Final Report

Arnika Chidambaram Caroline Hughes Mitch Negus

December 14, 2016

Table of Contents

1 Introduction

The purpose of this project is to study variance reduction methods in Monte Carlo using deterministic methods.The goal is to create an importance map for the Monte Carlo game and selecting parameters is an important step in deterministic calculations. After parameters are selected, a map that accelerates the Monte Carlo calculation while simultaneously lowering deterministic cost, is created. Deterministic methods provide fast solutions to the transport equation, and require relatively simple inputs. Currently there is high demand for a reliable and efficient transport methods that can be used to solve complex fixed source problems[1].

Two variance reduction methods that use deterministic methods are Consistent Adjoint Driven Importance Sampling (CADIS) and Forward Weighted-Consistent Adjoint Driven Importance Sampling (FW-CADIS). CADIS optimizes specific, local response functions, where as FW-CADIS optimizes global response functions of this work.

Throughout the course of this report, the mathematics behind variance reduction and the adjoint problem will be explained. After that, an overview of CADIS AND FW-CADIS will be given, and then the algorithms for ADVANTG and MCNP inputs and outputs will be provided. Finally, seven test cases will be run using CADIS and FW-CADIS and results will be plotted in VisIt.

2 Mathematics

2.1 Variance Reduction

Monte Carlo simulations have the unique advantage of being continuous and entirely free from the effects of discretization choices that are present in deterministic simulations; however, they tend to be much more computationally expensive. In analog Monte Carlo simulations, enormous numbers of particle histories must often be tracked to produce ample statistics for meaningful results.

To meet these criteria and be considered meaningful, data must sufficiently cover the solution space and have a low statistical uncertainty, represented by a small variance. The variance (the square of the standard deviation, σ) is defined as the squared sum of the deviation of each datapoint from the mean, $\bar{x}$, divided by the number of degrees of freedom in the problem (one less than the totlal number of datapoints) [2]:

$$\sigma^2 = \frac{\sum_i^N (x_i - \bar{x})^2}{N-1},$$

where N is the number of particle histories included in the calculation. Since the particle histories (and therefore x_i) are independent, the central limit theorem applies. It follows that the standard deviation of the mean, $S_{\bar{x}}$, decreases proportionally to $1/\sqrt{N}$. For comparative purposes, this measure is usually represented by the relative error, R, where

$$R \equiv \frac{S_{\bar{x}}}{\bar{x}} = \frac{\sigma^2}{\bar{x}\sqrt{N}}.$$

More particles lead directly to a reduced uncertainty.

While simulating more particles is beneficial for reducing uncertainty in a simulation, it unsurprisingly increases the time needed for simulation. In general, the efficiency of a simulation is given by its figure of merit (FOM), defined as

$$\text{FOM} \equiv \frac{1}{R^2 T}.$$

R is the relative error and T is the computer time taken for the simulation [3]. Assuming independent particles, the run time T is directly proportional to the number of particles: $T \propto N$. When the dependencies of both R and T on N are considered, the FOM becomes constant and no substantial gains are made in terms of performance for more particles ($R^2 T \propto (1/\sqrt{N})^2 N = const.$).

Specific variance reduction methods attempt to circumvent this dilemma using a variety of clever techniques. Often, these methods emphasize simulations in regions of the problem space that have the strongest contributions to the final solution, gleaning deeper insights without wasting as much computation as analog Monte Carlo simulations. Examples include truncation methods, population control methods, modified sampling methods, and partially-deterministic methods. For maximal effect, several of these methods may be employed together in a given simulation. Some of these methods are described below.

Truncation methods are undoubtedly the most common variance reduction technique, whereby the problem space is restricted in some dimension (technically, truncation methods are employed in almost every Monte Carlo simulation to some degree, as finite boundaries must exist on space, particle energy, time, etc.). Truncation methods speed up calculations by avoiding simulation of parts of a problem that will not contribute to the solution. While easy to incorporate into a simulation, care must be taken to ensure that these parts of the problem space actually *will not*, as opposed to just *should not*, contribute to the solution.

Population control methods allow more flexibility in sampling areas of importance in the phase space, and can be separated into two categories: splitting and rouletting. In both cases, particles are assigned weights that are adjusted depending on the particle's importance to the final solution. Splitting takes place when a particle enters an area of higher importance, since it is most important that the variance be low in this region. By dividing the particle into multiple copies, the total number of particle histories in the simulation, N, can be artificially inflated, and the relative error reduced. In order to preserve the initial particle's weight, w, on the final solution, the new n split copies are scaled appropriately by a factor of w/n.

$$\sum_{i}^{n} \frac{w}{n} = n\left(\frac{w}{n}\right) = w$$

Conversely, if a particle travels into a region of lower importance, the particle will undergo a "game" of Russian roulette, with a survival probability of $1/n$. In this way, we reduce the number of particles that must be tracked, thereby reducing the time T of the simulation. The relative error of the desired solution will experience less impact since these rouletted particles are in areas of low importance and less likely to contribute to that solution. Any particle that survives will have it's initial weight, w, increased by a factor of n, so that on average the total weights are conserved.

