Geant4 simulations of neutron flux using automated weight windows with applications to ESS

John Stenander

Spring 2015
Master's thesis
Department of Physics
Division of Nuclear Physics

EUROPEAN
SPALLATION
SOURCE

Supervisor: Douglas Di Julio
Co-supervisor: Joakim Cederkäll

Abstract

This thesis investigates the use of two automated weight window methods for variance reduction of flux in particle transport simulations. The methods are implemented in the Geant4 simulation software. Both methods are based on pre-simulations, using flux or relative error information as input. The methods are intended for use in shielding and instrumental background applications and are tested on flux and spectrum calculations using two versions of an European Spallation Source (ESS) instrument shielding target model. For each model, the methods were implemented by defining a parallel geometry mesh grid which overlaid the mass geometry. In each mesh cell the number of tracks in the cell and the relative error of the number of tracks were scored. This information was used to update the weight windows.

Two figures of merit were introduced to assess the methods. The first figure of merit is inversely proportional to the square of the relative errors and the simulation time. The second figure of merit is the standard deviation of the relative errors. Both figures of merit showed improvement for the two methods and almost all simulation times. Compared to the analog simulation, the first figure of merit increased by a factor of 2.17 to 3.10 when applying the relative error based method and by a factor ranging from 2.10 to 14.60 when applying the flux based method. The second figure of merit decreased by a factor ranging from 1.33 to 2.89 when applying the relative error based method and increased by a factor ranging from 1.58 to 3.15 when applying the flux based method. The improvement depends on the simulation time and the simulated model. The methods showed improved figures of merit, reduced the parts of the simulated geometry that saw few Monte Carlo particles and reduced the error significantly in almost all spatial regions.

Contents

Contents ii

Acknowledgements iii

List of Abbreviations iv

1 Introduction 1
- 1.1 General overview . . . 1
- 1.2 Aim and limitations . . . 1
- 1.3 ESS . . . 2
- 1.4 Monte Carlo particle transport simulation software . . . 2
- 1.5 Applications . . . 3
  - 1.5.1 Instrument backgrounds . . . 3
  - 1.5.2 Medical isotopes . . . 3

2 Theory and background 4
- 2.1 Physics overview . . . 4
  - 2.1.1 General neutron physics . . . 4
  - 2.1.2 Spallation physics . . . 5
  - 2.1.3 Moderators . . . 7
  - 2.1.4 Background & shielding . . . 8
- 2.2 Monte Carlo methods . . . 10
  - 2.2.1 Overview of Monte Carlo methods . . . 10
  - 2.2.2 Error calculation in Monte Carlo methods . . . 10
- 2.3 Variance reduction methods . . . 11
  - 2.3.1 Importance sampling . . . 12
  - 2.3.2 Weight windows . . . 12
  - 2.3.3 Some particular weight window generator methods . . . 14
  - 2.3.4 An easy to implement global variance reduction procedure . . . 16

3 Simulation analysis 17
- 3.1 Realization of the method in Geant4 . . . 17
- 3.2 Results . . . 23

4 Conclusion 31
- 4.1 Conclusion . . . 31
- 4.2 Outlook and further work . . . 31

References 32

Acknowledgements

I would like to thank my supervisor, Douglas Di Julio, for providing great support during the work with the thesis and always pushing me forward. He has been most helpful during all stages of work and has given me ample opportunities to find an audience for my work. Furthermore, I would like to thank ESS for providing funding for me during the project and also for giving me a hospitable working environment. A thank you also to Joakim Cederkäll, for giving me this opportunity and providing comments on the thesis towards the end of the work.

List of Abbreviations

ADS - Accelerator Driven System
CERN - Conseil Européen pour la Recherche Nucléaire
ESS - European Spallation Source
ESS TDR - ESS Technical Design Report
FLUKA - FLUktuierende KAskade
FOM - Figure Of Merit
FW-CADIS - Forward Weighted-Consistent Adjoint Driven Importance Sampling
Geant4 - GEometry ANd Tracking
GFWW - Global Flux Weight Windows
GRWW - Global Response Weight Windows
GVR - Global Variance Reduction
HP - High Precision neutron model
i.i.d. - Independent and Identically Distributed
INC - Intra-Nuclear Cascade
LVR - Local Variance Reduction
MCNP - Monte Carlo N-Particle Transport Code
PDF - Probability Density Function
PHITS - Particle and Heavy Ion Transport code System
TS - Thermal Scattering
S/N - Signal to noise ratio
QGSP - Quark-Gluon String Precompound

1 Introduction

1.1 General overview

Monte Carlo simulation (section 1.4) is a great tool to use in many of the quantitative sciences, ranging from economics to biology and of course physics. These simulations are computationally intensive and there is a need to work with complex systems and problems in an efficient way. This thesis will provide a method for efficient computer simulations for use in neutron transport problems in spallation source applications.

A computer can only handle just so many particles thus a simulation will inevitably work with less particles than a real life experiment. Such a simulation may not catch all rare events, and it will produce unreliable statistics for spatial or energy regions that show low or non-existent particle flux. This problem can in principle be solved by increasing the simulation time, but it is often so that the values which initially have a good precision are the ones that gain from this, while the values that need precision improvement do not gain as much. The methods presented in this thesis will provide a way to allocate computer resources more efficiently (section 2.3). In a complex simulation there is a tangible probability that this poses a real problem. The complex model that is the subject of this thesis is the ESS target model[1]. This model was constructed previous to this work in Geant4 [1], a particle transport simulation toolkit. The method that is used is a weight window method with windows set using flux and the relative errors of the flux as a basis [2] for the simulation bias.

This thesis will begin with an introduction to the tools of science it will consider, including both the ESS facility and the simulation software available. Furthermore, an introduction to the physics of the problem, spallation sources and background radiation will be given a brief overview as well as Monte Carlo methods in general and a more in depth discussion about variance reduction techniques. The results are presented in the final section where it is shown that this method does work as an efficient simulation bias to produce reliable statistics everywhere without with reduced simulation times. The methods are compared using figures of merit and data for calculated fluxes and the relative errors are presented.

1.2 Aim and limitations

The goal of this work is to present a usable non-deterministic weight window (section 2.3.2) method to be used in calculations of neutron flux in Geant4. Furthermore, some applications of this as well as the physics behind spallation (section 2.1.2) sources are explored.

The model to be used is fixed and so is the implementing platform. While no alternatives are considered for use, other methods are described and reviewed in this thesis. All applications considered will be taken using the ESS target model (in different forms). A method like this would have applications in many different particle transport simulations,

[1] *The ESS target model* here and here after refer to the *ESS target model for instrument shielding*, constructed in Geant4.

but only this model is considered in this thesis.

1.3 ESS

ESS is a neutron production facility being constructed in Lund and it is expected to be operational in 2019. ESS will be the world's most powerful and modern spallation neutron source when it reaches full capacity in 2025 [3].

The ESS facility will be a pulsed neutron spallation source with 125 MW peak power [4]. In general, spallation is a process where neutrons are released from a material as a result of bombardment with medium energy range particles (section 2.1.1). In the particular case of ESS, protons are used with a tungsten target [5].

The primary use of the facility will be the possibility to probe into condensed matter objects [3]. This is one of the largest research fields of physics as everything around us is made of condensed matter and the study of it has given us everything from cell phones to sturdy materials as well as a deeper insight into the structure of the universe. ESS will act as a large scale powerful microscope, but instead of using photons (light) as the probing particle, neutrons are to be used. Besides working as an effective microscope it will provide ways to probe matter in a non-intrusive way. This is useful, for example, to study old paintings or running engines [3]. Furthermore, it will be possible to conduct fundamental physics experiments at ESS [3].

The ESS facility will be of use for physicists, medical scientists, chemists and archaeologists amongst others. ESS will also aid the study and improvement of drugs, electronic components and new materials. It will give new ways to investigate historical artifacts and the properties of space.

1.4 Monte Carlo particle transport simulation software

There are many software solutions available for particle transport simulations. The framework being used for this project is Geant4, an open source CERN developed package [1]. It is important to make it possible for scientists to communicate ideas and to build upon their peers work. A closed source code makes this harder. Having these models available in Geant4 (or any other open source code) is important for the scientific community in large.

MCNP is another popular software for radiation transport and is available under proprietary license [6]. The methods that are being used in this project have been implemented using MCNP [2] and the challenge is to do the same in Geant4.

Geant4 is mainly used for high energy physics simulations while MCNP is developed for nuclear science problems. Both contain weight window algorithms (section 2.3.2) and what is missing in Geant4 is the method to set the weights.

Apart from the two code packages discussed above, one must also mention FLUKA [7] and PHITS [8]. FLUKA is - like the ones mentioned above - a tool for calculations regarding particle transport. FLUKA is distributed in binaries but the source code is available

under license agreement. PHITS is a Fortran based general purpose particle transport code and it is distributed as source packages.

While the proximate cause to use Geant4 in this thesis is that the ESS target model is already implemented and is used among scientists working in the ESS project, there is a more interesting ultimate cause. Geant4 is written in a modern language (C++), is open-source and may be modified, it is available freely in source format and is supported and tested on a number of platforms.

1.5 Applications

1.5.1 Instrument backgrounds

In a spallation source, higher neutron energies than those in a traditional nuclear reactor are observed. These can be in the GeV range, compared to $\approx 20\,\mathrm{MeV}$ for reactor sources. Shielding is used to lower the background, that would otherwise be detected by instruments and provide a noise for all measurements. Different shielding materials have different properties (2.1.4), differing in for example transparency, price and efficiency for different types of radiation. Thus, for design purposes, it is important to be able to evaluate the different shielding options. For higher energies, there is still room for improvements in calculations regarding shielding material properties [9]. The paper by Cherkashyna et al. [9] has made tentative Geant4 simulations of some possible shielding materials.

For a spallation source, shielding is important to reduce the high energy background on instruments. Cherkashyna et al. states that shielding may be the best way to reduce backgrounds. High energy backgrounds have an adverse effect on the S/N (signal to noise) ratio in instruments and reduce the functionality of the neutron scattering instruments. The contributions to the background can arise from deep penetration of neutrons, sky- or groundshine (neutrons scattered from the atmosphere or ground) and streaming along neutron guides (background radiation that follows neutron guides in the shielding). These are all computationally expensive and warrant a method to perform faster flux calculations. For further investigation of these shielding materials in large scale models, a method like the one in this thesis would improve the results.

1.5.2 Medical isotopes

While the primary use of ESS will be neutron scattering experiments, some alternative uses might be considered. One of these uses is the production of medical radioisotopes. There are many different isotopes in use that are possible for production at ESS and one of the more interesting is technetium-99m ($^{99\mathrm{m}}$Tc). This is a short-lived metastable (hence the m) isomer of technetium-99 (^{99}Tc). $^{99\mathrm{m}}$Tc is produced by the reaction $^{99}\mathrm{Mo} \rightarrow ^{99\mathrm{m}}\mathrm{Tc}$ [10]. This can be, and is, produced in nuclear reactors by neutron irradiation of ^{235}U and ^{98}Mo. This has some drawbacks. First of all it produces nuclear waste. Second, there is a predicted shortage and even an almost complete halt in production, as observed by the trend in historical production, with previous dips in production to set an example [11].

If production of medical isotopes is to be possible, the neutron spectrum has to be known

at certain places. A variance reduction method would be of good use to populate those regions properly.

2 Theory and background

2.1 Physics overview

2.1.1 General neutron physics

In this chapter - and further physics sections - some aspects of neutron physics are discussed. For this an outline of neutron naming conventions and categorization is needed. Neutrons are categorized and named based on their energy ranges. This nomenclature will be used in this thesis and is summarized in table 1.

Table 1: Used neutron naming conventions [12].

Energy range	**Name**
Up to 0.025 eV	Cold neutrons
Around 0.025 eV	Thermal neutrons
1 to 10 eV	Slow neutrons
1 to 20 MeV	Fast neutrons
> 20 MeV	Relativistic neutrons

Neutron transport is governed by the neutron transport equation [13]. The solution and applications of this equation are not considered here as this work will implement a non-deterministic Monte Carlo method.

