A time-dependent Monte Carlo simulation for nuclear reactor dynamics using GPUs

Ph.D. thesis

Balázs Molnár
Supervisor: Dr. Dávid Légrády

Budapest
2020
Contents

1 Introduction 1

2 Principles of dynamic Monte Carlo on the GPU 13
  2.1 Handling direct time dependence in the presence of feedbacks 14
    2.1.1 Neutron tracking in time 14
    2.1.2 A branchless neutron-history method with combing 14
    2.1.3 Robust delayed neutron treatment with forced decay 18
    2.1.4 Initial conditions 21
  2.2 Monte Carlo methods for neutron physics modeling 22
    2.2.1 Representation of geometry and nuclear data 22
    2.2.2 Sampling distance to collision on the GPU 22
    2.2.3 Interaction physics modeling 23
  2.3 Validation and verification 25
    2.3.1 Differential comparison 25
    2.3.2 Integral comparison 28

3 A framework for the efficient sampling of distance to collision 31
  3.1 Introduction 32
  3.2 The Biased Woodcock method 32
  3.3 Mathematical proof of correctness 35
  3.4 Previous work and relation to the exponential transform 38
    3.4.1 Work of Spanier 38
    3.4.2 Weighted Delta Tracking (WDT) 39
    3.4.3 Works of Galtier et al., Carter et al., and Cramer 39
    3.4.4 Residual Ratio Tracking (RRT) 40
3.4.5 Woodcock tracking in medical physics and computer graphics ........................................ 40
3.4.6 Relation to the exponential transform ................................................................................. 40
3.5 Error estimation and computational cost ................................................................................. 42
3.6 Optimization strategies ........................................................................................................ 44
  3.6.1 Problem a) - a numerical optimization study ................................................................. 45
  3.6.2 Problem b) - BW and exponential transform in the straight-ahead scattering model .... 51
  3.6.3 Problem c) BW and exponential transform in the Fermi-scattering model .................... 52
3.7 Practical considerations and the majorant mesh ................................................................. 53

4 Variance reduction and acceleration methods ................................................................. 57
  4.1 Importance-based population control ..................................................................................... 59
    4.1.1 The importance weighted comb ....................................................................................... 59
    4.1.2 Importance definition and generation ............................................................................... 60
  4.2 The event-based GUARDYAN .............................................................................................. 61
    4.2.1 Previous work on event-based Monte Carlo ..................................................................... 62
    4.2.2 Thread Divergence ........................................................................................................... 63
  4.3 Accelerating point-in-cell search ............................................................................................ 64
  4.4 Performance assessment and analysis ..................................................................................... 64
    4.4.1 Variance analysis of time-dependent tallies ...................................................................... 65
    4.4.2 Performance analysis of combing in GUARDYAN .......................................................... 67
    4.4.3 Performance of the event–based GUARDYAN ................................................................. 71
    4.4.4 Evaluation of acceleration methods .................................................................................. 75

5 Transient analysis with GUARDYAN ..................................................................................... 79
  5.1 Benchmarking a kinetic Monte Carlo solver ......................................................................... 79
  5.2 Experimental setup ................................................................................................................ 80
  5.3 Benchmark model .................................................................................................................. 82
  5.4 Transient simulations ............................................................................................................. 83

6 Conclusion and future work .................................................................................................. 91

Acknowledgment ...................................................................................................................... 93
Chapter 1

Introduction

The Monte Carlo (MC) simulation of time-dependent neutron transport for transient analysis has been subject to intensive research lately due to advancements of High Performance Computing (HPC) platforms making such calculations feasible. Transient analysis requires the modeling of reactor dynamics (neutron transport in the presence of feedbacks), thus the term dynamic MC is often used to refer to MC neutron transport methods dedicated to transient analysis. Dynamic capabilities of well-known MC tools are under development within e.g. the McSAFE project [1]. This particular area of MC methods became a hot-topic since the introduction of the Dynamic Monte Carlo (DMC) method [2] providing essential concepts that make time-dependent MC viable for transient analysis. With the progress made on multi-physics applications (coupling the neutronics to thermal-hydraulic and thermo-mechanic feedback) now allowing for the simulation of whole reactor cores, developing a high-fidelity calculation is critical. MC methods can be such tools, on a practical level however, dynamic MC is still very much limited by its inherent computational burden. Large computing costs can be addressed via massive parallelism available in modern HPC, and dynamic MC can soon be available to individual users. A major challenge in this respect is represented by scaling these codes to future computing platforms, e.g. clusters of multicore nodes or heterogeneous environments containing Graphics Processing Units (GPUs), which gives the primary motivation to endeavor on writing this thesis.

To better understand the challenges of MC neutron transport calculations for reactor dynamics (dynamic MC), we start with the mathematical formulation of the problem. The neutron field in nuclear reactor cores can be described by the time-dependent Boltzmann equations:

\[
\frac{1}{v} \frac{\partial \Phi(\vec{r}, \vec{\Omega}, E, t)}{\partial t} = -\vec{\Omega} \cdot \vec{\nabla} \Phi(\vec{r}, \vec{\Omega}, E, t) - \Sigma_t(\vec{r}, E, t) \Phi(\vec{r}, \vec{\Omega}, E, t) + Q(\vec{r}, \vec{\Omega}, E, t) + \sum_{i=1}^{I} \chi_i(E) \lambda_i C_i(\vec{r}, t) \tag{1.0.1}
\]

\[
\frac{\partial C_i(\vec{r}, t)}{\partial t} = -\lambda_i C_i(\vec{r}, t) + Q_i(\vec{r}, \vec{\Omega}, E, t) \quad \text{for} \quad i = 1, 2, ..., I \tag{1.0.2}
\]
where \( Q \) and \( Q_i \) incorporates the source of neutrons and delayed neutron precursors, respectively:

\[
Q(\vec{r}, \vec{\Omega}, E, t) = \int_{4\pi} \int_{0}^{\infty} \Sigma_s(\vec{r}, \vec{\Omega}' \rightarrow \vec{\Omega}, E' \rightarrow E, t) \Phi(\vec{r}, \vec{\Omega}', E', t) + \\
+ \chi_p(\vec{r}, E, t)(1 - \beta(\vec{r}, E', t))\nu(\vec{r}, E', t)\Sigma_f(\vec{r}, E', t)\Phi(\vec{r}, \vec{\Omega}', E', t)dE'd\vec{\Omega}' + S(\vec{r}, \vec{\Omega}, E, t) \tag{1.0.3}
\]

\[
Q_i(\vec{r}, \vec{\Omega}, E, t) = \int_{4\pi} \int_{0}^{\infty} \beta_i(\vec{r}, E', t)\nu(\vec{r}, E', t)\Sigma_f(\vec{r}, E', t)\Phi(\vec{r}, \vec{\Omega}', E', t)dE'd\vec{\Omega}' \tag{1.0.4}
\]

and we used notations of Table 1.1. The corresponding initial conditions can be written as \( \Phi(\vec{r}, \vec{\Omega}, E, 0) = \Phi_0(\vec{r}, \vec{\Omega}, E) \) and \( C_i(\vec{r}, 0) = C_{i,0}(\vec{r}) \) for \( i = 1, 2, ..., I \).