In addition to the splitting and rouletting routines, weight window restrictions are often placed on the simulation so that weights do not become too large or too small. Particles with large weights will be split to prevent any one particle from having too much effect on the final solution, while particles with small weights will be rouletted to prevent the simulation from wasting time tracking essentially meaningless histories.

Population control methods influence the number of particles based on where they travel, weighting them accordingly. Similarly, modified sampling methods use adjusted probability distributions to control the number of particles generated with a given set of initial conditions and weight those particles accordingly. If a given situation ought to occur with probability P, a modified sample would generate that same situation

with probability P', but with weight $w_{\text{new}} = P/P'$. The overall probability is preserved:

$$P = w_{new} \cdot P' = \frac{P}{P'} P' \tag{1}$$

but the number of particles is adjusted to more effectively contribute to the solution and reduce the variance.

2.2 The Adjoint Problem

In general, the adjoint $A^\dagger$ of an operator, function, or matrix A is the complex conjugate of the transpose of A [4]. It has the property:

$$\left\langle \psi^\dagger, A\psi \right\rangle = \left\langle \psi A^\dagger, \psi^\dagger \right\rangle, \tag{2}$$

where $\langle \cdot \rangle$ represents "integration over all independent variables," which, in phase space, is given by [5]

$$\langle \cdot \rangle = \int d^3\boldsymbol{r} \int dE \int d\hat{\boldsymbol{\Omega}}$$

The adjoint of the Boltzmann Neutron Transport equation (TE) represents the importance of the contribution of a particle at a particular location, time, energy, and angle to an objective function such as detector source [6]. For the transport operator, H, defined in Eq. (3a), the adjoint transport operator $H^\dagger$ can be derived using the property given in Eq. (2) and is defined in Eq. (3b).

$$H = \hat{\boldsymbol{\Omega}} \cdot \nabla + \Sigma_t(\boldsymbol{r}, E) - \int_0^\infty dE' \int_{4\pi} d\hat{\boldsymbol{\Omega}}' \Sigma_s(\boldsymbol{r}, E' \to E, \hat{\boldsymbol{\Omega}}' \cdot \hat{\boldsymbol{\Omega}}) \tag{3a}$$

$$H^\dagger = -\hat{\boldsymbol{\Omega}} \cdot \nabla + \Sigma_t(\boldsymbol{r}, E) - \int_0^\infty dE' \int_{4\pi} d\hat{\boldsymbol{\Omega}}' \Sigma_s(\boldsymbol{r}, E \to E', \hat{\boldsymbol{\Omega}} \cdot \hat{\boldsymbol{\Omega}}') \tag{3b}$$

From this equation, the adjoint transport equation can be used to set up the adjoint problem, analogous to the forward problem (except where the detector response replaces the external source and the the adjoint flux replaces the forward flux). Just as the forward problem is given by $H\psi = q$, the adjoint problem is given by $H^\dagger \psi^\dagger = q^\dagger$ where $q^\dagger$ is the detector response function.

The complete detector response R is defined as the integral over all phase-space of the product of the detector response function and forward flux:

$$R = \int d^3\boldsymbol{r} \int dE \int d\hat{\boldsymbol{\Omega}} \psi(\boldsymbol{r}, E, \hat{\boldsymbol{\Omega}}) q^\dagger$$

Additionally, using the adjoint equivalence identity, it can be shown that

$$R = \int d^3\boldsymbol{r} \int dE \int d\hat{\boldsymbol{\Omega}} \psi^\dagger(\boldsymbol{r}, E, \hat{\boldsymbol{\Omega}}) q(\boldsymbol{r}, E, \hat{\boldsymbol{\Omega}})$$

where q is a probability distribution function (PDF) for the source q.

2.3 CADIS

The Consistent Adjoint Driven Importance Sampling (CADIS) method employs the discrete ordinates S_N adjoint function for "automatic variance reduction of Monte Carlo calculations" [6], with a goal of constructing reduced variance parameters for Monte Carlo simulations. Accomplishing this requires finding window weights ww, birth weight w_0, and a biased PDF $\hat{q}$. These are given in Eqs. (4) to (6).

CADIS relies on both source and transport biasing, two of three significant categories of biasing schemes, with the third being collision biasing. Performing the CADIS method requires two major steps – explained in detail in Fig. 1 – to implement both biasing schemes.