Two important concepts in neutron physics are the *cross section* and the *mean free path*. The cross section (σ) is a factor measured in the unit of barn, $1\,\mathrm{b} = 10^{-24}\,\mathrm{cm}^2$. The cross section characterizes the probability that a given reaction occurs. The macroscopic cross section (Σ) is the cross section of a nuclide times the number of atoms per cm^3. The macroscopic cross section (Σ) can be defined as $N\sigma = \Sigma$ where N is the density of nuclei and thus gives the probability for reaction per travelled distance (cm^{-1}) [14].

Related to cross sections is the mean free path (λ). Take $p(x)$ as the probability that a neutron has *not* had a reaction while traveling the distance x in a homogeneous medium. It follows that

$$\mathrm{d}p = \Sigma p \mathrm{d}x \Leftrightarrow p(x) = e^{-\Sigma x} \Rightarrow \frac{\mathrm{d}p}{\mathrm{d}x} = \Sigma e^{-\Sigma x}, \tag{1}$$

which is the probability of a reaction when a neutron travels over the distance $\mathrm{d}x$. The expected value of x is called the mean free path and given as

$$\lambda = \int_0^\infty = x\Sigma e^{-\Sigma x}\mathrm{d}x = \frac{1}{\Sigma}, \tag{2}$$

where λ is a measure of how far we expect a neutron to travel before a reaction [13].

2.1.2 Spallation physics

A theory of spallation

Spallation in general is the complex process of reactions initiated by energetic particles and (heavy) nuclei [15]. In the particular case of ESS, the high energetic particles are in the GeV energy range.

For an incoming proton in this energy range, the de Broglie wavelength (3) is of the order of 10^{-16} m. This is smaller than the diameter of a nucleus (order of 10^{-15} m for lighter nuclei to 10^{-14} m for heavier nuclei [16]). The de Broglie wavelength is defined as

$$\lambda = \frac{h}{\sqrt{2mE}}. \tag{3}$$

This motivates the use of the intra-nuclear cascade model (INC) [18]. This model treats the interaction of neutrons and nucleons as collisions and scatterings with nucleons inside the nuclei.

A Bertini cascade model is implemented in the physics list "BERT" in Geant4 and is described in [17]. The intra-nuclear cascade (or the "hadron cascade") results in pions, neutrons and protons. The initial cascade (cascade stage) results in secondary (transition stage) cascades via an inter-nuclear cascade, where particles transfer energy and momentum to other nuclei.

In the end the nucleus is in an excited state and nuclear evaporation occurs (evaporation stage). The neutron spectrum is then described by

$$n(E) = A_1 E^{1/2} e^{-E/E_{T_1}} + A_2 E e^{-E/E_{T_2}} + A_3 E e^{-E/E_{T_3}}, \tag{4}$$

for some characteristic energies E_{T_i}, where i denotes the energies emanating from the three stages [5]. Stage 2 and 3 are responsible for about 90% of the emitted neutrons while neutrons emanating from stage 1 can have energies close to the incoming proton energy.

One thing to note here is that the neutron yield (number of neutrons per proton) does not only depend on the material. The size of the target also matters, and this is because of the inter-nuclear cascade. For a very thin target, the emitted nucleons would have less nuclei to interact with and make secondary reactions with. An empirical relation is,

$$Y(E) = a(A + 20)(E - b), \tag{5}$$

where Y is the neutron yield, A is the atomic mass and $b(\approx 0.12\,\text{GeV})$ is a threshold value [5]. Under this threshold, spallation is improbable. The parameter a is a material dependent value, set to 0.1 for all heavy elements excluding ^{234}U [5]. Spallation is possible in all nuclei but (5) shows that the yield increases with nuclear mass [18].

Spallation in practice

Spallation is one method to release neutrons. There are other sources for neutron emission with the most common man-made sources using fission and fusion. There are still other natural occurring sources stemming from spontaneous fission or radioactive decay [19]. Unlike the two other methods - fission and fusion - spallation is endothermal and does not currently find any direct use in the energy industry (but it would be possible to use a spallation source to drive a fission reactor, such a solution is called an accelerator driven system (ADS) [20]).

Spallation reactions are induced and maintained with an energetic beam. At ESS a 2 GeV proton beam will be used [3]. Furthermore, spallation sources generate a *hard* neutron spectrum (high neutron population at higher energies). See Figure 1 for a comparison between a fission neutron spectrum and a spallation neutron spectrum generated using an 800 MeV proton beam [5]. The neutron yield is ≈ 20 in practical applications [15] and can be calculated using (5).

So far, the described spallation source consists of two components, a proton source and a target (tungsten in the ESS case [3]). Further components include the moderators, shields, beam tubes and reflectors. The moderators and shielding will be discussed in some detail in sections 2.1.3 and 2.1.4. An outline of the ESS target target region is shown in Figure 2.

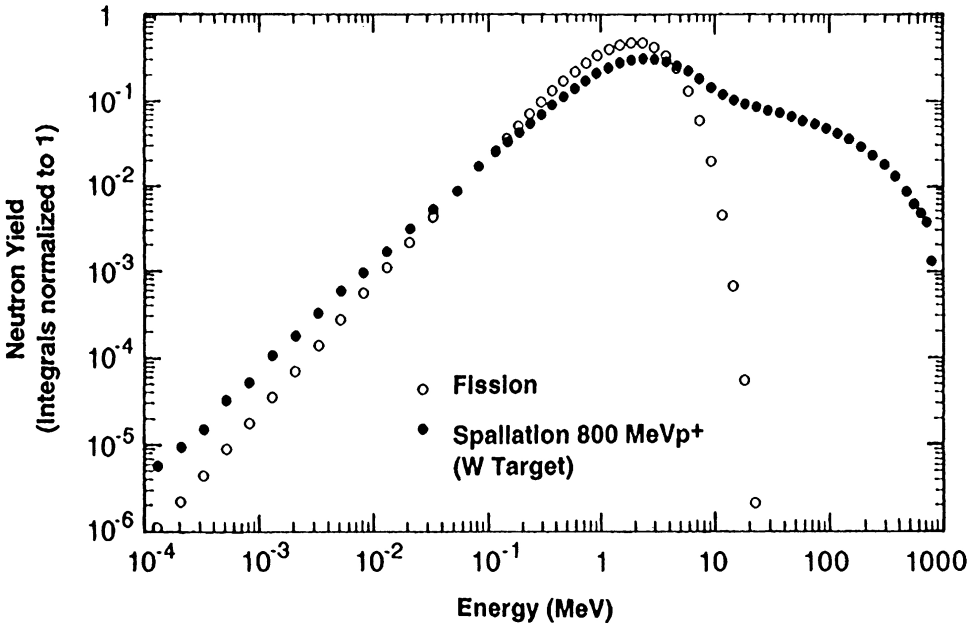

Figure 1: Calculated fission (∘) and spallation (·) neutron spectra [5][2].

[2]Reprinted from Nuclear instruments & methods in physics research, 463(3):505-543, G.S. Bauer, Physics and technology of spallation neutron sources, 39., 2001, with permission from Elsevier.

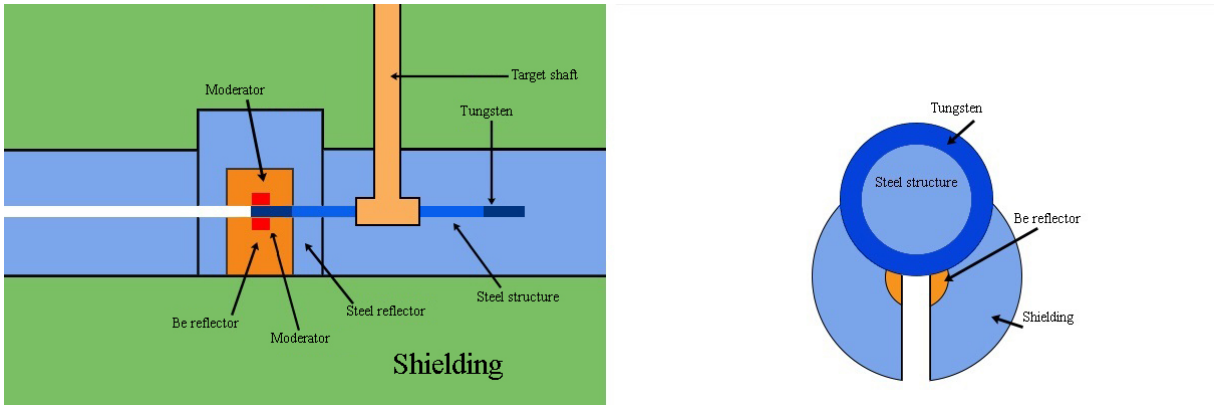

Figure 2: Outline of the ESS target region based on figures in Ref. [3]. In this figure the most important components are shown. The components that are discussed in the following sections in particular are the moderators, shielding and reflectors.

2.1.3 Moderators

In neutron scattering experiments, mainly neutrons in the cold or thermal ranges are useful (that is in the meV to eV range). Neutrons from a spallation reaction can be in the fast or even relativistic ranges [18]. This gives rise to the need of moderation as well as moderation with other characteristics than those in fission reactors. The primary purpose of the moderator is to lower the velocity of high energy neutrons and place them in a thermal or cold spectrum. For scattering experiments, long wave-length or cold neutrons are preferred.

Furthermore, the primary beam at ESS as well as in similar constructions, is pulsed. This puts an extra demand on the moderator as it has to preserve the time structure and neutron brightness. For this reason, moderators designed for spallation sources will be smaller than their counterparts in fission reactors. The energy loss of neutrons are due to elastic scattering (collisions without excitation) with the moderator nuclei [13, 21].

Another factor is how often these collisions occur, which is dependent on the mean free path (see section 2.1.1). If the energy loss per collision and expected number of collisions per traveled distance is known then it is possible to calculate the total energy loss for a certain volume of the medium. The mean free path is inversely proportional to the scattering cross section, hence a large value for Σ is desired to give a small mean free path [5].

The fraction of energy lost in each collision is constant ($\frac{E_1}{E_2} = \text{const.}$) and the logarithm of this is given by

$$\zeta = \ln E_1 - \ln E_2 = 1 - (\frac{\alpha_0 \epsilon}{1 - \alpha_0 \epsilon}), \tag{6}$$

using $\epsilon = \ln(\frac{1}{\alpha_0})$ and $\alpha_0 = \frac{(A-1)^2}{(A+1)^2}$ and atomic number A [5]. ζ is one for $A = 1$ and decreases with A, making hydrogen the most efficient moderator element.

To make the loss independent of initial energy it is useful to adopt another scale for energy. Let $u = \ln \frac{E_0}{E}$ with $E_0 = 10\,\text{MeV}$, chosen so that $u > 0$ for most relevant neu-

trons. This scale is called the *lethargy* [13].

The neutron energy spectrum in most moderators can be described by a Maxwellian distribution (7) [15]. This is a good approximation at thermal equilibrium [21], and given as

$$n(E) = \frac{2E^{1/2}}{\pi^{1/2}E_T^{3/2}}e^{-E/E_T}, \tag{7}$$

where E_T is the kinetic energy (kT) for a given temperature [5].

For a good moderator in a spallation source there are some criteria; the absorption cross section should be made low (to preserve the neutron brightness) and the scattering cross sections should be high (to give a low mean free path, and more collisions). Also from the discussion above it is clear that the atomic number should be low. This makes hydrogen and hydrogen derivatives the most efficient moderators [21].

The pulse at ESS is "remarkably long" [3]. The highest peak flux is achieved with a supercritical (a point above the critical point, where temperature and pressure are such that gas or liquid does not exist but matter takes a supercritical form) para-hydrogen (with anti-parallel nuclear spin) moderator. In the interesting low energy region, the cross section properties of para- and ortho-hydrogen (with parallel nuclear spin) are different. The scattering cross section increases at low energies for ortho-hydrogen while it decreases for para-hydrogen. Thus, a moderator of para-hydrogen is transparent for neutrons in the lower energy range. The number of collisions will also decrease and this in turn implies a lower chance of neutron capture [3].

2.1.4 Background & shielding

The ESS facility is a measurement tool. In any measurement, noise is a problem as it affects the quality of the results. At spallation neutron sources, a part of the background comes from the high-energy particles from the source itself (rather than for example the cosmic background). Compared with reactor neutron sources, the spallation backgrounds are more extensive, putting different demands on shielding requirements. This background is comparatively low, but still interesting since it has an adverse effect on measurements [9]. However, it is possible to assess the background using computer simulations of the spallation source.