- \( v \) - neutron speed
- \( \vec{r} \) - position
- \( \vec{\Omega} \) - direction of movement
- \( E \) - energy
- \( t \) - time
- \( \Phi \) - neutron flux
- \( \Sigma_t \) - total macroscopic cross section
- \( \Sigma_s \) - scattering cross section
- \( \Sigma_f \) - fission cross section
- \( \beta \) - delayed neutron yield
- \( \beta_i \) - delayed neutron yield of family \( i \)
- \( \nu \) - total neutron yield at fission
- \( \chi_p \) - prompt fission neutron spectrum
- \( \chi_i \) - delayed fission neutron spectrum of family \( i \)
- \( \lambda_i \) - decay constant of family \( i \)
- \( C_i \) - precursor concentration of family \( i \)
- \( I \) - number of precursor families
- \( S \) - external source

Table 1.1: Notations used in this thesis

The mathematical model of reactor dynamics is completed by formulas describing feedbacks on the cross sections in Eq. 1.0.1 and Eq. 1.0.2. For reactor transients in the order of seconds, the relevant feedbacks are due to temperature changes via neutron heating. This set of equations, where the neutron field is closely
coupled with thermal-hydraulic and thermo-mechanic phenomena, is solved by coupling independent, already validated tools. In this respect, dynamic MC stands for a neutronic MC tool that solves for Eq. 1.0.1 and Eq. 1.0.2, while also prepared for such coupling and updating cross sections from time to time. The MC solution of Eq. 1.0.1 and Eq. 1.0.2 relies on the integral formulation of the Boltzmann equations. It can be shown [3], that the above integro-differential form can be written as a Fredholm-type equation of the second kind:

\[ \Phi(P) = q(P) + \int k(P, P')\Phi(P')dP' \]  

(1.0.5)

where \( P \) is a shorthand notation for phase space \((\vec{r}, \vec{\Omega}, E, t)\), \( q \) stands for a source of neutrons and \( k \) is the so-called transport kernel. This kind of equation can be solved by MC via Neumann series expansion [4]. The idea is to write the neutron flux as a sum of the contributions to flux by non-collided particles, once-collided particles, twice-collided etc.:

\[ \Phi(P) = \sum_{j=0}^{\infty} \Phi_j(P) \]  

(1.0.6)

with \( \Phi_0(P) = q(P) \) and \( \Phi_j(P) = \int k(P, P')\Phi_{j-1}(P')dP' \). Then, any unknown quantity that can be written in the form \( \int f(P)\Phi(P)dP \) can be evaluated by constructing a random walk in the phase space (particles traveling between collisions with energy and direction changing only at collisions), and summing up contributions \( f(P) \) from non-collided particles, once-collided particles, twice-collided particles and so on. The random walk is generated by sampling probability density functions (pdfs) describing random events in a particle track, such as the particle birth (source term), the distance traveled between collisions (transition kernel) and the energy and direction of the particle(s) emitted from collisions (collision kernel). The transition kernel can be written as

\[ T(s) = \Sigma_i(\vec{r'})exp\left(-\int_0^s \Sigma_i(\vec{r} + \vec{\Omega}t)dt\right) \]  

(1.0.7)

assuming that the particle travels from \( \vec{r} \) to \( \vec{r'} \), and the collision kernel is

\[ C(P, P') = \frac{\sum_i(\nu\Sigma_i)\chi(P, P')}{\Sigma_i(P)} \]  

(1.0.8)

where \( i \) is the reaction type, \( \chi \) is the distribution of post-collision coordinate \( P' \) (and can be dependent on pre-collision coordinate \( P \)), and \( \nu \) is defined the following way: \( \nu = 0 \) for capture reaction, \( \nu = 1 \) for non-multiplying reactions, \( \nu = x \) for \((n, xn)\) reactions, and \( \nu = \nu_{tot} \) for fission reactions. Due to the stochastic nature of the solution, any estimate will have some statistical uncertainty around its mean value. Convergence of MC follows the \( \frac{1}{\sqrt{N}} \) rule, which means that this uncertainty (standard deviation) can be reduced by a factor of \( \sqrt{2} \) by simulating twice as many random walks. Some methods can improve the accuracy of the estimation without increasing the number of starters, these are called variance reduction.
methods and have an important role in MC. In particle transport, such algorithms are often implemented by introducing the statistical weight of a particle so that it represents a bunch or a fraction of particles. Instead of the original pdfs, other distributions are sampled that are more convenient to sample in some way or yield more accurate estimates and this bias is compensated by altering the statistical weight of the particle, keeping the unbiasedness of an estimate. These simulations are called non-analog MC simulations, since the random walk constructed is not analog to the true physical process. An important measure of MC solutions using variance reduction techniques is the Figure of Merit (FoM), which is defined as

$$FoM = \frac{1}{\sigma_r^2 t}.$$  \hspace{1cm} (1.0.9)

where $t$ denotes the simulation time and $\sigma_r$ can stands for the relative error of the estimation. FoM measure the efficiency of MC simulations and since usually $t$ is proportional to $N$ and $\sigma_r$ follows the $\frac{1}{\sqrt{N}}$ rule, FoM is independent of the number of starters. $\sigma_r$ is often calculated from the sample variance:

$$S^2 = \frac{1}{N-1} (x_i - \bar{x})^2.$$ \hspace{1cm} (1.0.10)

where $\bar{x}$ is the sample mean and $x_i$ is the contribution from a single starter. As a starter may contribute to an estimate many times during its history, it is important to note that $x_i$ is a sum of all contributions including contributions from progenies. Thus $x_i$-s are non-correlated quantities. Correlation between histories can however be introduced deliberately or as an unwanted byproduct of some phenomena either physical or non-physical (computer algorithm). In dynamic MC for example, physical feedback is a certain source of correlations since neutron histories affect each other through the feedback.