First, the source biasing parameters are calculated using a space- and energy-dependent adjoint function $\phi^\dagger$. The biased (also refered to as "nonanalog") probability distribution, $\hat{q}$, is given in Eq. (4) [6], with q being the unbiased source PDF, ψ the scalar flux, and R the detector response.

$$\hat{q}(\boldsymbol{r}, E) = \frac{q(\boldsymbol{r}, E)\phi^\dagger(\boldsymbol{r}, E)}{R} \tag{4}$$

The adjoint scalar flux is typically calculated using a simplified deterministic calculation, as this is sufficient for appreciably reducing variances.

Intuitively, the ratio expressed in Eq. (4) represents the fraction of the total detector response R which is due to a contribution from phase-space element $(\boldsymbol{r}, E)$. It is important to note here that CADIS primarily enhances space and energy components of the calculation *only*. Wagner and Haghighat describe that angular-dependent problems would require significantly more memory and could be less accurate as a result of the limitations of using discrete ordinates in three dimensions [6].

This source biasing is used to generate the weight windows, which are used in transport biasing. The weight windows form the basis for Monte Carlo space- and energy-dependent splitting and rouletting. The weight window target ww is given by Eq. (5) and the weight window birth weight, w_0, is given by Eq. (6) [6, 7].

$$ww = \frac{\phi^\dagger(\boldsymbol{r}, E)}{R} \tag{5}$$

$$w_0 \equiv \frac{q(\boldsymbol{r}, E)}{\hat{q}(\boldsymbol{r}, E)} \tag{6}$$

From Eq. (6) the weight-window lower bound w_ℓ can be defined as

$$w_\ell \equiv \frac{2}{1+r}\frac{q(\boldsymbol{r}, E)}{\hat{q}(\boldsymbol{r}, E)} \tag{7}$$

with r being the ratio of upper to lower weight window boundaries, such that the boundaries are defined with respect to a particle's birth weight. Since every particle is born into a corresponding target weight, the end result is consistent with Eq. (1). The biased source and the weight window information are then passed to a standard Monte Carlo algorithm for calculation.

The CADIS method begins to lose efficiency however, when used to determine responses for multiple tallies. These could be include problems such as those with multiple energy bin tallies or multiple detector locations. Several methods of handling these situations have been presented, though none have been optimal. These methods include performing N total simulations for each of the N tallies, or setting the total detector response equal to the sum of individual responses over the range of interest [8]

$$R = R_1 + R_2 + ... + R_N. \tag{8}$$

It is obvious that the former strategy becomes impractical when the number of simulations becomes large, meanwhile the latter suffers with distance from the adjoint source since Eq. (8) implies

$$q^\dagger = q_1^\dagger + q_2^\dagger + ... + q_N^\dagger. \tag{9}$$

The dependence on distance becomes clear when one considers that the importance map is determined by a particle's expected contribution to the total response, and particles far from the source will be less likely to contribute [8].

2.4 FW-CADIS

To improve upon the shortcomings of the CADIS method, the Forward Weighted Consistent Adjoint Driven Importance Sampling (FW-CADIS) method was developed. Since the treatment of the detector response R as a summation of individual responses has diminishing returns as sources gain distance from the source, FW-CADIS seeks to weight the adjoint source contribution $q^\dagger$ from any given component $q_i^\dagger$ according to its individual response R_i. If a contribution is far from the source, it will have a small relative contribution and thus it will be given greater weight. Then, in general, the adjoint source $q^\dagger$ will be given by

$$q^\dagger = \frac{1}{R_1} q_1^\dagger + \frac{1}{R_2} q_2^\dagger + ... + \frac{1}{R_3} q_3^\dagger. \tag{10}$$

In order to execute this procedure and construct the adjoint source $q^\dagger$ for use in the standard CADIS method, an additional deterministic *forward* calculation must be performed [8].

3 Algorithms

3.1 Overview

The hybrid method requires a deterministic calculation to find the weight window, ww, particle birth weight, and biased source probability, $\hat{\rho}$, which are then used in the stochastic calculation. The deterministic calculation can be accomplished with the AutomateD VAriaNce reducTion Generator (ADVANTG), which performs the deterministic calculations using Denovo, a "3-D, block parallel discrete ordinates transport code" from ORNL [8]. ADVANTG implements either the CADIS or the FW-CADIS method based on a pair of user-provided input files: an MCNP input and an ADVANTG input. The MCNP input specifies the problem's geometry, materials, and meshing, while the ADVANTG input supplies information for the Denovo calculation (such as quadrature type and order, and the P_N order of the scattering expansion). For both CADIS and FW-CADIS, ADVANTG outputs a modified MCNP input file as well as a weight window input file, `wwinp`, which contains information on ww. With these inputs, MCNP can be run to complete the calculations. The complete CADIS process is described graphically in Fig. 1.