Monte Carlo simulations can answer important design questions regarding the design and shielding [9]. As this work aims to produce efficient Monte Carlo simulations, this section will give a review of backgrounds and shielding as a possible application of the methods implemented in this project.

The important physics to consider for shielding design is highlighted in figure 3 which shows the ^{56}Fe total neutron cross section. The cross section shows four distinct regions. At low energies, the cross section follows $E^{-1/2}$. In this region, the neutron energy is lower than the bonding energy of atomic compounds and neutrons will interact with an aggregate of nuclei and will excite internal modes of the sample, i.e. vibrations. In the region with the more or less constant cross section the reactions are dominated by potential

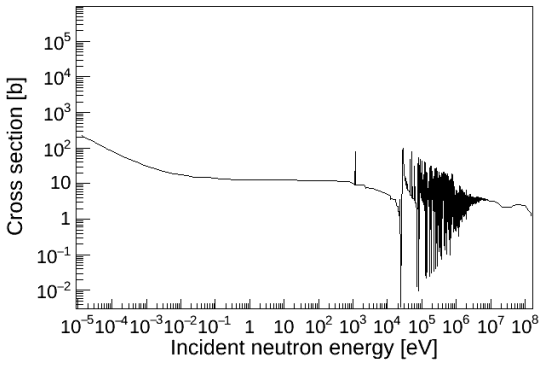

Figure 3: Total neutron reaction cross section for ^{56}Fe [25]

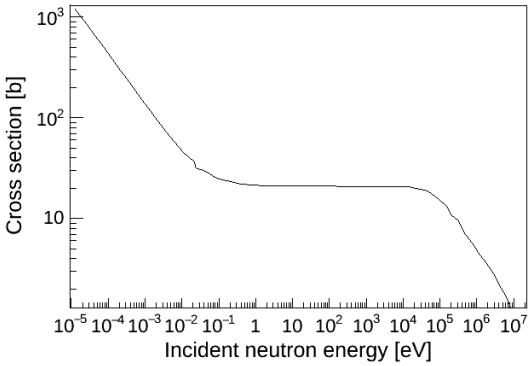

Figure 4: Total neutron reaction cross section for ^{1}H [25]

scattering and the cross section is more or less the geometric cross section of the nuclei. The spiked region is because of the resonance structure and can be described by (8). In the highest energy region the cross section drops off, this follows the energy dependent wavelength drop (3). The reaction probability will decrease with this wavelength [22].

Of particular importance are materials, like mild-steel for example, which show a pronounced resonance structure for given neutron frequencies [23]. These show up as the peaks in the cross section, as discussed above for Figure 3. This can be compared with hydrogen (Figure 4) which does not exhibit a resonance structure for any energy ranges. The peak is described by the Breit-Wigner formula [23], as $E \to 0$, given by

$$\sigma_{n,\gamma}(E) \approx \pi(\frac{\lambda}{2\pi})^2 \frac{\Gamma_n \Gamma_\gamma}{E_R^2}. \tag{8}$$

This formula gives a good predictive relation between cross section and the neutron energy, where λ is the de Broglie wavelength, $\Gamma_{n/\gamma} \propto \frac{1}{\tau}$ is the resonance width and E_R is the resonance energy. The *resonance* discussed here is a metastable state formed by an incoming particle. In general a resonance is defined as a short-lived particle state. The dips in the cross section are a result of interference between potential and resonance scattering. A mathematical despription of this (see Ref. [23]) involves three terms, one describing potential scattering, one describing resonance scattering and one describing the interference. With $E < E_R$ the last term is negative, thus showing destructive interference. This destructive interference give rise to the dips in the cross section. A low cross section implies a long mean free path, that is neutrons travel longer distances before reacting with the shielding material.

For neutron scattering there are four primary materials to use as shielding.

- **Concrete** - A cheap hydrogen containing material. Used to shield high energy photons and to moderate fast neutrons.
- **Steel** - A cheap material. Used to stop fast neutrons. Some steels show neutron transparency for some energy ranges.
- **Plastic** - Slow fast neutrons, i.e. moderation.

- **Boron-containing substance** - A material to capture slow and thermal neutrons.

All of these materials have relevant cross section properties for neutron shielding. However, they may not be ideal for other types of radiation, such as photons. Often a combination of the materials is useful when constructing shielding for neutron sources. Koprivnikar and Schachinger [24] have studied multi-layered and sandwich designs for the ESS and stressed the need of Monte Carlo simulations with variance reduction (section 2.3).

2.2 Monte Carlo methods

2.2.1 Overview of Monte Carlo methods

The name *Monte Carlo* is given to members of a class of simulation techniques that utilize repeated and random sampling to achieve the convergence to a true distribution. This can be contrasted to traditional methods that use "abstract thinking" [26] that often result in a neat closed form expression that can give the answer to what will happen in a system under all circumstances. The prime example here would be Newton's laws of motion but even these laws will have problems making predictions already with three bodies. In particle transport problems, the real problem would have to consider millions of particles (or more). This stresses the need of a Monte Carlo method, a method to take random samples of particles to make predictions.

One of the questions considered in this thesis is how to take the samples. It might be that the *analog simulation* - the direct simulation of real physical events - is the best [27]. Another possibility is to sample different geometrical regions more or less. In the analog simulation, particles would be produced with a uniform weight of one and unevenly spread over the geometry of the simulation. In a biased simulation one goal may be to spread particles evenly over the simulated geometry. Of course it is impossible to find a method to make the geometrical spread of particles perfect, but this thesis is concerned with the problem of finding a good spread.

In a computer simulation, one would either want some specific *tallies* (measurements over the mesh grid) or a good distribution with low relative errors over the entire grid [28]. In this thesis both cases will be considered but the method presented is best suited for the latter case.

2.2.2 Error calculation in Monte Carlo methods

In all statistical models and simulations of physical processes the error is an important characteristic. The error quantifies the uncertainty of an experiment, measurement or simulation. The uncertainties in a Monte Carlo simulation arise not from instrumental difficulties but only from a limited sampling process. In this thesis, the relative error is used to quantify the end result and to compare the quality of simulation runs and also as a basis to set the weight windows.

The error is given by (9) for a random variable X with measurements i. Assuming a

normal distributed variable, the error in a given mesh cell is given by

$$\text{error} = \frac{\sigma}{\sqrt{N}}, \tag{9}$$

where

$$\sigma = \sqrt{\frac{1}{N}(\sum_{i=1}^{N} X_i^2 - \frac{1}{N}(\sum_{i=1}^{N} X_i)^2)}. \tag{10}$$

The relative error is the error relative to the the variable X itself, as in

$$Re = \frac{\text{error}}{\bar{X}} \propto \frac{1}{\sqrt{N}}. \tag{11}$$

The relative error will decrease with the square of N, i.e. the number of measurements. This means that the more events, the lower the relative error will be. This measure is not to be confused with the accuracy of the simulation. It is possible to be off the mark by a large distance, for example if the model is faulty or completely wrong, while still having a low error. A low error only refers to the precision in the Monte Carlo simulation [29]. A guideline to interpreting the relative errors can be found in table 2.

Table 2: Guidlines for interpreting relative error [29].

Range of Re	**Quality**
0.5 to 1	Not meaningful
0.2 to 0.5	Factor of a few
0.1 to 0.2	Questionable
<0.1	Generally meaningful

2.3 Variance reduction methods

This thesis will consider the *weight window* method which is a member of a broader class of variance reduction methods. These methods can be divided in two; local variance reduction (LVR) or global variance reduction (GVR). This thesis will consider GVR, where the task is to reduce the variance in all parts of a geometry. These are methods used with Monte Carlo simulations in physics to increase computational efficiency. The aim of GVR is to transport particles everywhere in a model, to catch as many events as possible and to give statistically relevant data everywhere. This has to be done while being conservative with computational power [2, 30].

The general idea will be to weight particles according to some rule. In real life or in the *analog* simulation all particles will be weighed as one. With a GVR method particles will have a weight ≤ 1. A GVR method lets events happen in a spatial-energy region where they would otherwise be non-existent or rare with a computationally feasible analog run. The question now is how to find a method to weight particles in proper ways to let them propagate through the model without sacrificing computational time.

This section gives an overview of importance sampling, as well as the weight window method itself along with a review of some common methods to set weight windows. One of the methods, Van Wijk's method [2] is used in all simulations carried out in this work.

2.3.1 Importance sampling

While this thesis will concentrate on the weight window method, importance sampling is the most commonly used method for simulation of rare events. Instead of sampling the direct distribution ($p(\cdot)$), to estimate the expected value of $f(X)$ for the random variable X, the sample is taken from

$$E[f(X)] = \frac{1}{k}\sum_{i=1}^{k} f(Y_i)\frac{p(Y_i)}{q(Y_i)}, \tag{12}$$

where $p(\cdot)$ is the PDF for X, $q(\cdot)$ is the importance distribution, Y are i.i.d. (independent and identically distributed) random numbers and k is the number of sampling points [31]. (12) is an unbiased estimate of the expected value of X[3].

It is reasonable to ask why this is useful. Different choices of $q(\cdot)$ result in different variances. There is an optimal choice of $q(\cdot)$ but this is unattainable for any practical purpose. The general idea would be to choose this distribution to hit the rare events more often [32].

For the particular problem of particle transport, the sampled distribution is the neutron flux in the geometry where some regions are more populated and some less. In this geometry, importance sampling of the distribution is formed by dividing the geometrical regions in cells and assigning an importance to each one of them. When a particle crosses the boundary between a cell A and another cell B, then $r = \frac{I_B}{I_A}$, where I is the importance of the cells. This fraction may be exactly one, in which case the particle carries on. If $r < 1$ tracks are 'killed' (their track is removed from the simulation) with probability $1-r$ and weights of the remaining particles are adjusted to compensate for this. If $r > 1$ and r is an integer the particle tracks are split into r tracks and particle weights are adjusted to keep the simulation results physical. If r is not an integer then the particle is split into $\text{int}(r) + 1$ tracks with probability $r - \text{int}(r)$ and into $\text{int}(r)$ tracks with probability $1+\text{int}(r)-r$ [1]. The int() function truncates its input value to the integer value. This will create an unbiased distribution with a lower variance according to (12). The discussion of how to set the importance is beyond the scope of this thesis.

2.3.2 Weight windows

A sophisticated variance reduction method - and the one that is going to be used - is the weight window method [1]. This method works on cells created in the spatial-energy dimensions, hence the cells are four dimensional and stretch out in energy and in the three spatial dimensions.

[3]At least this holds true if $\text{supp}(p \cdot f) \subset \text{supp}(q \cdot p)$ where, in general, $\text{supp}(g(x)) = \{x \in X | g(x) \neq 0\}$.

The basic idea is to split[4] the particles that are higher weighted than the window and to *Russian roulette* particles that are below the window. The split particle results in multiple particles with a weight lower than the mother particle. The particles that have to play the game of Russian roulette are either killed or they survive and are moved up to the survival weight. The point with this is to produce a computer simulation that gives flux information over the whole system rather than just some cells [33].

Having particles with weights equal to one would result in too many particles in regions with high flux and too few particles in regions with lower flux. Too many would mean that unnecessary computer power is diverted to a problem that already has significant results. Too few would mean that not enough particles are sampled to give meaningful results. Instead of - as in the analog case - having varying numbers of Monte Carlo particles over the system, the GVR method aims to even out the number of Monte Carlo particles but still keeps the 'real' particle proportions by varying the particle weights [33]. In summary, a particle entering a cell in the simulation will follow one of these three rules:

- The particle weight is higher than the weight window, in this case the particle will be split and weighted to be inside the window.
- The particle weight is in the weight window, this is fine and nothing happens.
- The particle weight is below the weight window. In this case *Russian roulette* is played to either kill the particle or move it to the survival weight.