Beside MC, other approaches of neutronics modeling are the deterministic and hybrid (deterministic + MC) methods. Up to the 2010’s, neutronics during reactor transients has been solely the area of such methods, a large hiatus being that these calculations are difficult to validate lacking a high-fidelity reference calculation. Dynamic MC can remedy this as it does not suffer from any error resulting from the discretization of phase space or assumptions regarding neutron transport, e.g. diffusion theory. Computation time is however significantly higher for dynamic MC, therefore it may not eliminate deterministic methods for some time, but can provide a benchmark for faster tools. As an example, a state-of-the art hybrid method based on the improved quasi static method, solve Eq. 1.0.1 and Eq. 1.0.2 via employing factorization of $\Phi(\vec{r},\vec{0},E,t)$ into a shape function that varies slowly with time, and another factor that incorporates fast changes. The shape function is calculated on a lattice scale by a static MC calculation, and the time evolution is then approximated on a coarser grid by some deterministic (e.g. diffusion) solver. Handling time dependence becomes thus significantly less expensive, than including it directly into a MC calculation. On the other hand, fidelity of the method is limited by the factorization, as it is not capable to fully capture changes.
during strong transients or localized phenomena, when the flux profile changes quickly. In the modeling of reactor transients involving large and/or local reactivity changes, dynamic MC can prove to be the only option.

To include time dependence directly in the MC calculation and thus absolve the solution from approximation and discretization problems has only been recently considered. Up to the 2010’s, even the feasibility of such calculations [5] has been called into question. Insufficient computational power has not been the only limitation, but a proper methodology to treat problems imposed by time dependence was absent. Key issues that need to be overcome by dynamic MC can be summarized in the following points:

- In Eq. 1.0.1 and Eq. 1.0.2 system parameters are not constant in time, i.e. cross sections are also time-dependent. Time dependence of cross sections can be due to geometry alterations, e.g. rod movements during transients, or to thermal feedback. Feedback mechanisms represent a more serious challenge for dynamic MC, since while geometry alterations could be implemented with continuous parametrization, feedback can only be calculated with prior knowledge of the neutron field. This problem can however be circumvented by the fact that thermal feedback mechanisms take much longer to come into effect than neutronic events. Thus an opportunity presents itself to evolve the neutron population time step by time step, assuming cross sections change only at certain time boundaries due to feedback. The downside is that the standard way of processing neutron histories independently should be, to this end, discarded, neutron histories need to be synchronized in time to allow feedbacks to act upon cross sections at the end of time steps. Further consequence is that time will be explicitly present in the simulation and MC tallies can not be averaged over whole neutron histories, but must be gathered into time bins.

- Eq. 1.0.1 and Eq. 1.0.2 contains the density of delayed neutron precursor atoms $C_i(\vec{r}, t)$. Typically, precursors are not sampled in criticality MC calculations, but delayed neutrons are produced instantaneously. Dynamic MC must however account for the time delay. A straightforward way to go would be just to treat delayed samples the same way as prompts, inserting them into the simulation at some later time by sampling an exponential decay, with energy that is characteristic for the delayed neutrons. However, the serious difference between prompt and delayed timescale would undermine this analog strategy in two ways. First, computer memory would be occupied unnecessarily by delayed samples being idle for many prompt generations. Second, stability of the simulation would be compromised as prompt fission chains can vanish in the absence of delayed neutrons. Considering that the typical length of prompt fission chains in thermal systems are in the order of 10ms, and that delayed neutrons may not be produced for even seconds after a fission event, prompt fission chains must be abundant
to provide MC tallies evenly in time and do not go extinct at any instant during the transient. In real reactors, the neutron population exceeds the numbers which a computer simulation can accommodate by many orders of magnitude, thus the abundance of fission chains only holds for real cases and not MC simulations. A simple solution to this is to sample delayed neutron precursors along with prompt neutrons. Precursor sampling can not be carried out in an analog way however, since the number of precursors greatly exceeds the number of neutrons, for one neutron there can be in fact thousands of precursors.

- In Eq. 1.0.1 the time derivative of $\Phi(\vec{r},\vec{\Omega},E,t)$ is proportional to $\Phi(\vec{r},\vec{\Omega},E,t)$ itself, yielding an exponential change of flux over time. The number of neutron samples during transients may thus grow unbounded (supercritical) or become low (subcritical) rapidly. The neutron population must therefore be controlled automatically by dynamic MC.

- To solve Eq. 1.0.1 and Eq. 1.0.2 by MC, initial conditions are needed in the form of an initial neutron sample distribution and an initial precursor sample distribution. One could expect these distributions as an input given by the user, but it is also sensible to assume that the simulation starts from an equilibrium state since the future would otherwise depend on the previous history of the reactor via the precursor atoms.

Addressing the above issues has been the main target in the development of dynamic modules for some production-level codes, but most are still subject to strong approximations. Delayed neutron treatment is still simplified in an extension of the OpenMC code [6], while the G4-STORK code [7] built on the Geant4 software [8] ignores delayed neutrons or produces them instantaneously. In other studies the time shift of delayed neutrons has been taken into account, but either only 2D cases [9] or diffusion approximations [10] are considered to reduce the computational burden. A different approach is described in a research [11], where the difference between time scales is addressed by solving the frequency domain transport equation by MC. The fundamental concepts addressing all of the above points were laid down by Legrady and Hoogenboom [5] and were first implemented in the DMC method [2]. Other successful implementations seem to follow this methodology, e.g. the dynamic Serpent 2 (dynSerpent) [12] [13] or dynamic TRIPOLI-4 (dynTripoli) [14], and, as such, are also taken into account in this thesis. This methodology includes the following ideas:

- Time-stepping: The simulation is split into time intervals. During time steps, cross sections are assumed to be unaffected by thermal feedback. Fission power tallies are collected and summed over a time step to allow thermal feedback to be calculated and admit cross section changes at the end of the step.
• Forced decay of delayed neutron precursors: Precursor samples represent a population of precursor atoms via precursor statistical weight, thus the ratio of precursor samples to neutron samples do not have to reflect the true, analog conditions. The decay of a precursor sample is forced to happen in each time step while part of the precursor sample is also forced to survive. In practice, a precursor sample is split into a decaying and a surviving part yielding a delayed neutron with weight \( (1 - e^{-\lambda \Delta t}) \) times the precursor weight, while a fraction of \( e^{-\lambda \Delta t} \) of its weight is kept as a precursor sample.