The FW-CADIS method, depicted graphically in Fig. 2, involves two major steps: a forward deterministic calculation to find the adjoint source, $q^\dagger$, and an adjoint deterministic calculation to create an importance map for Monte Carlo calculations [1]. ADVANTG accomplishes this by performing a Forward deterministic calculation to find the Denovo flux solution, which is used to create the FW-CADIS adjoint. Then, ADVANTG runs an adjoint deterministic calculation to find the adjoint Denovo flux solution [8].

3.2 Implementation

With 7 unique geometries, 12 combinations of input parameters, and processing with both CADIS and FW-CADIS, a total of 168 distinct runs were required to assemble a complete set of data. To handle this process efficiently, automation procedures were established for ADVANTG and MCNP implementation so that high-

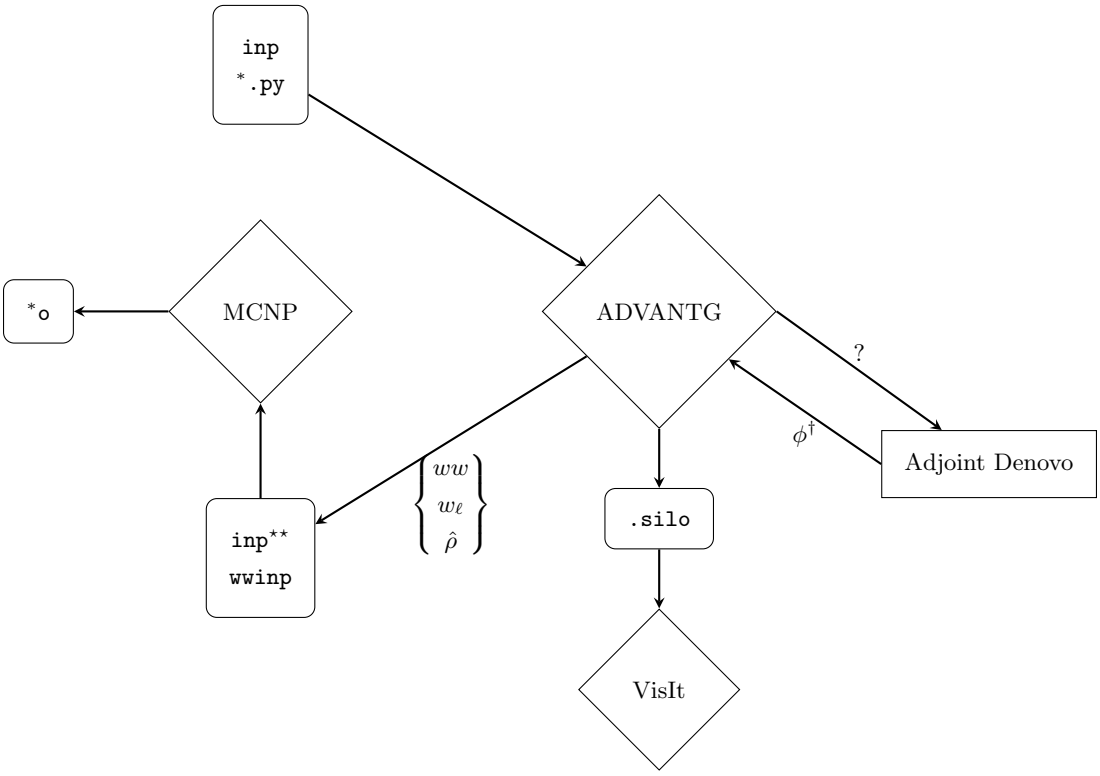


Figure 1: CADIS Method – The process begins with a user-generated MCNP input file `inp` and an ADVANTG input (`*.py`). Running ADVANTG will generate an output `.silo` file, modified MCNP input, denoted `inp**`, and a `wwinp` file. The latter two are used to run MCNP, which outputs a file `*o`.

performance computing capabilities could be utilized. The whole process from start to finish was segmented such that each group member took primary responsibility for managing the implementation of each software package: ADVANTG, MCNP, and VisIt. Under these guidelines, no two group members worked with the same stage of the calculations simultaneously, but excellent communication between members was crucial for passing inputs and outputs between stages.

3.2.1 ADVANTG Input

Inputs into ADVANTG were selected to cover a wide range of the parameter space for a given input geometry. A full set of geometries is provided in section 4.1. For each geometry, various quadrature types were tested with each type being sampled for a range of quadrature orders. Quadrature types included level-symmetric, Gauss-Legendre, and quadruple range. For Gauss-Legendre and quadruple range quadrature types, both product quadratures, the quadrature order was given as number of azimuthal and polar angles (both set equal in all of our tests). For level-symmetric, a triangular quadrature type, the order was given as a single value denoting the number of levels per octant.

A complete list of parameter assignments that were tested for a given case are listed in Table 1. The specific assignments were selected to give a spread of each parameter within the ADVANTG-imposed limits on that parameter.