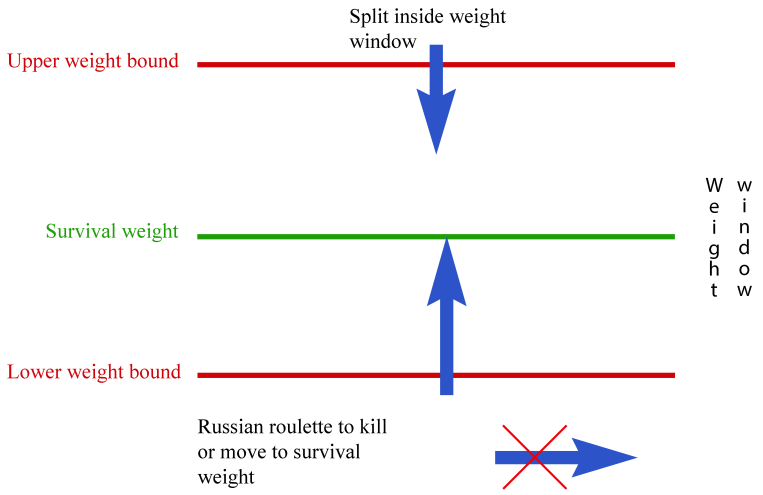

Figure 5: The principle behind weight windows. Particles with a high weight are split to particles inside the window. Particles with a low weight are either removed from the simulation or moved inside the window [1].

Figure 5 shows a summary of this. This can also be illustrated using a simple example, consider Figure 6. This simulation consists of one weight one particle and one weight

[4]The Geant4 manual states that particles are split to the survival weight. However, in the G4WeightWindowAlgorithm the line that would perform this is commented out and replaced with a line that splits to the upper weight bound [1].

0.7 particle entering a first cell. The weight window is set with lower boundary 0.3 and higher boundary 0.8 and the survival weight at 0.5. The particle weighted 0.7 is inside the window and nothing will happen to it. The particle above the window will split in two particles now weighted 0.5. The three particles will enter the second cell that has another window set, with the lower boundary at 0.6, and the higher boundary at 1.1 and the survival weight at 0.8. The particle with weight 0.7 is still in the window and nothing happens. The two particles with weight 0.5 are under the window and will either be removed from the simulation or moved to the survival weight (0.8). In figure it is assumed that one survives and that the other one is killed.

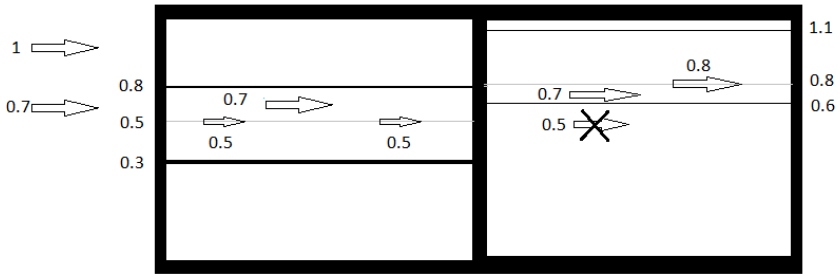

Figure 6: An example of two particles traveling in a two cell weight window mesh. This figure shows splitting and killing of particles traveling through a very simple system of cells.

This method is built in Geant4 and requires the user to set a lower weight bound for the whole problem as well as upper weight factors and survival factors, so that

$$W_U = C_U \cdot W_L \tag{13}$$

$$W_S = C_S \cdot W_L. \tag{14}$$

The user also has to define a mesh of cells. In Geant4 this mesh will be set in a *parallel geometry*, that is the cells will exist in a parallel world to the simulation geometry. In the simulation, particles are transported through the simulation geometry and then also the parallel geometry, which is a grid of cells overlapping with the mass geometry.

In (13) the upper weight bound is constructed from the user given parameters and in (14) the survival weight is constructed. The upper limit factor and the survival factor (C_U and C_S) are set for all windows and are used to calculate the upper weight and the survival weight (W_U and W_S). The lower weight (W_L) is set for each cell. This results in equally sized windows that are just being higher or lower over the mesh grid [5].

2.3.3 Some particular weight window generator methods

The previous section discussed what weight windows are and how they are used. What was not discussed was how to set the weights. For this a generator method is needed, which provides a rule to set the windows in all the cells in the mesh. This could be performed manually but is a tedious task and not a very reliable method. Instead it is useful to have an automated method. The following section will discuss some alternative

weight window generators, described in [28].

The first of these method is Cooper's method. The ideas of Cooper and Larsen [33] are based on the following reasoning; for each cell i the density of particles (N_i) is proportional to the fraction total weight (W_i) of all Monte Carlo particles in i over the volume of the cell (V_i), as in (15). But the density of particles is just the mean weight of those particles (f_i) times the number of Monte Carlo particles in that cell (M_i), i.e.

$$N_i \propto \frac{W_i}{V_i} = \frac{f_i M_i}{V_i}. \tag{15}$$

Then choose the center of the weight window ($= f_i$) to be proportional to N_i to get

$$f_i \propto N_i \Rightarrow \frac{M_i}{V_i} = \text{constant}, \tag{16}$$

or in more useful terms (with flux related to the number of particles and the weight to the mean weight), let the weights of the particles be proportional to the flux in the cell as in

$$w(\vec{r}) \propto \phi(\vec{r}). \tag{17}$$

This means that the density of Monte Carlo particles is evenly spread over the system, because the center of the weight window is proportional to the density of particles, which in turn implies a good distribution of computational power. Also note that this method is a hybrid deterministic method, i.e. a method that uses an algebraically estimated flux as a basis for the weight windows. This is in contrast to, for example, Van Wijk's method that uses pre-simulations to calculate flux and hence is a non-deterministic method [28, 33].

Van Wijk's method [2] is the primary method used in this work. It builds upon Cooper's method and is a non-deterministic method to use flux or the relative error of the flux to set windows in an iterative sequence. The method is described in detail in section 2.3.4 [28].

GFWW (Global Flux Weight Windows) and GRWW (Global Response Weight Windows) are two extensions of Cooper's method. Cooper's method works on spatial cells, whereas the GFWW and GRWW methods give windows as a function of spatial and energy positions. The GFWW method will give an even distribution of non-analog particles over energy and space. The weight window is set proportional to the forward scalar flux - as before - but is energy dependent. Windows are hence set as in (17) but with an energy dependence.

The GRWW methods are to be used when one wants a response (biological dose is given as an example [34]) over the whole geometry integrated over all energy regions. The problems considered in this thesis aim to calculate neutron flux, hence the GFWW would be the potential alternative. There are also an important group of hybrid methods, including the FW-CADIS (Forward Weighted-Consistent Adjoint Driven Importance Sampling) as the most known [28].

In [28] there is a comparison of these methods. The conclusion there is that Van Wijk's

performs worse on the harder problems (the problems with higher flux differences over spatial regions) and that overall FW-CADIS has the best performance. In this paper only the relative error based method is tested. FW-CADIS is however more complex to implement than for example van Wijk's method.

2.3.4 An easy to implement global variance reduction procedure

In the previous section, the general weight window method was described, and was followed by a short overview of methods on how to set weight windows. This section provides a deeper description of one of these methods. Furthermore, none of these methods are implemented in Geant4 and to explore this is one of the main points with this thesis.

The ideas in this section are based on two papers; one by Cooper and Larsen [33] and the other by Van Wijk, Van den Eynde and Hoogenboom [2]. The first paper shows that it is indeed possible to use the forward scalar flux (see section 2.3.3) to set weight window thresholds and the other one gives an - in MCNP - implemented method to use.

This method will take one of two paths, either we give birth to particles in regions with the highest forward scalar flux according to

$$W_L = (\frac{C_U + 1}{2})^{-1} \frac{\text{Min}(\vec{Re})}{Re_i}, \tag{18}$$

or in the cells with the lowest relative error according to

$$W_L = (\frac{C_U + 1}{2})^{-1} \frac{\phi_i}{\text{Max}(\vec{\phi})}. \tag{19}$$

In equation (18) and (19) ϕ refers to the flux, Re the relative error (with their respective vectors $\vec{\phi}$ and $\vec{Re}$), W to the lower weight threshold and C_U to the upper limit factor defined in section 2.3.2.

The W_L are thus proportional to flux or inversely proportional to the relative error. To motivate the factor $(\frac{C_U+1}{2})^{-1}\frac{1}{\text{Max}(\vec{\phi})}$ take the region with the highest flux. In this region particles are born and their weights are equal to one. Let those particles be born in the middle between the weight window boundaries. Then the particles will not be split or killed. The factor places the weight window for the highest flux cells with one in the middle, because for the highest flux cells; it is set to

$$W_L = (\frac{C_U + 1}{2})^{-1}, \tag{20}$$

and the midpoint can be expressed as

$$\frac{W_U - W_L}{2} + W_L = \frac{C_U \cdot W_L + W_L}{2} = \frac{C_U}{C_U + 1} + \frac{1}{C_U + 1} = \frac{C_U + 1}{C_U + 1} = 1. \tag{21}$$

Hence, using this factor the windows will - according to (20) and (21) - have the midpoint at one for the cell(s) with the highest flux. The same reasoning is applied to the case of the relative error of the flux.

The flux in these equations is given by an analog run, that is a run without any weight windows. This simulation runs with only 'real' particles, i.e. particles with a weight of one. The flux based weight window generating method can however be used iteratively. That is, one analog run generates the flux for a first weight window run that in turn generates flux for a second run and so on. This can be repeated until a proper distribution of relative errors is reached. The relative error based method will flatten out and is not possible to use in this manner.

If a cell has flux exactly equal to zero, the method described so far has no rule to handle this. There are two suggested strategies to solve this. For the method based on the relative error, the cells without particles are set to have a relative error at 100%. For the flux based method, the cells are skipped and no window is set. This will mean that all particles move through it without any splitting or killing.

The efficiency is to be quantified in two ways, using two different figures of merit (FOM). The first one (22) is calculated using the mean of the squared relative errors of the cells and the computational time. These are quantities that should be minimized and since the expression is inverted the FOM should be made as large as possible using the weight window procedure. The second FOM (23) will simply be the standard deviation of the relative errors, a figure that should be made small using this procedure. FOM_1 is defined as

$$FOM_1 = (\sum_{i=1}^{N} \frac{Re_i^2}{N} \cdot T)^{-1}, \tag{22}$$

and FOM_2 is defined as

$$\sigma_{re} = FOM_2 = \sqrt{\frac{1}{N}(\sum_{i=1}^{N} Re_i^2 - \frac{1}{N}(\sum_{i=1}^{N} Re_i)^2)}. \tag{23}$$

This method is implemented in MCNP by the authors of the paper and the goal is to do the same in Geant4. The above gives a strategy to set the weight windows for each cell, shows how to handle cells with a zero flux in the analog run and gives a figure of merit to use for testing and comparing the model [2].

3 Simulation analysis

3.1 Realization of the method in Geant4

The method described in the previous sections was implemented and tested in Geant4. First using a test geometry with particles traveling in two dimensions and with cells in one dimension and then with the two models of the ESS target. Most of the relevant code is within one class, ImportanceDetectorConstruction. The method was tested and made sure to give reasonable and good results before being applied to the target model itself.

Some useful terminology in this context is *track*, *event*, *run* and *hit*. A track is in this context the current information about the particle (see Ref. [1], G4Track). An event in

this simulation refers to the actions of one primary particle and its secondaries. A run is a collection of events. A hit is a detector registration [1].

In a Geant4 model there is at least one tracking or mass geometry containing the objects to be modeled. Besides this there can exist a parallel geometry. In this geometry, it is possible to define overlapping objects (otherwise forbidden), sensitive regions (i.e. detectors) or weight maps [1].

Included in the target model with the weight window method are the following classes:

- **EventAction** - Contains two methods, beginOfEventAction() and endOfEventAction(). They are invoked at the beginning and end of each event. This class contains calculations done on a per event basis, for example the error calculations and writing to histograms are done here.
- **ImportanceDetectorConstruction** - This class sets up a parallel geometry and weight windows when created.
- **TrackerHit** - For each event many hit objects are processed. In this simulation this class handles output based on hits and draws tracks in the visualization manager.
- **TrackerSD** - This class determines the behavior of the detectors. For a user of biased simulations, it is important to take care and make the detectors sensitive to particle weight.
- **ActionInitialization** - This class initializes the run and creates objects of the classes that are used.
- **DetectorConstruction** - This class sets up the mass geometry.
- **PrimaryGeneratorAction** - Gives the parameters for each event, i.e. sets up the proton beam.
- **PVolumeStore** - A registry that contains a list of all physical objects in the simulation.
- **Run** - Initializes maps and collections needed during a run and records events.
- **RunAction** - This is similar to EventAction but handles calculations performed on a per run basis.
- **QGSP_BERT_HP_TS** - This is a *physics list* based on the QGSP_BERT_HP list and determines what physics models Geant4 will use in the simulation. This particular list uses thermal scattering data under 4 eV, high precision neutron data below 20 MeV, the Bertini cascade model for energies up to ≈ 10 GeV and the QGS model for higher energies. This list is suitable for shielding problems and is a good choice when modeling a spallation source process [1].