• Improved branchless method: Fission and capture reactions are treated implicitly. History branching is not allowed at fission (branchless method), i.e. instead of creating multiple progenies, only one fission neutron is sampled and the bias is compensated by increasing the statistical weight of the sample. Survival biasing, also known as implicit capture, is applied at collisions preventing sample loss, i.e. instead of terminating the neutron history, capture is never sampled, but the neutron goes on with a weight reduced by the probability of survival. Since the branchless method and implicit capture can result in large fluctuations in the weights of the sample population, the biasing of fission probability is also used (forced fission). In forced fission, the probability of fission is increased so that neutrons can not take on large weight increments when in fissile material, and can not lose weight quickly while in non-multiplying media. The above techniques yield the improved branchless method, that uses the following nonanalog probabilities for fission, scattering and capture:

\[
P_f = \frac{(1 - \beta) \nu \Sigma_f}{(1 - \beta) \nu \Sigma_f + \Sigma_s} \quad \text{(1.0.11)}
\]

\[
P_s = \frac{\Sigma_s}{(1 - \beta) \nu \Sigma_f + \Sigma_s} \quad \text{(1.0.12)}
\]

\[
P_c = 0. \quad \text{(1.0.13)}
\]

This particular choice of probabilities ensure that the neutrons undergo the same weight change at each collision:

\[
w' = w \frac{(1 - \beta) \nu \Sigma_f + \Sigma_s}{\Sigma_f} \quad \text{(1.0.14)}
\]

• Generation of initial conditions: Equilibrium initial conditions are generated by a static MC calculation. A k-iteration is used to converge on the fission source, but instead of the fission source, the neutron and precursor source is tallied which gives an initial distribution of particles to launch a dynamic calculation.

Despite these examples of dynamic MC implementations, practical applicability is still an issue. In the case of dynamic TRIPOLI, authors report runtimes of 800 CPU hours for the simulation of 1 second in real
time without thermal feedback [14], the same with thermal feedback is reported as 1200 CPU hours. With
dynamic Serpent, runtime figures were found in the same order of magnitude (800 CPU hours) by a recent
paper [15], indicating that such calculations are still very slow.

To speed up the simulation application of GPUs represent an alternative to clusters of CPUs, however,
transition between platforms is not trivial, and may even be unreasonable [16] since the entire code would
have to be restructured to approach similar performance due to the fundamental differences between GPU
and CPU architectures. In other words, simply transporting a CPU code to run on GPUs would not
meet the expectations of having any performance boost, and the GPU version would most likely be less
efficient. Also, decades of careful optimization of CPU codes should not be overlooked in this respect. Thus,
starting from scratch, an implementation of dynamic MC on the GPU was investigated, and is presented
in this thesis. The resulting code, GUARDYAN (GpU Assisted Reactor DYnamic ANalysis) [17], boasts of
performance approaching that of state-of-the-art MC methods via overcoming hardware constraints at the
cost of abandoning some widely used MC techniques and introducing overall a unique methodology.

Capabilities of GPUs have grown exponentially in the last years, a single commercially available GPU
possesses performance of $10^9$ floating point operations per second (TFLOPS), and the difference between
computing power of GPUs and CPUs tends to be even greater. In scientific computations the application
of GPUs proved to be successful many times in the past. Examples from the MC neutron transport field
include the WARP continuous energy MC code [18], the ALPS MC test code at LLNL (Lawrence Livermore
National Laboratory) [19] and the ARCHER MC code [20]. A comparison of CPU and GPU implementation
of a criticality calculation showed 7 times speedup in favor of the GPU [21]. Time-dependent MC has not
been implemented on the GPU yet, thus there are no standard ways available as reference. The main subject
of this thesis, GUARDYAN, is a pioneering work in this field.

Success of the code lies in the methodology developed to exploit advantages of GPUs and avoid such
constructions that would degrade parallel efficiency. To ensure peak parallel performance on the GPU,
memory management and the thread divergence issue must be understood. Parallelization is performed by
the independent working units of the GPU called threads. A GPU function called a kernel is executed by
groups of threads called thread blocks, and these are further organized on a grid, as seen in Fig. 1.2. This is
also important with respect to that it ensures automatic scalability of the program, as blocks of threads can
be scheduled on any multiprocessors of the device, yielding faster execution time when more multiprocessors
are available.

In contrast to CPUs, threads can access on-chip memory of a limited size (also fixed for each block),
but are more numerous, shown in Fig. 1.3. The number of concurrent threads is bounded by their memory
need, thus using more memory by a kernel essentially results in less parallelization. E.g. peak parallelization
is achieved by using no more than 32 registers, overall 32 single-precision or 16 double-precision variables, on the current GPU card available for running GUARDYAN (Nvidia GTX 1080). The challenge here lies in the implementation of kernel functions, so that kernels do not use many local variables. Typical kernel functions executing neutron transport in GUARDYAN use 60 to 100 registers in the current version, thus some parallelization is lost due to this limitation. Note however, that the level of parallelization (the number of concurrent threads) is not the only factor affecting overall execution time, there is also memory latency and work-load balance to consider. Simultaneously running threads are organized in warps on the GPU operating as a vector processor, i.e. they execute the same instruction on multiple data. Results are obtained only when the slowest thread finishes. Therefore it is paramount to distribute tasks evenly, otherwise performance will be lost due to thread divergence [22]. Implicit loops, conditional branching have to be avoided in GPU
codes. In MC neutron transport this means that branching of neutron histories at fission, splitting/Russian roulette, and any other processes requiring various amount of resources for different threads are strictly forbidden, or should be minimized.

Another issue relating to memory management is the allocation of memory for geometry and nuclear data. On the GPU one has several options to allocate memory for a variable, one can use register, local, shared, texture, constant and global memory, shown in Fig. 1.4. Memory transactions between GPU (device) and CPU (host) are carried out through reading and writing global memory. As the access of global memory is slow, these transactions can also take a considerable time, and can have a significant impact on overall performance. Registers ensure the fastest memory access and are assigned to each thread. Global, constant and texture memory can be accessed by all threads, while the scope of shared memory is only a block. In exchange, it is much faster. Texture memory is not truly a distinct memory type, it only labels a part of global memory that is bound to texture. Textures are implemented with hardware interpolation, thus they would be ideal for storing cross section data. But due to random memory access patterns inherent in MC simulations, using cached memory is not advised [18]. Due to the size of nuclear data, one can only consider global memory of the GPU. E.g. geometry and nuclear data needed for the MC model of the Training Reactor at Budapest University of Technology and Economics takes up 201MB of memory for one temperature and assuming fresh fuel composition. Larger scale problems like VVER-440 cores assuming fuel burnup occupy about 2GB, that is considerable on the current card used by GUARDYAN with global capacity of 8GB.

