2**, *2*, 97 (2021); 10.3390/jne2020010., URL `https://www.mdpi.com/2673-4362/2/2/10`.

[39] K. R. Kiesling and P. P. Wilson, "Preliminary Results for Particle Tracking on Weight Window Isosurface Geometries for Monte Carlo Variance Reduction," *ANS Winter Meeting 2019 Transactions*, vol. 121, 755–758, American Nuclear Society, Washington, D.C. (2019).

[40] R. McConn, Jr., C. Gesh, R. Pagh, R. Rucker, and R. Williams, III, "Compendium of Material Composition Data for Radiation Transport Modeling," PIET-43741-TM963 PNNL-15870 Rev. 1, U.S. Department of Homeland Security; U.S. Department of Energy (DOE) Pacific Northwest National Laboratory (PNNL), Richland, WA, United States (2011).

[41] W. Schroeder, K. Martin, and B. Lorensen, *The visualization toolkit: an object-oriented approach to 3D graphics ; visualize data in 3D - medical, engineering or scientific ; build your own applications with C++, Tcl, Java or Python ; includes source code for VTK (supports Unix, Windows and Mac)*, Kitware, Inc, Clifton Park, NY (2006).

[42] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Computers & Graphics*, **22**, *1*, 37 (1998); 10.1016/S0097-8493(97)00082-4., URL http://www.sciencedirect.com/science/article/pii/S0097849397000824.

[43] R. Bade, J. Haase, and B. Preim, "Comparison of fundamental mesh smoothing algorithms for medical surface models. SimVis06," *In Simulation und Visualisierung (2006*, 289–304 (2006).

[44] R. Biswas and R. C. Strawn, "Tetrahedral and hexahedral mesh adaptation for CFD problems," *Applied Numerical Mathematics*, **26**, *1*, 135 (1998); 10.1016/S0168-9274(97)00092-5., URL http://www.sciencedirect.com/science/article/pii/S0168927497000925.

[45] W. Schroeder, J. Zarge, and W. Lorensen, "Decimation of triangle meshes," *SIGGRAPH Comput. Graph.*, **26**, 65 (1997); 10.1145/133994.134010.

[46] L. Kobbelt, S. Campagna, and H.-p. Seidel, "A general framework for mesh decimation," *in Proceedings of Graphics Interface*, 43–50 (1998).

[47] J. Wu and L. Kobbelt, "Fast Mesh Decimation by Multiple-Choice Techniques," (2002).

[48] J. F. Shepherd, M. W. Dewey, A. C. Woodbury, S. E. Benzley, M. L. Staten, and S. J. Owen, "Adaptive mesh coarsening for quadrilateral and hexahedral meshes," *Finite Elements in Analysis and Design*, **46**, *1*, 17 (2010); 10.1016/j.finel.2009.06.024., URL http://www.sciencedirect.com/science/article/pii/S0168874X09000791.

[49] "VTK: vtkDecimatePro Class Reference," URL https://vtk.org/doc/nightly/html/classvtkDecimatePro.html.

[50] Y. Qin, K. R. Kiesling, E. Pickhardt, P. P. Wilson, and C. Moreno, "dagmc_stats," https://github.com/svalinn/dagmc_stats.git (2021).

[51] G. Taubin, T. Zhang, and G. Golub, "Optimal Surface Smoothing as Filter Design," (1996); 10.1007/BFb0015544.

[52] J. Wang, Y. Li, Y. Choi, C. Lee, and J. Kim, "Fast and Accurate Smoothing Method Using A Modified AllenCahn Equation," *Computer-Aided Design*, **120**, 102804 (2020); 10.1016/j.cad.2019.102804., URL http://www.sciencedirect.com/science/article/pii/S0010448519305366.

[53] P. Alliez, M. Meyer, and M. Desbrun, "Interactive Geometry Remeshing," *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, 347–354, ACM, New York, NY, USA (2002); 10.1145/566570.566588., URL http://doi.acm.org/10.1145/566570.566588, event-place: San Antonio, Texas.

[54] Y. Ohtake, A. G. Belyaev, and I. A. Bogaevski, "Polyhedral surface smoothing with simultaneous mesh regularization," *Proceedings Geometric Modeling and Processing 2000. Theory and Applications*, 229–237 (2000); 10.1109/GMAP.2000.838255.

[55] S. A. Canann, M. B. Stephenson, and T. Blacker, "Optismoothing: An optimization-driven approach to mesh smoothing," *Finite Elements in Analysis and Design*, **13**, *2*, 185 (1993); 10.1016/0168-874X(93)90056-V., URL http://www.sciencedirect.com/science/article/pii/0168874X9390056V.

[56] N. Amenta, M. Bern, and D. Eppstein, "Optimal Point Placement for Mesh Smoothing," *Journal of Algorithms*, **30**, *2*, 302 (1999); 10.1006/jagm.1998.0984., URL http://www.sciencedirect.com/science/article/pii/S0196677498909841.

[57] L. A. Freitag, "On combining Laplacian and optimization-based mesh smoothing techniques," ANL/MCS-P-645-0297; CONF-9706106-2, Argonne National Lab., IL (United States) (1997)URL https://www.osti.gov/biblio/505716.

[58] J. Vollmer, R. Mencl, and H. Mller, "Improved Laplacian Smoothing of Noisy Surface Meshes," *Computer Graphics Forum*, **18**, *3*, 131 (1999); 10.1111/1467-8659.00334., URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00334.

[59] "VTK: vtkWindowedSincPolyDataFilter Class Reference," URL https://vtk.org/doc/nightly/html/classvtkWindowedSincPolyDataFilter.html.

[60] N. Moreau, C. Roudet, and C. Gentil, "Study and Comparison of Surface Roughness Measurements," *Journes du Groupe de Travail en Modlisation Gomtrique (GTMG 2014)*, Lyon, France (2014)URL https://hal.archives-ouvertes.fr/hal-01068988.

[61] G. Lavou, "A roughness measure for 3D mesh visual masking," *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, APGV '07, 57–60, Association for Computing Machinery, New York, NY, USA (2007); 10.1145/1272582.1272593., URL https://doi.org/10.1145/1272582.1272593.

[62] K. Wang, F. Torkhani, and A. Montanvert, "A fast roughness-based approach to the assessment of 3D mesh visual quality," *Computers & Graphics*, **36**, *7*, 808 (2012); 10.1016/j.cag.2012.06.004., URL http://www.sciencedirect.com/science/article/pii/S0097849312001203.