As a part of this project all of these classes except for ActionInitialization, PVolumeStore and Run have been modified. ImportanceDetectorConstruction has been fully implemented and major changes where made to both RunAction (to keep track errors and

store information between runs) and also to DetectorConstruction (the model were modified to that of Figure 9).

The above classes are derived from other classes in Geant4, for example the EventAction derives from G4UserEventAction and TrackerHit derives from G4VHit [1].

The relevant code from the ImportanceDetectorConstruction class (Code 1) places a given number of rectangles in a parallel geometry. This can be adapted to any geometrical size of a model and uses this size as well as the number of cells in the three spatial dimensions as parameters.

Code 1 gives an example of how physical objects are created in Geant4. G4Box is one of the solids and contains the dimensional information. The *logical volume* takes the solid and a material to create an object that can be placed [1]. This is done in line 11. Lines 30-36 place this volume in the world at the given coordinates.

Code 1 Set up of mesh grid, in ImportanceDetectorConstruction

```
1
2   G4double BoxX=worldXY/cellsX/2;
3   G4double BoxY=worldXY/cellsY/2;
4   G4double BoxZ=worldZ/cellsZ/2;
5
6   //Box solid
7   G4Box *aShield = new G4Box("aShield", BoxX, BoxY, BoxZ);
8
9
10  //Creating the logical volume
11  G4LogicalVolume *aShield_log_imp =
12    new G4LogicalVolume(aShield, dummyMat, "aShield_log_imp");
13  fLogicalVolumeVector.push_back(aShield_log_imp);
14
15  // physical parallel cells
16  G4String name = "none";
17  G4String Detectorcellname="none" ;
18  G4int i=1,j=1,k=1,ijk=0;
19  G4double startx = -worldXY/2+BoxX;
20  G4double starty = -worldXY/2+BoxY;
21  G4double startz = -worldZ/2+BoxZ;
22  //  for (i=1; i<=18; ++i) {
23  for (i=1; i<=cellsX; i++) {
24    for (j=1; j<=cellsY; j++) {
25      for (k=1; k<=cellsZ; k++) {
26        name = GetCellName(ijk);
27        G4double pos_x = startx + BoxX*(i-1)*2;
28        G4double pos_y = starty + BoxY*(j-1)*2;
29        G4double pos_z = startz + BoxZ*(k-1)*2;
30        G4VPhysicalVolume *pvol =
31          new G4PVPlacement(0,
32                    G4ThreeVector(pos_x, pos_y, pos_z),
33                    aShield_log_imp,
34                    name,
35                    worldLogical,
36                    false,
```

```
                        ijk);
          G4GeometryCell cell(*pvol, ijk);
          fPVolumeStore.AddPVolume(cell);
          ijk++;
        }
      }
    }
```

The loop in code 2, which is given in method CreateWeightWindowStore() in the class ImportanceDetectorConstruction, iterates over the created cell geometry and sets the weights of all the cells according to (18) and (19). The commented lines are the relative error based method and the corresponding active lines are used for the flux based method.

Code 2 The weight window procedure, in ImportanceDetectorConstruction

```
  for (cell=0; cell<=noCells-1; cell++) {
    G4GeometryCell gCell = GetGeometryCell(cell);
    lowerWeight=(2./(beta+1.))*previousFlux[cell]/maxFlux;
    // lowerWeight=(2./(beta+1.))*minError/relativeError[cell];
    if(!lowerWeight || isinf(lowerWeight)) {
      lowerWeight=(2./(beta+1.))*minFlux/maxFlux;
      // lowerWeight=(2./(beta+1.))*minError/maxError;
    }
    lowerWeights.clear();
    lowerWeights.push_back(lowerWeight);
    wwstore->AddLowerWeights(G4GeometryCell(gCell.GetPhysicalVolume(),cell)
        , lowerWeights);
```

The error calculations in section 2.2.2 are not implemented by default in Geant4. This has to be done manually for the weight window procedure to work properly and is especially true for the relative error method as this method uses the error as input.

Calculations are done both on a per run (one full iteration of the simulation) basis and a per event basis, as shown in codes 3 and 4. Any user of the weight window method that needs the error calculations would have to insert this in the pre-existing code. One might choose to exclude these on reasons of computing time, however when applying the relative error based method the error calculations must be included.

Code 3 Error calculation, in RunAction

```
    G4int nofEvents = aRun->GetNumberOfEvent();
    if (nofEvents == 0) return;
    G4double sqrt_n = std::sqrt(nofEvents);
    std::map<G4int,G4double> RsumTE  = B2EventAction::Instance()->sumTE;//
        sum in RunAction
    std::map<G4int,G4double> Rsum2TE = B2EventAction::Instance()->sum2TE;
    std::map<G4int,G4double> RsumWE  = B2EventAction::Instance()->sumWE;//
        sum in RunAction
    std::map<G4int,G4double> Rsum2WE = B2EventAction::Instance()->sum2WE;
    std::map<G4int,G4double> sigma2TE,sigma2WE,sigma2tot,errorTE,errorWE,
        error;
    std::map<G4int,G4double>::iterator itrTE = RsumTE.begin();
    //std::ofstream ofile("error.txt");
    //if(!ofile) return;
    for(; itrTE != RsumTE.end(); itrTE++) {
      i = itrTE->first;
```

```
14        sigma2TE[i] = Rsum2TE[i]/nofEvents-(RsumTE[i]/nofEvents)*          (
              RsumTE[i]/nofEvents);
15        sigma2WE[i] = Rsum2WE[i]/nofEvents-(RsumWE[i]/nofEvents)*(RsumWE[i]/
              nofEvents);
16        G4double *WeightEnterre = (*WeightEnter)[i];   //TAKING WEIGHT ENTER
              IN EACH CELL
17        if(!WeightEnterre) WeightEnterre = new G4double(0.0);
18        G4double *TrackEnterre = (*TrackEnter)[i];   //TAKING TRACK ENTER IN
              EACH CELL
19        if(!TrackEnterre) TrackEnterre = new G4double(0.0);
20        if (sigma2TE[i] > 0.) sigma2TE[i] = std::sqrt(sigma2TE[i]);   //Going
              from sigma^2 to sigma
21        else sigma2TE[i] = 0.;
22        errorTE[i] = sigma2TE[i]/sqrt_n;    // Applying error formula
23        errorTE[i] = errorTE[i]/RsumTE[i]*nofEvents;
24
25        if (sigma2WE[i] > 0.) sigma2WE[i] = std::sqrt(sigma2WE[i]);   //Going
              from sigma^2 to sigma
26        else sigma2WE[i] = 0.;
27        errorWE[i] = sigma2WE[i]/sqrt_n;    // Applying error formula x*sigma/
              sqrt(n)
28        errorWE[i] = errorWE[i]/RsumWE[i]*nofEvents;
29    }
```

Code 4 Error calculation in EventAction

```
1  G4int colIDTE = SDMan->GetCollectionID("ConcreteSD/TrackEnter");
2    G4int colIDWE = SDMan->GetCollectionID("ConcreteSD/WeightEnter");
3    G4THitsMap<G4double>* evtMapTE = (G4THitsMap<G4double>*)(HCE->GetHC(
         colIDTE));
4    G4THitsMap<G4double>* evtMapWE = (G4THitsMap<G4double>*)(HCE->GetHC(
         colIDWE));
5
6    std::map<G4int,G4double*>::iterator itrTE = evtMapTE->GetMap()->begin();
7    for(; itrTE != evtMapTE->GetMap()->end(); itrTE++)
8      {
9        sumTE[itrTE->first] += *(itrTE->second);
10       sum2TE[itrTE->first] += (*(itrTE->second))*(*(itrTE->second));
11     }
12    std::map<G4int,G4double*>::iterator itrWE = evtMapWE->GetMap()->begin();
13    for(; itrWE != evtMapWE->GetMap()->end(); itrWE++)
14     {
15       sumWE[itrWE->first] += *(itrWE->second);
16       sum2WE[itrWE->first] += (*(itrWE->second))*(*(itrWE->second));
17     }
```

When changing the geometry, the problem remains similar, except for the scale and the dimensions. The number of cells increases exponentially with dimensions as there will be N^d cells where N is the number of cells along an axis and d is the number of dimensions. A map of the geometry with a replica of the mesh grid is shown in Figure 7. The method was used on two target models. For the first one, Figure 8, the radius of the cylinder is 3 m. Figure 9 shows the second model which has a radius of 5.5 m and a ring of concrete surrounding the inner geometry. This ring is 0.5 m thick.

The models include a water filled pre-moderator, a liquid hydrogen moderator, a beryllium

reflector, a steel reflector, iron shielding (the large cylindrical structure) and in Figure 9 also the concrete ring. The overview of the ESS target region in Figure 2 highlights the important components in figures 7 to 9. The wing like structure in Figures 8 and 9 are extraction areas for neutron beamlines [3].

The weight window method is constructed to be general and could with a few changes be applied to another Geant4 geometry. The main() function would need a few lines to call and set up the weight window-algorithm and the error calculations are made outside the ImportanceDetectorConstruction class (scoring of flux is done inside the class). Also, the FOM in (22) is calculated outside the class because the time of the whole run is needed. These calculations demand some code in the RunAction and EventAction classes as described above. Besides this, all of the code used for biasing is contained within the ImportanceDetectorConstruction class.

A user of the class has to call the constructor giving values for C_U and C_S. For all simulations presented here, the values used were $C_U = 5$ and $C_S = 2$.

The data presented in the coming sections were produced using the weight window method and the ESS target models. It was made presentable using ROOT [35] and some additional scripts.

There is no easy and documented way to turn off a weight window in a cell in Geant4 (unlike MCNP) and the suggested strategy [2] to handle cells with no flux had to be abandoned in favor of a simpler rule, where the cell with the lowest (but existing) flux sets the window for the cells without a flux. A similar approach was adopted for the relative error method. Another alternative would be using a mean of neighboring cells to set the weight window for a cell without a flux in the previous run. This, however, is a time consuming and complex task if there are many empty cells.

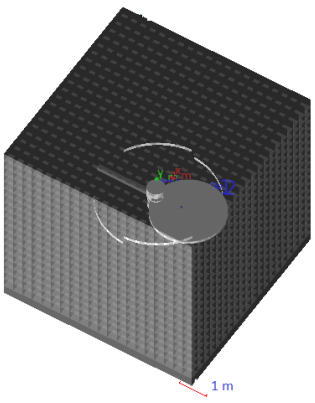

Figure 7: A mesh grid overlay on the ESS target model.

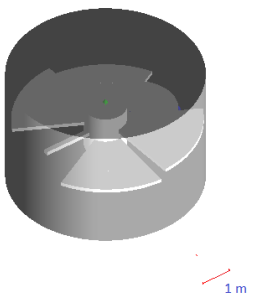

Figure 8: The ESS target model.

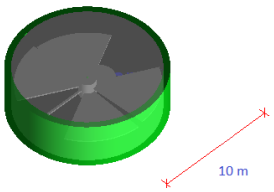

Figure 9: The extended ESS target model. The green part shows the concrete ring.

All tests performed in this thesis are carried out on a regular laptop. This is not much computing power in this context and does constrain the simulation parameters. A simu-

lation like this would benefit from running on a computer cluster, as this would increase the simulation times and produce results faster. The principle would not change and all results would carry over to such a simulation..

The setup of the cells and weight windows take about 5 min using this configuration. This number is dependent on the number of cells. The comparatively low number of cells is chosen with this in regard. This time is excluded from all run times, figures of merit and such. This is because this would make it harder to extrapolate results to longer run times. Using relatively short simulations, as many of the cases in this thesis, the additional five minutes would be a small fraction of the total time for a longer simulation.

3.2 Results

Figures 10 to 19 show the flux per primary proton and the relative error of the flux for the simulations. The maps in Figures 10 to 13 are based on data generated from an analog simulation with run times of 955 s and 47 385 s. These are to be compared with Figures 14 to 17 showing the same data but for a simulation using the flux based method. Figures 18 and 19 show the relative error results from the relative error based method.