Currently GUARDYAN runs on a machine containing two Nvidia GeForce GTX 1080 cards, each with 8 GBytes of global memory and 5500 GFLOP/s single precision performance according to NBody GPU benchmark. The GTX 1080 cards are based on the Pascal architecture and have 2560 scalar working units (CUDA cores). These cores can launch warps of 32 concurrent threads, resulting in a theoretical maximum of 81920 parallel working units.
Figure 1.4: CUDA memory types
Chapter 2

Principles of dynamic Monte Carlo on the GPU

Undertaking the MC simulation for reactor dynamics is highly challenging since it calls for concepts completely absent in standard approaches of MC neutron transport. Although some efforts were already put into deriving proper algorithms that make such simulations feasible, practical application is still limited by the time required to obtain accurate results. Graphics Processing Units (GPUs) can be used to accelerate calculations but again, a novel methodology is needed, since a conventional MC implementation is destined to fail due to the structural differences between CPU and GPU.

This chapter introduces GUARDYAN as a proof-of-concept implementation of dynamic MC on the GPU. We start with the methodology required by the dynamic nature of the problem. In this first section we only summarize techniques that are specific for the dynamic problem and assume that the reader is familiar with standard MC neutron transport methods. These classical MC concepts implemented in GUARDYAN, e.g. the representation of continuous-energy cross sections, geometry, neutron physics modeling are described in Section 2.2, and in more detail in the Appendix. The third section elaborates on some validation and verification efforts, targeting mainly the correct implementation of neutron physics and also some of the correct coding of time-dependence, but the benchmarking of full kinetic capabilities is found in Chapter 5. Up to the writing of this thesis, the coupling of GUARDYAN to a thermal-hydraulic solver was not yet implemented, thus truly dynamic tests will not be presented in this piece.
2.1 Handling direct time dependence in the presence of feedbacks

Starting from the basic concepts of Ref. [5] and Ref. [2] briefly described in the Chapter 1, we have developed the critical algorithms for dynamic MC on the GPU. Compared to the above references novel contributions of this section include the utilization of the time-stepping methodology for population control, a truly branchless neutron tracking method, a robust sampling of delayed neutrons and the generation of initial conditions, also found in papers [23] and [24].

2.1.1 Neutron tracking in time

In transient scenarios, perturbations to cross section data, material composition, or geometry must be evidently accounted for in order to model thermal-hydraulic (TH) feedback or geometry changes. While the latter could be calculated on-the-fly, considering TH feedback is only possible if neutron histories are synchronized in time, and so the conventional way of processing neutron histories independently has to be discarded. Strictly speaking, neutrons affect each other at every collision since TH feedback alters cross sections, but in practice, it is sufficient to admit feedback effects only between certain time steps. This way, one can still simulate neutron histories independently during a time step, but further processing must wait for the feedback. After the power distribution in a time step is scored over all histories, a TH calculation is run, and cross sections are then updated for the next time step according to the feedback effect.

It is reasonable to choose time steps shorter than the timescale of dynamic changes (rod movements, thermal feedback), so that system properties can be considered constant during a step. It is important to note however, that synchronizing neutron histories more often is of course not prohibited, and in GUARDYAN, this is exactly what is exploited. One can also consider this as a choice of shorter time steps, but not necessarily allowing feedback after each step. Typical length of time steps in GUARDYAN is in the range of a few generation times, e.g. $10^{-6} - 10^{-4}$s in thermal systems, while admitting feedback effects only occur with the rate of e.g. tenth of a seconds, so after every 1000-100000-th time step. This is a major difference in contrast with existing dynamic MC codes, and is motivated by the GPU implementation. In the next section, we show that a major opportunity for GUARDYAN to control the exponentially diverging neutron population presents itself at time boundaries, and if time steps are short, population control can be achieved without employing splitting and Russian roulette on-the-fly.

2.1.2 A branchless neutron-history method with combing

While the reactor is in a sub- or supercritical state the neutron flux decreases/increases exponentially, thus the number of neutron samples in a MC simulation can quickly become very low and yield poor statistics,
or high and exceed computer resources. From the GPU perspective, it is also essential to limit the change in the number of neutrons as not to undermine parallelization. Branching (e.g. allowing neutrons to produce progenies) or terminating histories are disastrous for GPU work-load balance. If a neutron would be allowed to create secondaries stored in a bank to be processed later, serious thread divergence would occur since there would be threads responsible for only one neutron, while others would have to deal with many more. Keeping a neutron bank on the GPU would also degrade performance due to the limited size of register memory per thread. Register memory assigned to one GPU thread would quickly overflow and the GPU would be forced to allocate global memory with considerably slower access speed. For these reasons, such a solution is needed that remedies the exponentially changing sample population issue, but in the meantime, also preserves parallel performance on the GPU. The basic approach to the problem is to use non-analog MC techniques by introducing the statistical weight of neutrons. A neutron sample no longer represents one neutron, but rather a population of neutrons or a fraction of one. Neutron multiplication and absorption can thus be treated via weight change. Capture can be simulated by multiplying the weight by the probability of survival:

\[
w' = w \left(1 - \frac{\Sigma_c}{\Sigma_t} \right)
\]  
(2.1.1)

where \(\Sigma_c\) is the capture cross section. Similarly, fission can be simulated by multiplying the statistical weight by the fission neutron yield \(\nu\). Note that with these methods, we essentially traded the fluctuation of neutron samples for fluctuating particle weights, solving the computer resource and thread divergence issue (at least as history branching is concerned), but further efforts must now be made in order to control diverging weights. The improved branchless method [2] was chosen and implemented in GUARDYAN for this reason. The technique, while preventing history branching, also forces fission events to occur more frequently with the idea that neutron weights will then take on smaller increments instead of the multiplication by \(\nu\). Overall, a constant population of samples is thus achieved in GUARDYAN, treating neutron multiplication (fission and \((n,xn)\) reactions) and capture events by changing the statistical weight of a neutron. Saving some memory space on the GPU, we store cross section data in a \(\nu\Sigma\) format, redefining \(\nu\) the following way: \(\nu = 0\) for capture reaction, \(\nu = 1\) for non-multiplying reactions, \(\nu = x\) for \((n,xn)\) reactions, and \(\nu = \nu_{\text{tot}}\) for fission reactions. GUARDYAN samples reaction of type \(i\) with probability of

\[
P_i = \frac{(\nu \Sigma)_i}{\sum_j (\nu \Sigma)_j}
\]  
(2.1.2)

and the weight change at each collision is

\[
w' = w \frac{\sum_j (\nu \Sigma)_i}{\Sigma_t}.
\]  
(2.1.3)

This collision sampling procedure is a slightly modified and generalized version of the improved branchless
method which has been proved to be unbiased in [2].

While the above algorithm ensures that in any collision the weight of a neutron changes the same way independently of the reaction type, the time evolution of weights of different neutron histories can still be very diverse. The improved branchless method does not solve the issue of diverging weights, but was nevertheless found to be mandatory to use for the calculation to converge. This is demonstrated in Fig. 2.1, where we show a serious underestimation of total power by the analog method in a very simple homogeneous system.