In order to run parallel ADVANTG calculations, a series of programs sets were developed to formulate

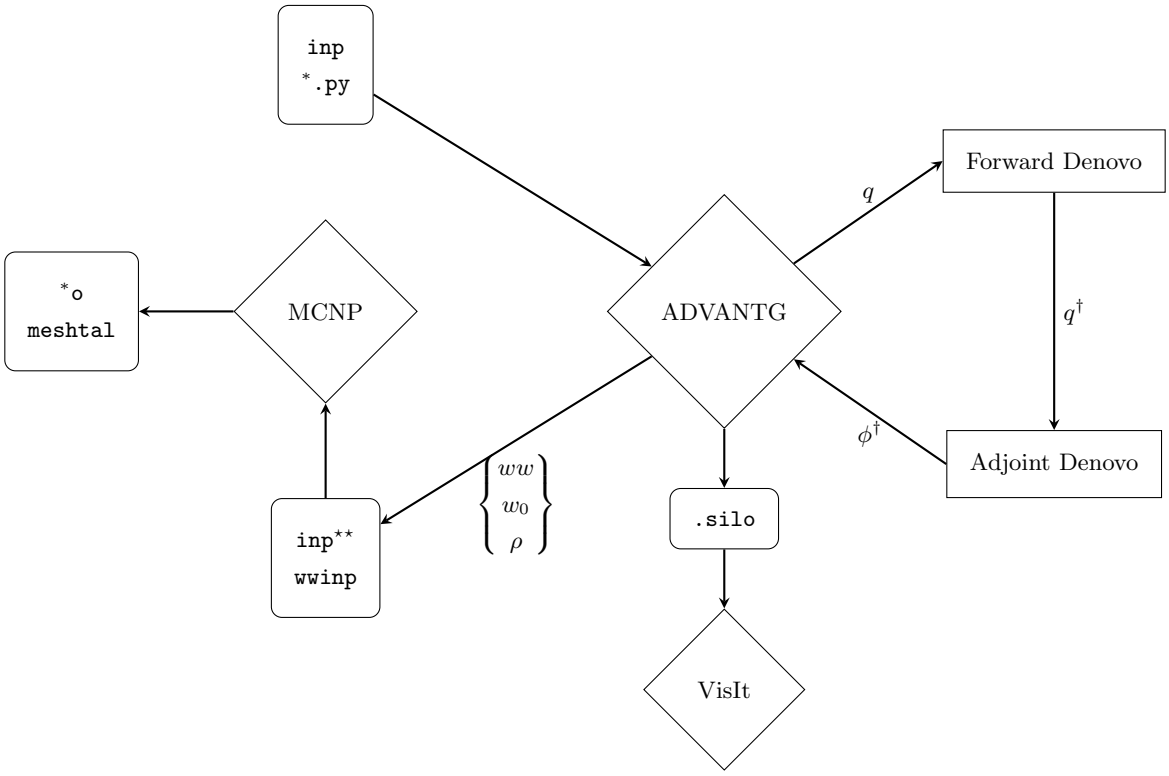


Figure 2: FW-CADIS

Table 1: List of assignments to be tested for each parameter. Optimization will seek the combination of parameter assignments which gives the lowest uncertainties as well as maximal efficiency.

Parameter	Parameter Assignment	Notes
Quadrature Type	Quadruple range (tri.) Level Symmetric (tri.) Gauss Legendre (prod.)	
Quadrature Order *Triangular only*	$10, 16, 20$	default 10; max. 32 for Q.R.; max 24 for L.S.
Quadrature Angle Number *Product only* Ordered Pair: (Azimuthal, Polar)	$(2,2), (4,4), (10,10), (16,16)$	default $(4,4)$; max. $(37,16)$
Legendre Scatter Expansion Order (P_N order)	$3, 5^*$	default 2

$^*P_N$ order 5 was only tested for level-symmetric, order 10 quadrature sets

each distinct test case and then execute ADVANTG through the ADVANTG python driver. The python driver allows ADVANTG to be executed through python (2.7) by supplying input parameters as key-value

pairs in a standard python dictionary.

The first set of codes synthesizes user inputs and an ADVANTG python driver input template to generate and store distinct python driver input files and associated run-scripts. Run-scripts alotted one geometry-parameter configuration per job, and were designed for submission to an HPC cluster running Slurm Workload Manager.

The second set of programs managed the submission of these jobs to the cluster. The code finds python driver files and associated run-scripts for an input geometry and submits those inputs for batch processing. After processing, output files are generated and, if desired, outputs are moved to a high-capacity storage location.

Submission was kept separate from run-script generation so that jobs could be submitted independently if necessary, and because of the disparity between generation and ADVANTG runtime (a few seconds as opposed to several minutes or hours). This arrangement is especially useful in situations where computing resources are limited or new configurations are to be tested and the user does not want to run all cases simultaneously.