These figures are generated using a $31 \times 31 \times 31$ mesh grid. The figures show a plane with data taken from the middle (cell number 16 in the z direction) cells of the x-y-plane in the target model. The axes represent spatial coordinates and the color bars show particle flux and the calculated error respectively. The figures also show an overlay of a 2D projection of the geometry of the simulation (compare to Figure 2). N.B. the scales for the figures showing flux are shown in logarithmic scale.

The analog method does not populate the whole geometry in the alloted simulation time. Both of the weight window methods manage to do that. The flux based method shows very little spread of relative errors after the simulation is complete.

The evolution of flux using the flux based method shows that all cells in this plane get a flux after about 7000 s, which is not the case for the analog maps, where there is no flux in some cells during the whole simulation. A comparison between the flux based method and the relative error based method shows that the flux based method leads to overall improvement of the relative error in the cells. In particular, for cells in the outer parts of the geometry, there are lower relative errors. One could also note that in the center of the geometry, the relative error is actually higher than in the analog case. This is a pattern also mentioned in [2].

Figure 20 shows the number of cells without any flux after a given time for the three methods. The flux based method has no zeros after just two iterations. For a more complex model, the shape is more similar to the analog case (see figure 35).

In Figures 22 and 21 the average track weight in a typical outer region cell is shown. For the flux based method, the initial update for the wight window overshoots and the window is slightly updated during the whole simulation. One hypothesis would be that it grows asymptotically to a value where it would be stable. The error based weights are

updated more rarely and show a flat pattern between updates.

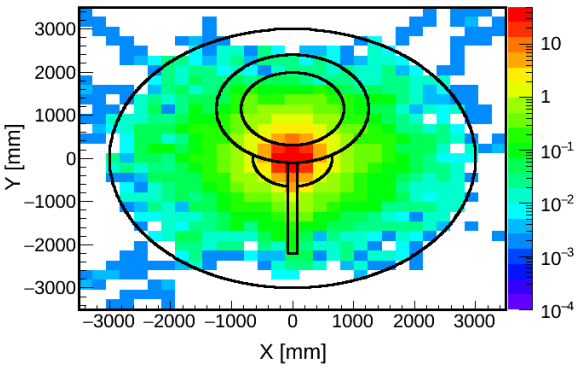

Figure 10: Flux map of an early analog simulation (955 s).

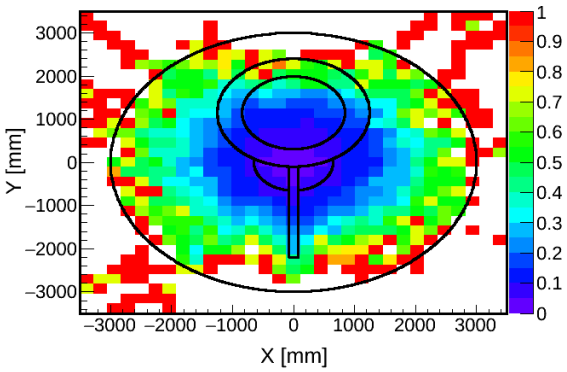

Figure 11: Relative error map of an early analog simulation (955 s).

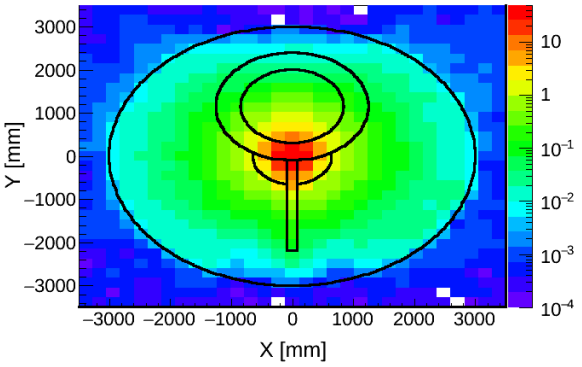

Figure 12: Flux map of a late analog simulation (47 385 s).

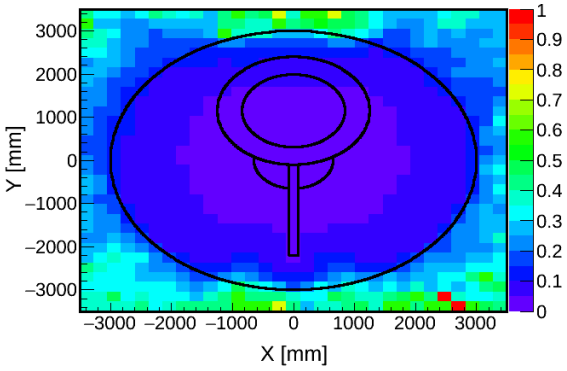

Figure 13: Relative error map of a late analog simulation (47 385 s).

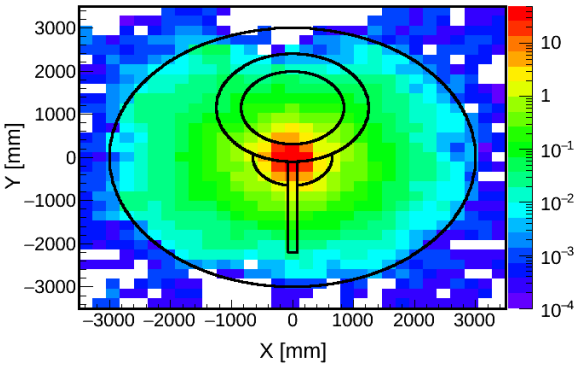

Figure 14: Flux map of first weight window simulation (896 s).

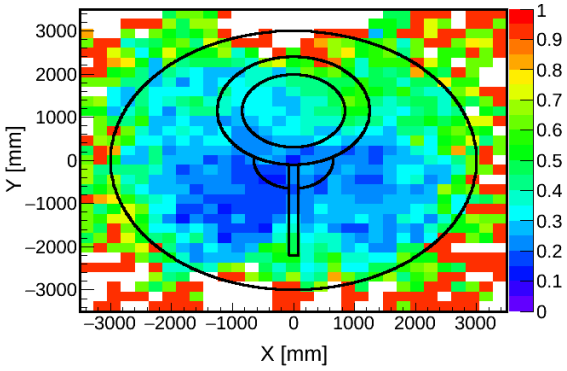

Figure 15: Relative error map of first weight window simulation (896 s).

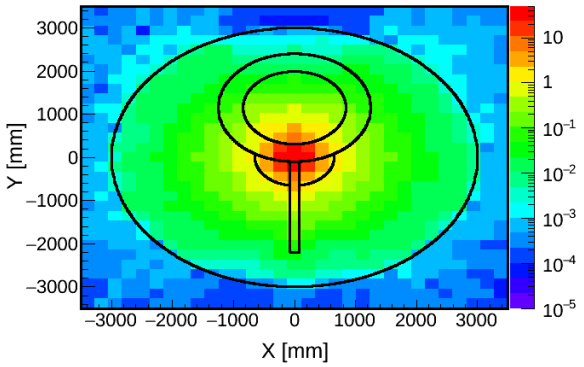

Figure 16: Flux map of the ninth weight window simulation (47 394 s).

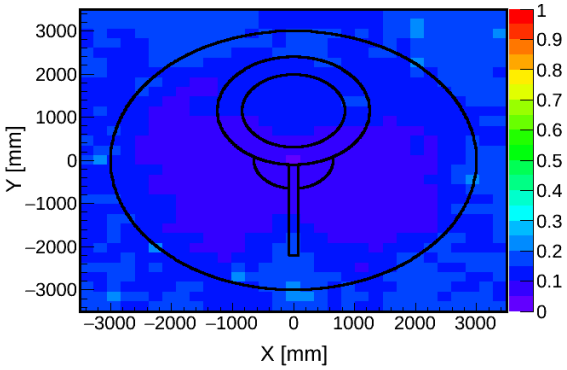

Figure 17: Relative error map of the ninth weight window simulation (47 394 s).

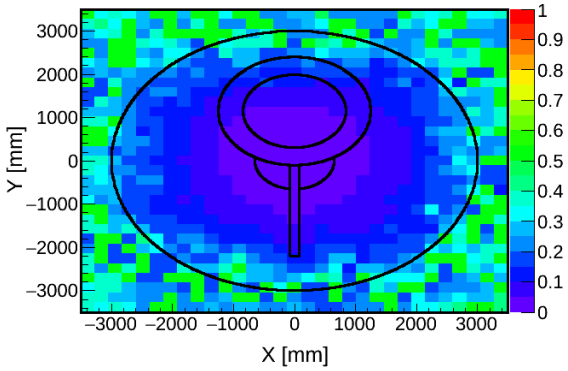

Figure 18: Relative error map of third relative error based run (17 511 s).

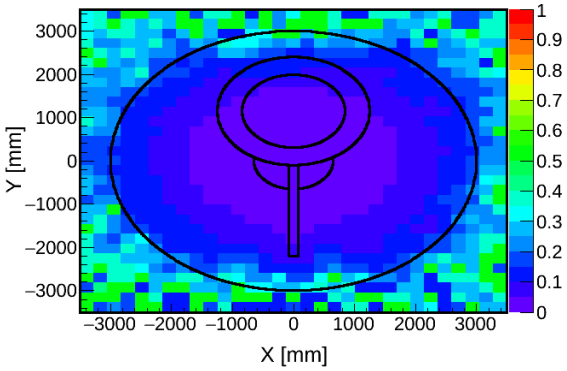

Figure 19: Relative error map of eight relative error based run (46 532 s).

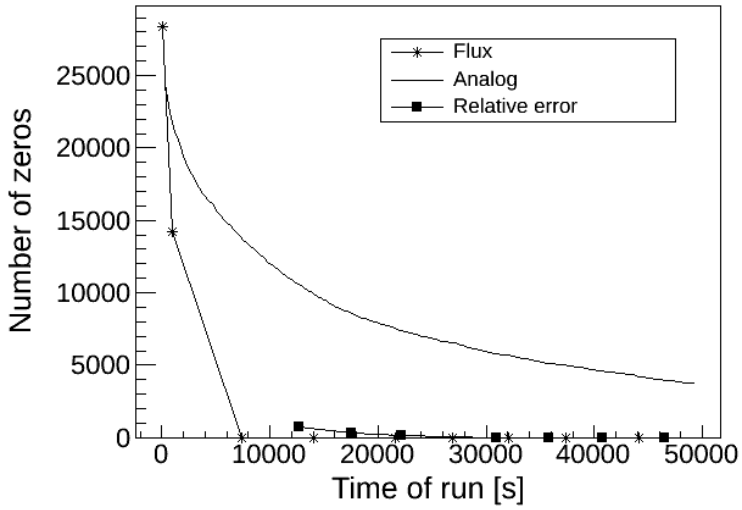

Figure 20: The number of cells that see no flux in the small target model using the two methods as well as in the analog simulation plotted against the simulation time.

In figures 23 and 24, the figures of merit are given for some run times. For the weight window methods all measured points are marked. The analog run is shown as a line without marks. For the analog run there are 209 points spaced around 300 s apart. Both methods show improvements over the analog simulation. FOM_1 decreases steadily; with an increasing number of events there is less chance to hit the cells that would give payoff in terms of reduced error. Hence, the more computer time that is diverted to the problem, the less the payoff. The weight window methods show a higher and more stable FOM_1.

The second figure of merit is lower for the flux based method and both this and the analog case show fluctuating figures of merit but without trend. For the relative error based method, the trend in the figure of merit is decreasing.

The improvements in the figures of merit are shown in table 3. The figure of merit that has the highest time dependence is FOM_1 for the analog method. Table 3 show comparisons of the weight window method with the FOM for the analog case. The difference in the results in this table are mostly a result of changes in the FOM for the analog case. The figure of merit for the flux method is stable while it is steadily decreasing for the analog simulation (as well as for the relative error based simulation). After the whole simulation, the flux method shows the highest increase in FOM_1 relative to the analog simulation while the relative error based method decreases FOM_2 relative to the analog simulation the most. FOM_1 would be the preferred measure as this takes computing time into consideration and is more stable; at least after some time when the shape of the analog FOM_1 graph is sufficiently flat. This also shows that the flux based method is better at actually producing lower errors over all.

Table 3: Improvements compared to the analog simulation in FOM_1 and FOM_2 for the two methods for some simulation times.