Figure 2.1: MC estimation of power evolution with analog fission vs. forced fission. The example model was a subcritical homogeneous cuboid with two energy groups. Combing was used for population control in both cases, but the analog simulation used the analog fission probability, while forced fission used Eqs. 2.1.2 and 2.1.3. Time step size was $10^{-4}$ s.

To contain particle weights within reasonable bounds splitting and Russian roulette is commonly used in MC particle transport. The usual implementation based on the weight window technique involves monitoring particle weights at each collision and splitting trajectories if weights exceed a threshold and playing a roulette if weights drop below a lower bound. Such population control is not very favorable for dynamic calculations however. First, note that the weight window technique fundamentally accomplishes the exact opposite of what is achieved by the branchless method and survival biasing, i.e. it transfers weight fluctuations back to fluctuations in the number of samples. The situation would be much different if splitting and roulette would only be applied at time boundaries, where particles are synchronized, thus thread divergence and
register memory problems would not occur. Another issue in dynamic MC would be to determine where weight windows should jump in the next time step during a strong transient. A more favorable method called the combing method [25] was taken into consideration for implementation in GUARDYAN. Unlike the splitting-Russian roulette technique, the combing method does not change the population size or require accurate knowledge of the desired particle weight at a certain time step. It handles a batch of particles together, and enables the use of population characteristics to set the surviving average weight following the power evolution of the system. Applied to a population of $N$ particles the method redistributes their total particle weight to a set of $M$ particles randomly chosen from the original stack. A simple comb disregarding the importance of particles combs the population of $N$ particles by randomly choosing $M$ copies (one may be represented multiple times in the new array) each assigned with a new weight of

$$w' = \frac{1}{M} \sum_{i=1}^{N} w_i$$  \hspace{1cm} (2.1.4)

A particle is chosen if a tooth of the comb hits the particle in an array of weights put side-by-side, the teeth positions being:

$$t_m = \rho w' + (m - 1)w'$$  \hspace{1cm} (2.1.5)

where $\rho$ is a random number from $(0, 1)$ and $m = 1...M$. This selection procedure is seen in Fig. 2.2. As a consequence, each particle is selected with probability proportional to the weight of the particle. Combing is used at time boundaries, thus particles are synchronized in time during combing. The main advantage of this method is that the power evolution of the system is therefore easily followed by the average particle weight; while on the other hand, on-the-fly application of the weight-window technique would require a desired particle weight that would be difficult to guess, as it would change with time.

![Figure 2.2: Illustration of a simple comb. A comb with equidistant teeth positions is shifted with a random offset $\rho w'$ resulting in the termination of particle 1, but making two copies of particle 2, and keeping particles N-1 and N.](image)
2.1.3 Robust delayed neutron treatment with forced decay

While prompt neutrons have an average lifetime in the order of $10\mu s$ (for thermal systems), the expected lifetime of a delayed neutron precursors can range from 0.1s to 100s. The difference is several orders of magnitudes, thus analog simulation of delayed neutrons is clearly unfeasible in a dynamic simulation. In practice this problem would arise in two ways: first, prompt fission chains would disappear due to the heavy undersampling of delayed neutrons. Second, a delayed neutron produced at a certain time would only participate in the simulation after many generations of prompt neutrons, taking up valuable computational resources in the meantime. Papers on time–dependent MC suggest to sample delayed neutron precursors instead [2] [5]. Key question is how to distribute samples between neutrons and precursors. Analog sampling would dictate to almost completely suppress neutrons in favor of precursors, as in an equilibrium state for instance, the equation defining the ratio of precursor atoms to live neutrons reads as

$$C_i(\vec{r}) = \frac{\beta_i}{\lambda_i\Lambda} n(\vec{r})$$  \hspace{1cm} (2.1.6)

where $C_i$ stands for the concentration of precursors from the $i$–th family, $\beta_i$ and $\lambda_i$ are the delayed neutron fraction and decay constant respectively, $\Lambda$ is the mean generation time, and $n$ denotes the density of live neutrons. With $\Lambda$ in the range of $10−100\mu s$, the number of precursor samples per one live neutron should be around $1000 − 10000$ if reality is faithfully modeled. Obviously, the importance of precursors with respect to the power evolution of the system would be thus highly overestimated. Therefore GUARDYAN treats precursors in a nonanalog way with a MC precursor sample representing a population of precursor atoms by introducing precursor statistical weight. The optimal ratio of precursor to neutron samples is not trivial to guess, as it changes with the time evolution of the system. In a subcritical system, precursors generated in the past play a more important role as prompt neutron chains would die out quickly in the absence of precursors, while in a supercritical case delayed neutron samples would not be that important. Conclusively, in order to optimally distribute MC samples, transient simulation would require a time–dependent importance of particles, in other words it would need to foresee transient behavior. To bypass this in GUARDYAN, we have developed a scheme that allows robust delayed neutron treatment without taking time–dependent importance into account.

Delayed neutron treatment in GUARDYAN was constructed along the lines of promoting GPU utilization and stability awareness. Neutrons and precursors are stored in GPU global memory, comprising a particle array. Changing the size of the array is prohibited at all times in order to ensure peak GPU thread occupancy. To allow dynamic changes in the population of live neutron samples, part of the array is left blank acting as a buffer. Delayed neutron samples are first forced to fill this blank space via forced decay [5]. Forced
decay yields delayed samples from all precursors while also preserving the precursor samples themselves. Delayed samples are then combed to the average weight of prompt neutrons using the simple comb [25], that is used as a population control technique on neutrons. In a supercritical state, where prompt neutron samples are numerous and can produce stable power evolution, this combing results in less delayed samples, since the average weight of prompts is high. In a subcritical state, the average weight of prompt neutrons decreases, and the combing yields many delayed neutron samples, ensuring that neutron fission chains are not undersampled in the upcoming time step. Meanwhile precursor samples are also conserved by the application of forced decay.

Supporting a better understanding of delayed neutron treatment and particle array operations, Fig. 2.3 is presented. Simulation starts with the initial distribution of particles given by the user or, as an alternative, critical initial conditions can be constructed (cf. Section 2.1.4). The GPU particle array is divided into two equal parts. One half is allocated for prompt neutrons, the other is again equally split between precursors and delayed neutrons for the upcoming time step. At the beginning of a time step, there are no delayed neutrons, the allocated memory space is at first empty. This can be seen as the first column in Fig. 2.3. Each time step ends with redistributing particles to this structure. We distinguish neutrons participating in the next time step (>n<) from neutrons that pass the next time step (denoted by >n′<), that is determined by the distance to collision sampling routine. Depending on the speed of the neutron and the sampled path length, it may be that the neutron has no interaction in the following time step(s). This neutron is essentially moved to stack >n′< and only put back to stack >n<, when the time of the interaction lies in the upcoming time step. Precursors are denoted by >C<.
The following operations are executed on the particle array during a time step:

1. All precursor particles are forced to produce delayed neutrons via forced decay [5], filling up the empty space in the particle array. A precursor particle is split into a decaying and a surviving part at the beginning of the time step, yielding a delayed neutron with weight \((1 - e^{-\lambda \Delta t})\) times the precursor weight, while a fraction of \(e^{-\lambda \Delta t}\) of its weight continues as a precursor.