3.2.2 ADVANTG Output and MCNP Input

ADVANTG's main output is a weight window input (WWINP) MCNP file based on the input MCNP inp file. The WWINP file appends two types of cards to the original inp file: replacement biased source (`SB`) cards, which contain importance-weighted biased probabilities $\hat{\mathbf{rho}}$, and weight window parameter (`WWP`) cards [8]. Inspecting the Silo file and visualizing the data with VisIt will be automated.

ADVANTG also outputs Silo files, which can be inspected using the VisIt visualization toolkit [9] for quality assurance. Upon passing inspection, the corresponding MCNP file will run to calculate the dose measured for the given problem. The process of running MCNP input files in the MCNP command window will be automated. When optimizing the computational time and result accuracy for various input parameters, it will be important to know the Denovo run times so that these can be contributed to the total run time of the process and studied individually. This process will also be automated.

3.2.3 Read MCNP Output

Once MCNP runs the WWINP file created by ADVANTG, it will create an output file containing all of the information it collected about the test problem and while running the test problem. The user is able to select whether they want MCNP to return the solution to first- or second-order accuracy [10]. Reading these error terms will be beneficial in determining MCNP's confidence in the final result.

One of the major benefits in using a deterministic method to compute the variance reduction is to reduce computational time, since deterministic methods are generally faster than Monte Carlo methods. Monitoring the run time of both MCNP and Denovo will reveal when the deterministic method is improving run time and by how much.

First, the ADVANTG output files, `inp` and `wwinp`, were copied to the user directory. Savio runs faster within user directories than on the scratch node, where the team files are shared. Copying files into user directories for every execution of code for which run time would be important to the FOM (for Advantg and MCNP runs) helped achieve optimal FOM for more accurate and consistent method comparisons. Moving the files significantly improved computational time: an MCNP run that took fifteen minutes in the scratch directory might take twenty seconds to finish in a user directory.

A drawback of using the user directories is that only a limited amount of space is available. For the amount of data we had, this became challenging and required careful monitoring of remaining space available and copying files between the user and scratch directories.

The process by which shell files to run MCNP in user directory.ies was automated in the `mcnprun.sh` script. This calls the `navigator.py` and

4 Code Use

4.1 Test Cases

As specified by the work of Madicken Munk [11], seven test cases were run using ADVANTG:

1. A five legged labyrinth running across a Lithium-doped polyethylene shield.
2. A three legged labyrinth running across a 100 centimeter-thick concrete shield.
3. A 100 centimeter-thick block of polyethylene with a steel bar going through it, acting as a streaming channel.
4. A labyrinth with a source on its left and a detector on its right.
5. A polyethylene shielding wall thick shielding with a source on its left and a detector on its right and steel support structures running through the shield.
6. A beam source of neutrons directed towards a section of NaI.
7. A therapy room derived from a radiation therapy vault.

4.2 Generating ADVANTG Inputs

All files associated with running ADVANTG are located within a single `ADVANTG_tests` directory. Within this directory, a subdirectory exists for executing operations up to and including running ADVANTG; this is the `ADVANTG_tests/main` subdirectory. To create the python driver files and corresponding runscripts, the user must run the advinpGen.py file. This generator script relies on two ADVANTG python driver templates (one for product quadrature types and another for triangular quadrature types). For successful operation, the ADVANTG generator file must be edited by the user prior to execution to include the following parameters:

- **Method:** CADIS or FW-CADIS
- **MCNP Input:** MCNP geometry (*i.e* maze1, maze2, beam)
- **MCNP Tallies:** tallies for MCNP geometry
- **Denovo Quadrature:** quadrature set type to use
- **Denovo Quadrature Order:** quadrature order
- P_N **Order:** scattering expansion order

Additionally, the path to the template files must be supplied in the script along with the path to a pair of user-defined directories where ADVANTG driver inputs and runscripts will be generated. The template files are designed to be generic so that unless the user wishes to change other parameters than those listed above, no modification to the template files must be made.

The script iterates through all combinations of these parameters and generates outputs. The program is able to differentiate between product and triangular quadrature sets, and will modify the appropriate template.