Flux method			**Relative error method**		
Time	FOM_1	FOM_2	Time	FOM_1	FOM_2
14092	$\times(5.62 \pm 0.04)$	$\times(3.15 \pm 0.01)$	17510	$\times(2.17 \pm 0.01)$	$\times(1.33 \pm 0.01)$
32112	$\times(11.50 \pm 0.01)$	$\times(2.54 \pm 0.01)$	35763	$\times(3.11 \pm 0.01)$	$\times(2.33 \pm 0.01)$
44087	$\times(14.60 \pm 0.02)$	$\times(2.64 \pm 0.01)$	46532	$\times(3.08 \pm 0.01)$	$\times(2.89 \pm 0.01)$

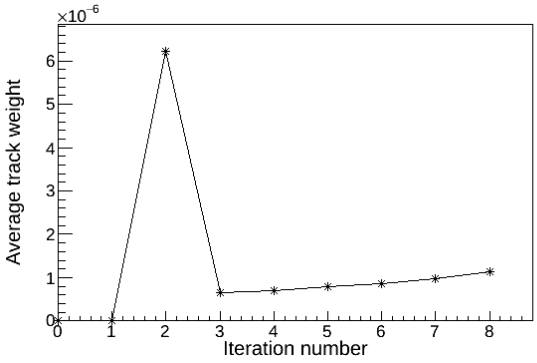

Figure 21: The average track weight of a typical cell in the small geometry's outer regions using flux based method.

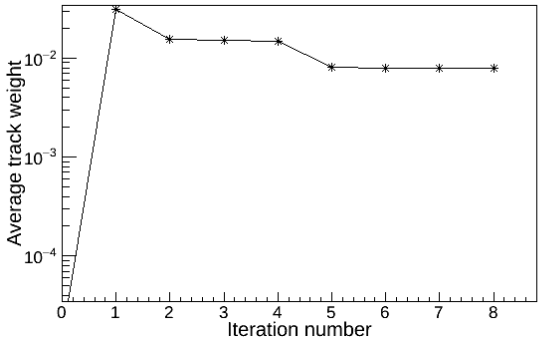

Figure 22: The average track weight of a typical cell in the small geometry's outer regions using relative error based method.

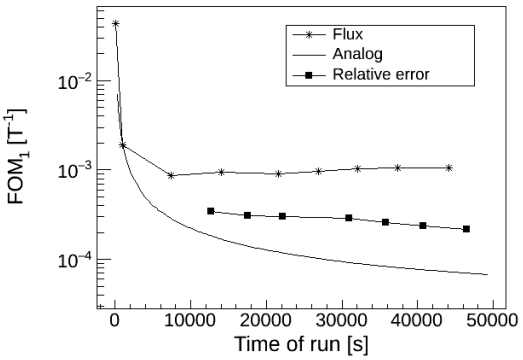

Figure 23: A comparison of the methods efficiency when applied to the small model using FOM_1.

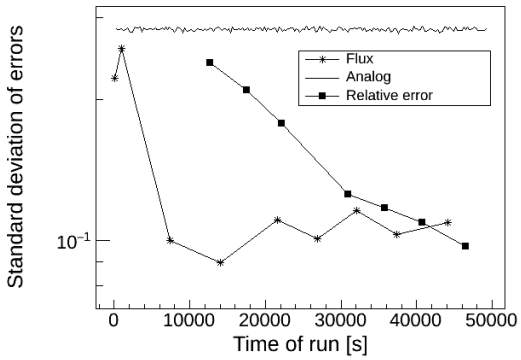

Figure 24: A comparison of the methods efficiency when applied to the small model using FOM_2.

Figure 25 show the cumulative histogram of the relative errors in the cells. The flat profile of the flux based method is pronounced, most of the cells are in the same error range. The analog simulation has cells that see just a few particles, the steps are longer for the high error range. Figure 26 shows the time evolution number of cells with an error ≤ 0.1 for the different methods. The number of cells with a reasonable error will increase with run time. It is clear that the flux based method performs best for runs longer than about 40 000 s and improves much faster than its counterpart.

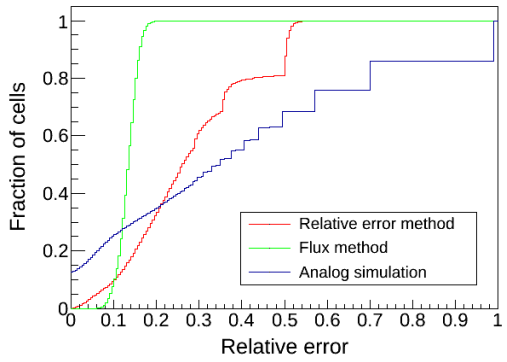

Figure 25: A cumulative histogram of the fraction of cells with a given relative error when applying the two methods, as well as the analog simulation, on the small mode.

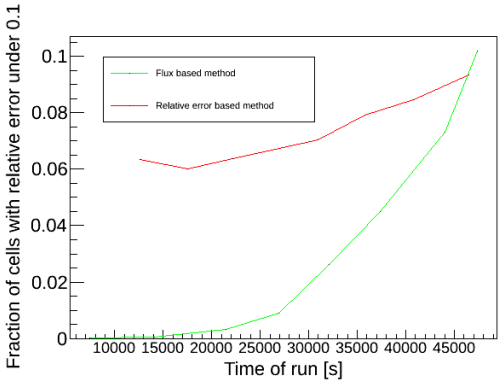

Figure 26: Time evolution of cells with relative error ≤ 0.1, when applying the two methods on the small model.

As one possible application of the method is to calculate energy spectra, the energy spectrum produced using the method were compared to the spectrum produced by the analog simulation. The energy spectrum in Figure 27 is generated from the four detectors shown as bent stripes in Figure 7. The detector is located at a distance of 2 m from the moderator.

The lower section ranging from 10^{-10} MeV to 10^{-6} MeV are the moderated spectra. Peaks in this range can be related to the Maxwellian distribution (7). These are neutrons slowed

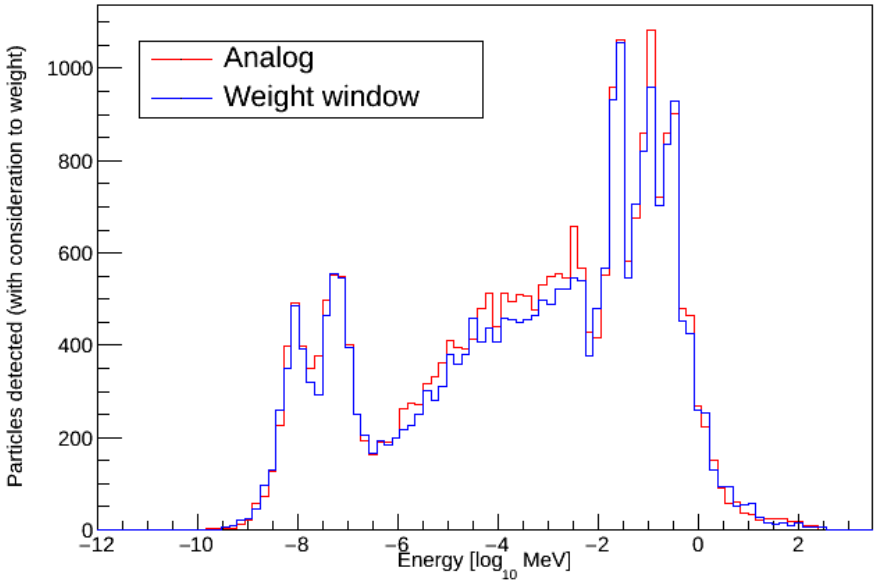

Figure 27: A comparison between the analog and biased spectra in the detectors. The figure shows neutron hits in all the detectors with consideration to particle weight. The spectra are normalized (by setting the maximum peak for the weight window spectrum to to maximum value for the analog spectrum) for easy comparison.

down by the liquid hydrogen and water moderators. The higher peaks come from the transparency in the steel and iron shielding. Figure 3 shows the cross section of iron and its transparent part can be related to the peaks in Figure 27. The higher energy neutrons in the spectrum come from direct spallation products. These may be in the range up to the incident proton beam, see section 2.1.2.

The flux of the small model spanned 5 orders of magnitude. The flux in the large model spans 11 orders of magnitude. The problem remains the same, even if the geometry is slightly changed, but this would be considered a "hard" problem [28]. It is clear that the simulation has to be run longer in this case to produce relevant results, even if using weight windows.

Figures 28 and 29 show the relative errors of the analog simulation and the flux based method. The flux based method produces a flatter map of relative errors but it is not evident that the errors are generally lower than in the analog case. Close to the center of the model the analog case shows lower relative errors. The analog simulation however, has many empty cells, which is not the case for the simulation run with the weight window method.

The shape of FOM_1 for the flux method now visibly follows the analog shape, as seen in Figure 30. The improvements in the figures of merit are smaller. Table 4 shows the improvements in the two figures of merit. FOM_1 shows a trend to a relatively higher figure for the weight window method. It is plausible that this trend will go on as in the case of the smaller model. For FOM_2, in Figure 31 it is possible that it follows the same

Table 4: Improvements compared to the analog simulation in FOM_1 and FOM_2 for one method for some simulation times.

Flux method

Time	FOM_1	FOM_2
19046	$\times(2.10 \pm 0.10)$	$\times(1.58 \pm 0.01)$
76687	$\times(2.43 \pm 0.01)$	$\times(1.58 \pm 0.01)$
119643	$\times(3.98 \pm 0.01)$	$\times(1.79 \pm 0.03)$

pattern as in the large model. The trend is downward sloping, but this might change with a longer simulation time.

The qualitative results remain the same and the weights in this model are still updated after 150×10^3 s (Figure 33). Figures 32 and 33 show the average track weight for some typical cells in the geometry's outer regions. The figures show that some cells peak and then find a low weight window while some others peak and then trend upwards. The peak can be explained by the same reasoning as in the small model, the window is set too high at the first iteration. Some cells will then find an immediate stability while some are still updated when the simulation is finished. These cells are still updated when the simulation is finished. Both of these cells see no flux in the analog run and gets a weight that is higher than what would be the result if it had some flux already. This is then corrected in the following iterations. The cell in Figure 33 is still updated even after the simulation is finished.

Figure 34 is the cumulative histogram of relative errors for the large model and has the same shape as the smaller model (Figure 25) but is shifted to the right. Most cells have a relative error ≤ 0.4 using the weight window method but very few have an error ≤ 0.1. It is clear, however, that the spread in the errors are significantly lowered using this method even if the simulation time is short. Figure 35 shows the number of empty cells as a function of time and in this simulation the weight window method needs some iterations to fill all the cells.

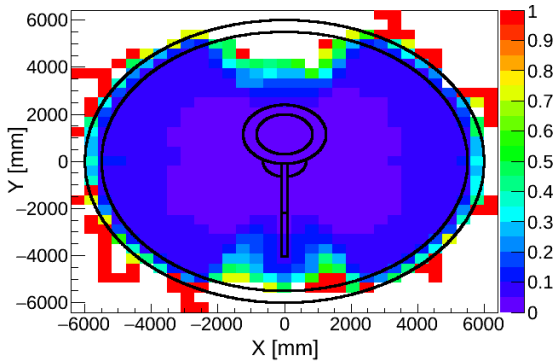

Figure 28: Relative error map of the 119 927 s long analog run.

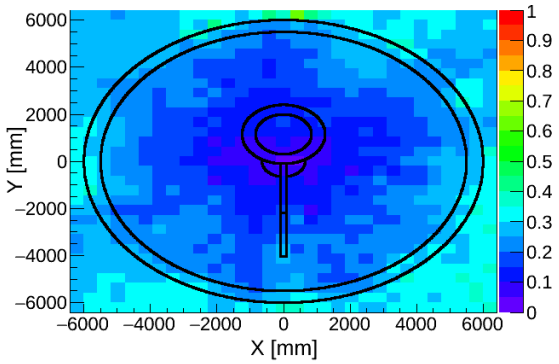

Figure 29: Relative error map of the 119 643 s long weight window run.