2. Delayed neutrons are combed to the average weight of prompt neutrons with the simple comb described in Section 2.1.2.

3. Simulation of a time step is performed on live neutrons (on stacks \(n\) and \(d\) in Fig. 2.3). After the time step, the population of neutrons participating in the next step is changed by definition; there will be neutrons which did not participate in this time step but will in the next one, and vice versa. Hence neutrons from stack \(n\) can be reassigned to stack \(n'\) and the other way around. Precursor production during a time step is considered as a separate reaction; a neutron may emerge from fission as a precursor with probability \(\beta\), and if the sampled decay time falls outside the current interval, the
time step ends with a precursor sample substituting that particular neutron. Thus, part of neutrons
from stacks \( n \) or \( d \) before the time step will now be flagged as precursors (denoted by \( C_{new} \)).

4. Next, GUARDYAN sorts the particle array to an ordered set of neutrons, blank space and precursors.

5. Combing is performed on neutrons participating in the following time step, and separately on the
precursors, yielding a data set structurally analog to the initial array.

2.1.4 Initial conditions

Transient simulation with GUARDYAN requires an initial distribution of neutrons and precursors first.
One way to handle this is to give the initial conditions as user input. However, in most cases it is reasonable
to assume that the transient starts from an equilibrium state, since the time evolution of the system would
depend on its previous history otherwise due to precursor atoms. Equilibrium initial conditions can be
generated by a static MC simulation. In GUARDYAN, a power iteration is executed on the GPU to converge
on the fission source, then the simulation flow is changed to the default time step by time step methodology.
Thus, the prompt source for the transient simulation will simply be a snapshot of neutrons at the end of a
certain time step. To obtain an initial distribution of precursors, the array reserved for precursor samples
in the first column of Fig. 2 should be filled up. While this could have been done by replacing prompt
neutron samples in the static calculation with appropriate weighting, as described by Eq. (29) in Ref.
[2], GUARDYAN creates initial precursor samples in the time step by time step simulation flow. This is
motivated by the GPU bookkeeping of particle samples, depicted by Fig. 2. The most straightforward way
to obtain the initial samples filling up the first column in Fig. 2, is to keep all live neutrons from the static
calculation, run a few time steps, and fill in the precursor samples generated from live neutrons exploiting
that the precursor concentration is proportional to the live neutron density in equilibrium (cf.Eq. Eq. 2.1.6).
Precursor samples can be created from neutrons crossing the time boundaries, without deleting any live
neutron samples. At the end of a time step, precursors from randomly selected families are created in fissile
material where neutrons are located at that instant, with weight

\[
w_{\text{precursor}} = w_{\text{prompt}} \cdot K \frac{\beta_i \nu \Sigma_f}{\lambda_i} v
\]

(2.1.7)

where \( K \) is the number of precursor families, \( \beta_i \) is the fraction of delayed neutrons for the chosen family, \( \nu \)
is the fission neutron yield, \( \Sigma_f \) is the fission cross section, \( v \) is the speed of the prompt neutron and \( \lambda_i \) is the
decay constant of the precursor. This method is in line with the source generation of the DMC [2].

Source convergence is currently only monitored by effective multiplication factor \( k_{eff} \) and dynamic mul-
tiplication factor \( k_d \) (see Section 2.3.2 for definition). Since power iteration is not well suited for the GPU
architecture, the time–dependent methodology is intended to replace the iteration in the next version of
GUARDYAN, also, higher level tools for monitoring source convergence are under development and will be
available for the time–dependent mode.

2.2 Monte Carlo methods for neutron physics modeling

GUARDYAN is specifically intended to simulate time-dependent behavior of nuclear reactors. Treating
time dependence on the GPU have already been addressed in the previous chapter. Naturally, this takes up
only a small portion of the complete MC model of nuclear reactors. The implementation of e.g. handling
nuclear data, geometry or neutron physics modeling are now very standard in MC neutron transport due to
decades of work put into the development of static MC calculations. Neither the dynamic mode nor the GPU
hardware call for any major revision of the methodology and can therefore be adopted by GUARDYAN.
This chapter is dedicated to summarize these techniques. Novel contribution to the field does not lie in the
implementation details, but rather in the validation work done, considering that thousands of lines of coding
were performed. This will take up the last part of this chapter and can be found in [26] and [23].

2.2.1 Representation of geometry and nuclear data

The input logic of GUARDYAN is similar to the logic followed by production level codes MCNP [27],
Serpent [28] or OpenMC [29]. Cells are assumed to be homogeneous and can be defined by second order
bounding surfaces and boolean operators. Transformations, hexagonal and rectangular lattices, universes can
also be given in an input. Materials are defined by isotopic composition (atomic ratio or mass fraction) and
density. GUARDYAN uses nuclear data of ENDF-B-VII.1 [30] library. An ACE-format data [27] is generated
using NJOY [31] nuclear data processing system. For further information, please see the Appendix.

2.2.2 Sampling distance to collision on the GPU

GUARDYAN uses the Woodcock method (delta tracking) for sampling distance between collisions.
Surface-to-surface tracking, implemented in other MC codes like MCNP, is not particularly well suited
for GPU architecture, the method essentially fails on two levels. First, the calculation of ray-surface inter-
sections may be very time consuming, and second, a considerable amount of thread divergence may be added
by simulating neutrons in regions with various spatial dimension due to different computational cost.

The original delta tracking algorithm described by Woodcock [32] assumes a virtual material present in
the system with a scattering cross section (scattering without changing direction or energy) such that the
total cross section (real + virtual material) is the same in the whole volume. The resulting cross section is called the majorant cross section $\Sigma_m$. Path length sampling is thus extremely simple, as there is no need to track a particle surface to surface. To account for the bias in collision frequency, the particle interacts with the real material with probability $\Sigma_{\text{real}}/\Sigma_m$ and with the virtual material with probability $\Sigma_{\text{virtual}}/\Sigma_m$ at a collision site. Interaction with the virtual material is defined as a delta-scatter, when neither direction nor energy is altered.