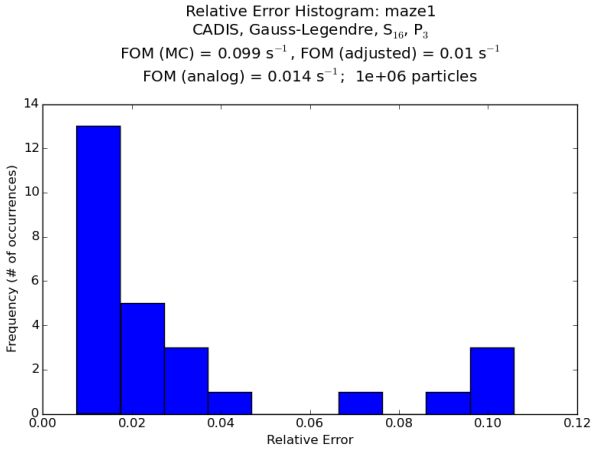
During execution of the ADVANTG input generator, the runscript generator module, runscriptGen.py, is also invoked. This module will produces runscripts in the directory specified in the advinpGen.py script which are compatible with the Slurm Workload Manager available on UC Berkeley's Savio. If another workload manager is used, the runscriptGen.py module must be updated accordingly.

To submit jobs to a high-performance computing cluster, the runscripts can be submitted using the $\texttt{runADVANTG}_geom.shBASHshellscript(foundinthe$

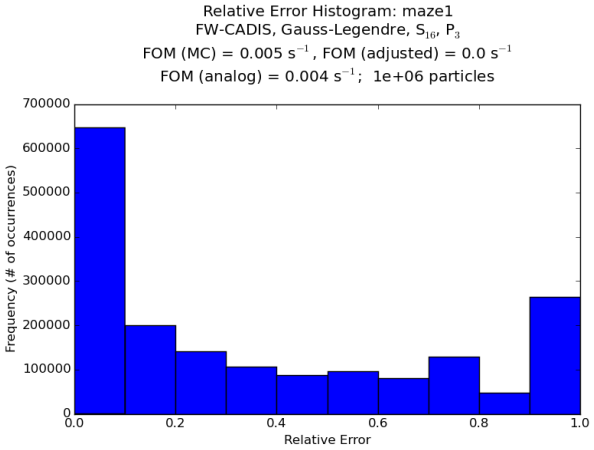
5 Test Projects and Results

We ran the simulations for the beam, maze1, maze2, prob_1, prob_2, prob_4, each simulation, we looked at the maximum relative error, simulation run time, and both MCNP and adjusted figure of merit, where the adjusted FOM includes the Denovo run time as well as the MCNP run time. For CADIS, it was possible to plot both the tally output results and relative errors as functions of energy bin. For both CADIS and FW-CADIS, we plotted histograms of the distribution of the relative errors. Examples of these four plots are shown in Figs. 4a and 4b.

For calculation analysis, we restricted our test cases to maze1, maze2, and beam geometries. Our findings indicate that the CADIS method consistently provided better figure of merit values for maze1 and maze2, while the beam geometry performed better using FW-CADIS.

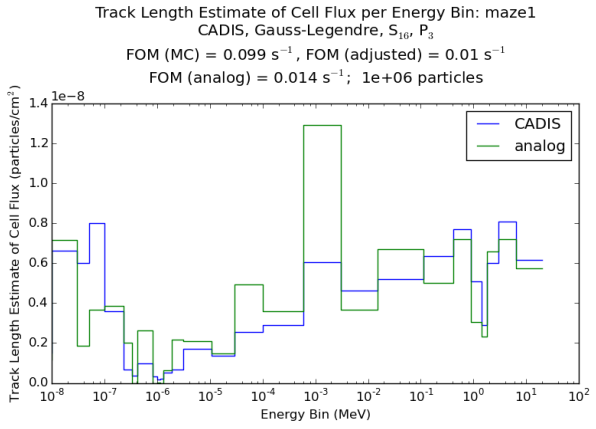


(a) Histogram of Relative Error for CADIS

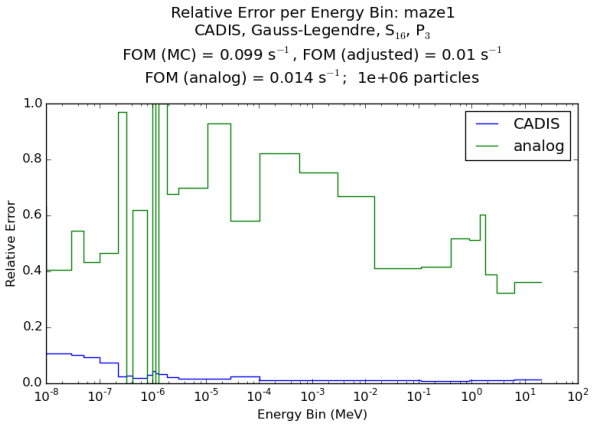


(b) Histogram of Relative Error for FW-CADIS

Figure 3: Maze 1 Geometry



(a) Relative error vs. of energy bin



(b) Relative error vs. of energy bin

Figure 4: Maze 1 Geometry for CADIS

6 Summary

Throughout the course of this project, CADIS and FW-CADIS runs were performed for seven geometries, each with varied parameters. After running CADIS and FW-CADIS, we find the behavior of the figure of merit for a given method is certainly dependent on the geometry in question.