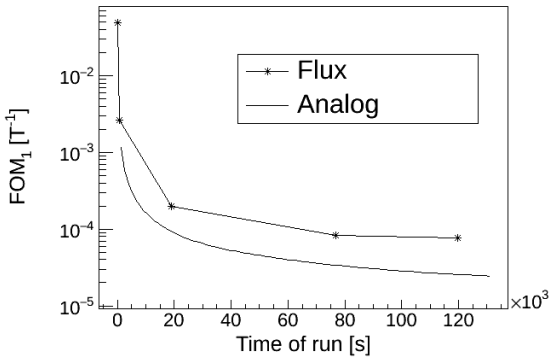

Figure 30: A comparison of the methods efficiency when applied to the large model using FOM_1.

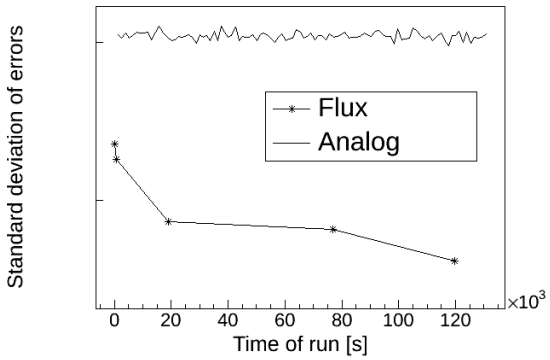

Figure 31: A comparison of the methods efficiency when applied to the large model using FOM_2.

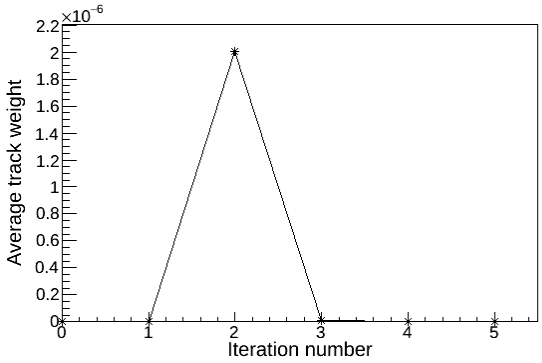

Figure 32: The average track weight of a typical cell in the large geometry's outer regions using flux based method. This cell is not significantly updated after iteration number 2.

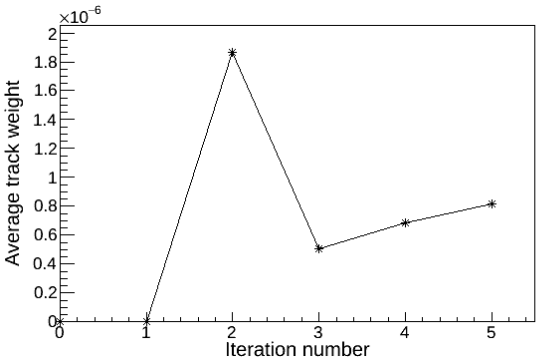

Figure 33: The average track weight of a typical cell in the large geometry's outer regions using flux based method. This cell shows a trend in the update even after some time.

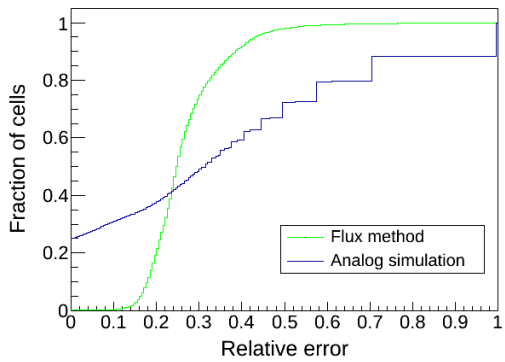

Figure 34: A cumulative histogram of the fraction of cells with a given relative error when applying the two methods, as well as the analog simulation, on the large model.

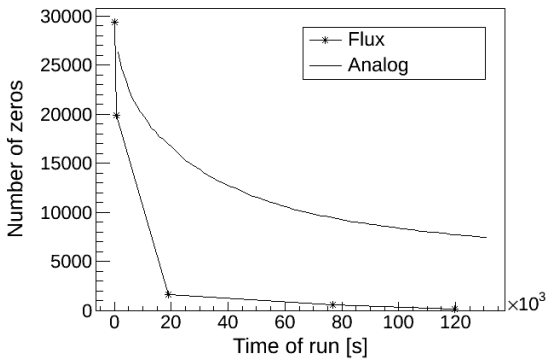

Figure 35: The number of cells that see no flux in the large target model using the two methods as well as in the analog simulation plotted against the simulation time.

In practice, a simulation can be run until a sufficient relative error is reached in the least sampled cell and then invoking a break condition. The update of the weight windows, if using the iterative approach, can also be stopped on a condition regarding average track weight. If this is stable for the whole grid there is no need to update the weight windows.

4 Conclusion

4.1 Conclusion

The methods discussed in this thesis both give an improvement in flux simulations of the ESS TDR target region. Less computation time has to be spent to get relevant statistics. For this thesis the simulations have been run for relatively short times on a regular laptop. To get a better simulation results, longer run times or more computer power is needed.

The flux based method dominates the relative error based method with regards to their respective figures of merit, hence this is the method primarily suggested for use.

Furthermore, the method is indeed easy to use and implement in principle. There is still some user judgment needed, but it is possible to implement this as a 'black-box' method taking only the grid and previously generated flux information as input. The results presented here are consistent with results presented in [2].

4.2 Outlook and further work

The choice of how to set cells with zero previous flux is based on what was possible within the time frame of the thesis work. This could be investigated further as there are some options. One considered possibility was to set weight in cells as a mean of neighboring cells. Another option is the one presented by Van Wijk [2]. The best choice for the cells with no flux is an uninvestigated topic. This is connected to the flux based method as there is a straightforward way to set the zero flux cells using the relative error method.

There is room for improvement and optimization. For example, the time it takes to set up the mesh grid and give all cells their weights is about 5 min for a $31 \times 31 \times 31$ on the used computer. This is a property of Geant4 since this time is recorded even when doing no calculations and using the importance sampling algorithm to set all importances to one.

The program was also tested on a multicore computer but does not yet produce proper results. With some corrections it could work in such an environment. Furthermore the method does not perform any energy dependent weighting. This would be a good addition to the procedure.

The method shows good improvement of flux calculations on the ESS target model. It produces correct spectra for neutron energies and is indeed easy to use and implement without much further consideration on any geometry. These methods should be able to find use in practical neutron flux simulations. With some further work this method or something like it could be included in the Geant4 framework.

References

[1] Geant4 Collaboration. *Geant4 User's Guide for Application Developers Version: 10.1*. Geant4 Collaboration, 2014.

[2] AJ Van Wijk, Gert Van den Eynde, and JE Hoogenboom. An easy to implement global variance reduction procedure for MCNP. *Annals of Nuclear Energy*, 38(11):2496–2503, 2011.

[3] Steve Peggs, R Kreier, C Carlile, R Miyamoto, A Pahlsson, M Trojer, and JG Weisend II. ESS technical design report. *European Spallation Source ESS AB*, 2013.

[4] Konstantin Batkov, Alan Takibayev, Luca Zanini, and Ferenc Mezei. Unperturbed moderator brightness in pulsed neutron sources. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 729:500–505, 2013.

[5] Günter Siegfried Bauer. Physics and technology of spallation neutron sources. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 463(3):505–543, 2001.

[6] R Arthur Forster, Lawrence J Cox, Richard F Barrett, Thomas E Booth, Judith F Briesmeister, Forrest B Brown, Jeffrey S Bull, Gregg C Geisler, John T Goorley, Russell D Mosteller, et al. MCNP™ version 5. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 213:82–86, 2004.

[7] FLUKA Collaboration et al. FLUKA manual. *ASCII or. pdf file available from FLUKA website and contained in FLUKA package*, 2011.

[8] Tatsuhiko Sato, Koji Niita, Norihiro Matsuda, Shintaro Hashimoto, Yosuke Iwamoto, Shusaku Noda, Tatsuhiko Ogawa, Hiroshi Iwase, Hiroshi Nakashima, Tokio Fukahori, et al. Particle and heavy ion transport code system, PHITS, version 2.52. *Journal of Nuclear Science and Technology*, 50(9):913–923, 2013.

[9] Nataliia Cherkashyna, Richard J Hall-Wilton, Douglas D DiJulio, Anton Khaplanov, Dorothea Pfeiffer, Julius Scherzinger, Carsten P Cooper-Jensen, Kevin G Fissum, Stuart Ansell, Erik B Iverson, et al. Overcoming high energy backgrounds at pulsed spallation sources. *arXiv preprint arXiv:1501.02364*, 2015.

[10] JE Beaver and HB Hupf. Production of 99mtc on a medical cyclotron: a feasibility study. *Journal of Nuclear Medicine*, 12(11):739–741, 1971.

[11] Richard Van Noorden. The medical testing crisis. *Nature*, 504(7479):202–204, 2013.

[12] Neal Jay Carron. *An introduction to the passage of energetic particles through matter*. CRC Press, Boca Raton, first edition, 2006.

[13] Alvin M Weinberg and Eugene Paul Wigner. The physical theory of neutron chain reactors. Chicago Univ. Press, Chicago, first edition, 1958.

[14] John R Lamarsh and Anthony John Baratta. *Introduction to nuclear engineering*. Prentice Hall, Upper Saddle River, third edition, 2001.

[15] JM Carpenter. Neutron production, moderation, and characterization of sources. *Argonne National Laboratory*, 2004.

[16] Kenneth S Krane. *Introductory nuclear physics*. John Wiley and Sons Inc., New York, second edition, 1987.

[17] Aatos Heikkinen, Nikita Stepanov, and Johannes Peter Wellisch. Bertini intra-nuclear cascade implementation in geant4. *arXiv preprint nucl-th/0306008*, 2003.

[18] Günter Siegfried Bauer. Neutron sources. *INIS collection*, 30(47), 1999.

[19] Norbert Ensslin. The origin of neutron radiation, chapter 11, passive nondestructive assay nuclear materials. Technical report, NUREG/CR-5550, 1991.

[20] JW Boldeman. Accelerator driven nuclear energy systems. *Energy For Ever: Technological Challenges of Sustainable Growth, Australian Academy of Technological Sciences and Engineering, Academy Symposium*, 1998.

[21] Bertram Terence Martin Willis and Colin J Carlile. *Experimental neutron scattering*. Oxford Univ. Press, Oxford, first edition, 2009.

[22] James J Duderstadt and Louis J Hamilton. *Nuclear reactor analysis*. John Wiley and Sons, Inc., New York, first edition, 1976.

[23] KH Beckurts and K Wirtz. Production and nuclear interaction of neutrons. In *Neutron Physics*, pages 1–21. Springer, 1964.

[24] Igor Koprivnikar and E Schachinger. The biological shield of a high-intensity spallation source: a monte carlo design study. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 487(3):571–584, 2002.

[25] National nuclear data center. `http://www.nndc.bnl.gov/sigma/`. Accessed: 2015-05-21.

[26] Roger Eckhardt. Stan Ulam, John von Neumann, and the monte carlo method. *Los Alamos Science*, 15:131–136, 1987.

[27] Ivan Lux and László Koblinger. *Monte Carlo particle transport methods: neutron and photon calculations*, volume 102. CRC press, Boca Ration, first edition, 1991.

[28] Douglas E Peplow. Comparison of hybrid methods for global variance reduction in shielding calculations. *Transactions of the American Nuclear Society*, 107:512, 2012.

[29] Monte Carlo Team. MCNP–a general Monte Carlo N-particle transport code, version 5. *Book MCNP-A General Monte Carlo N-Particle Transport Code Version*, 5, 2003.

[30] Andrew Davis and Andrew Turner. Comparison of global variance reduction techniques for Monte Carlo radiation transport simulations of ITER. *Fusion Engineering and Design*, 86(9):2698–2700, 2011.

[31] James Bucklew. *Introduction to rare event simulation*. Springer Science & Business Media, New York, first edition, 2004.

[32] Pierre L'Ecuyer and Bruno Tuffin. Splitting with weight windows to control the likelihood ratio in importance sampling. In *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 21. ACM, 2006.

[33] Marc A Cooper and Edward W Larsen. Automated weight windows for global monte carlo particle transport calculations. *Nuclear science and engineering*, 137(1):1–13, 2001.

[34] TL Becker and EW Larsen. The application of weight windows to "global" monte carlo problems. In *Proc. Conf. Advances in Mathematics, Computational Methods, and Reactor Physics*. American Nuclear Society Saratoga Springs, New York, 2009.

[35] Rene Brun and Fons Rademakers. ROOT—an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1):81–86, 1997.