References

[1] R. N. Slaybaugh and S. C. Wilson. "Deterministic Parameter Study for Fixed-Source Calculations Using FW-CADIS". *Proceedings of the 2013 ANS Annual Meeting*. Vol. 108. Atlanta, GA, June 2013. DOI: 10.13140/2.1.2255.1688.

[2] I. G. Hughes and T. P. A Hase. *Measurements and their Uncertainties*. Oxford University Press, 2010. ISBN: 978-0-19-956633-4.

[3] T. E. Booth et al. "MCNP A General Monte Carlo N-Particle Transport Code, Version 5". *LA-UR-03-1987* (2008).

[4] M. L. Boas. *Mathematical Methods in the Physical Sciences*. 3rd. John Wiley & Sons, 2003. ISBN: 978-0-471-19826-8.

[5] E. E. Lewis and W. F. Miller. *Computational Methods of Neutron Transport*. John Wiley & Sons, 1984. ISBN: 0-471-09245-2.

[6] J. C. Wagner and A. Haghighat. "Automated Variance Reduction of Monte Carlo Shielding Calculations Using the Discrete Ordinates Adjoint Function". *Nucl Sci Eng* 128.2 (Feb. 1998), pp. 186–208. DOI: 10.13182/NSE98-2.

[7] R. N. Slaybaugh. "Lecture Notes from NE 250: Nuclear Reactor Theory on Oct 28". University of California, Berkeley, 2015.

[8] S. Mosher and etal. "ADVANTG–An Automated Variance Reduction Parameter Generator" (2013).

[9] Hank Childs et al. "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data". *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*. Oct. 2012, pp. 357–372.

[10] T. Goorley. "MCNP6.1.1-Beta Release Notes". *LA-UR-14-24680* (2014).

[11] M. Munk. *caskmodels*. 2016. URL: https://github.com/munkm/caskmodels/tree/master/characterization_tests.

[12] .

A Code Organization

Group files were shared on the UC Berkeley high-performance computing cluster, Savio, in the /global/scratch/co_nuclear/NE255project/ directory. Consistent and thoughtful file organization was extremely important to the automation process.

Code for automation of ADVANTG inputs can be found on GitHub at https://github.com/mnegus01/caskmodels

Table 2: Compare P_3 and P_5

				Max.	Run Time (s)		Figure of Merit (s^{-1})	
Quadrature	S_N	P_N	Method	Rel. Err.	Denovo	MCNP	MCNP	Adjusted
LS	10	3	CADIS	0.0159	457.85	979.2	4.039 562 636	2.752 541 48
LS	10	5	CADIS	592.67	1006.8	0.418 792 476	0.263 612 487	
LS	10	3	CADIS	326.05	5809.2	0.213 425 836	0.202 083 593	
LS	10	5	CADIS	519.56	6607.8	0.046 743 215	0.043 335 795	
LS	10	3	CADIS	0.1286	44.345	16.2	3.732 527 776	0.998 710 876
LS	10	5	CADIS	10.742	49.8	0.582 300 08	0.478 982 26	
LS	10	3	CADIS	0.0013	278.147	14 644.2	40.406 166 01	39.653 010 1
LS	10	5	CADIS	376.221	25 250.4	0.000 775 38	0.000 763 997	
LS	10	3	CADIS	0.0204	280.533	8363.4	0.287 314 005	0.277 989 424
LS	10	5	CADIS	375.919	9454.2	0.013 087 474	0.012 586 989	
LS	10	3	CADIS	341.656	16 026	0.001 285 718	0.001 258 881	
LS	10	5	CADIS	440.767	15 902.4	0.001 762 274	0.001 714 746	
LS	10	3	CADIS	0.0461	280.837	61.8	7.613 943 927	1.373 295 163
LS	10	5	CADIS	479.715	62.4	0.133 709 198	0.015 390 561	

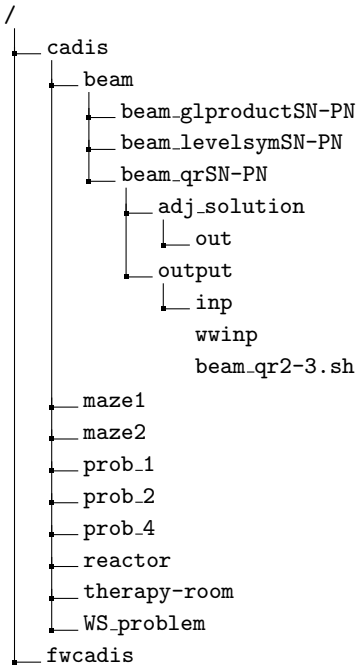


Figure 5

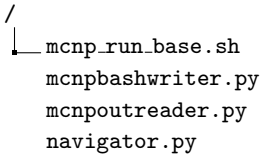


Figure 6