A common drawback associated with the Woodcock method is called the localized heavy absorber problem, arising from the fact that a very small volume can contain a material with cross section of a few orders of magnitude higher than the average. This results in the majorant cross section chosen to be unreasonably high and producing virtual collisions unnecessarily often. In GUARDYAN, we use the Biased Woodcock framework that will be described in Section 3.2. Instead of $\Sigma_m$ we introduce the sampling cross section $\Sigma_{\text{samp}}(P)$ that is a positive function of phase space variables $P = (\vec{r}, E, \vec{\Omega})$ and $q(P)$ as the probability of real collisions taking values between 0 and 1. During the path length selection routine we sample the probability density function

$$ f_{BW}(s) = \Sigma_{\text{samp}}(\vec{r} + \vec{\Omega}s, E, \vec{\Omega}) \exp\left(-\int_0^s \Sigma_{\text{samp}}(\vec{r} + \vec{\Omega}t, E, \vec{\Omega}) dt\right) $$

(2.2.1)

and with probability $q(\vec{r} + \vec{\Omega}s, E, \vec{\Omega})$ we sample a real collision, otherwise a delta-scatter is simulated. To account for this modification particle weight must be set to

$$ w' = w \frac{\Sigma(P')}{q(P')\Sigma_{\text{samp}}(P')} $$

(2.2.2)

if a true collision happened, and

$$ w' = w \frac{1 - \Sigma_{\text{samp}}(P')}{1 - q(P')} $$

(2.2.3)

if the collision was a delta-scatter, where $P' = (\vec{r} + \vec{\Omega}s, E, \vec{\Omega})$ is the post-collision coordinate. With this algorithm, the above mentioned local heavy absorber problem can be circumvented. Additionally, the two parameters $\Sigma_{\text{samp}}$ and $q$ can be chosen arbitrarily (taking the above restrictions into account), and the algorithm is still unbiased (for proof see Section 3.2). Without the loss of generality we wrote $\Sigma_{\text{samp}}(P)$, but most of the times we want $\Sigma_{\text{samp}}$ to be constant or piecewise constant in the space variable in order to reduce the cost of sampling. In GUARDYAN, we use piecewise constant sampling cross section that allows us to sample Eq. 3.2.2 with the inverse cumulative method, the exact algorithm will be detailed in Sec 3.7.

### 2.2.3 Interaction physics modeling

GUARDYAN considers neutron interactions with MT number listed in Table 6.2. To prevent branching and terminating neutron histories the probability of these reactions is non-analog, it is in line with what has
been previously derived in Section 2.1.2. Survival biasing is applied thus capture is never simulated, fission reactions are intentionally oversampled and neutron production is treated with weight change. GUARDYAN samples reaction of type $i$ with probability of

$$P_i = \frac{\nu \Sigma_i}{\sum_j (\nu \Sigma_j)_j}$$

(2.2.4)

where we use the notation $\nu \Sigma$ with a redefined $\nu$ as follows: $\nu = 0$ for capture reaction (MT=101), $\nu = 1$ for non-multiplying reactions, $\nu = x$ for ($n,xn$) reactions, and $\nu = \nu_{tot}$ for fission reactions. and the overall weight change is

$$w' = w \frac{\sum_j (\nu \Sigma_j)_j}{\Sigma_i}$$

(2.2.5)

which takes both survival biasing, non-analog probabilities and implicit fission into account and yields an unbiased scheme. When a $n,xn$ reaction happens, the nucleus goes to an excited state, and each produced neutron has different typical energy as the nucleus gradually relaxes to steady state with each leaving neutron. Since GUARDYAN produces only one secondary particle ($n,xn$ reactions are also treated with weight change like fission), we choose randomly from typical outgoing energy distributions. In practice, instead of one interaction we assume $x$ different interactions with cross sections of the original cross section multiplied by the probability of producing a neutron with the associated energy distribution. This probability of course depends on the incident neutron energy.

For fission reactions the total number of produced neutrons $\nu_{tot}$ must be calculated, which depends on the energy of the incoming neutron. ENDF data provides two representations to deal with this. First is to treat $\nu_{tot}$ as an $N$-th order polynomial function of the incoming energy:

$$\nu_{tot}(E) = \sum_{i=0}^{N} c_i E^i.$$ 

(2.2.6)

Another option is to use a tabulated function with a certain interpolation law. The number of prompt and delayed neutrons $\nu_p$ and $\nu_d$ are also available from ENDF, but $\nu_d$ is only provided by the tabulated function representation. The delayed neutron yield $\beta$ is calculated as

$$\beta = 1 - \frac{\nu_p(E)}{\nu_{tot}(E)}.$$ 

(2.2.7)

A secondary particle may emerge from a collision as a neutron or a precursor. Other particles are not tracked in the current version of GUARDYAN. Precursor production is considered as a separate reaction during transport, occurring with probability $\beta$ if a fission event is sampled. In case the secondary particle is a neutron, the outgoing angle and energy is determined by sampling laws according to ACE-data. The sampling procedures applied in GUARDYAN are similar to the ones used in the OpenMC [29] or MCNP [27] MC codes. These methods are summarized in the Appendix. In some special cases, at low neutron
energies, we need to consider the motion of the target nucleus, or use precalculated $S(\alpha, \beta)$ tables to account for chemical bounds and crystalline effects (also see the Appendix).

2.3 Validation and verification

GUARDYAN, being built from scratch, needs proper validation and verification (V&V) to check for possible errors in the code. Code-to-code comparison offers a nice possibility for V&V and is a very standard way to go, usually, novel implementations are challenged against other already validated high-fidelity MC codes e.g. MCNP. It is impossible however to have GUARDYAN completely validated this way, since these reference codes have limited ability to do time-dependent or dynamic calculations. In this section we restrict ourselves to scenarios where reference solutions can actually be obtained. Code-to-code comparison was therefore able to provide proof of the correct implementation of interaction physics rather than the proper treatment of time dependence. Full kinetic capabilities were benchmarked against measurement data, also, code-to-code comparison was performed against a deterministic solver involving a strong, local transient. These will be described in Chapter 5. Now, first we analyze problems which can be treated by MCNP as a source–detector problem and MCNP is able to provide time-dependent results, and second, GUARDYAN is modified to run in static mode making it possible to include criticality benchmarks in the validation. In the first case we prepared approximately 2000 time-dependent scenarios including various isotopes to make sure all kind of neutron interactions occur. When fissionable materials were present in a system, it was designed to be very subcritical in order to make an MCNP run feasible. This investigation produced a comparison of differential quantities, overall 445000 data points were compared. In the second case, some criticality benchmark scenarios were selected from the International Handbook of Evaluated Criticality Safety Benchmark Experiments [33] and $k_{eff}$ values were compared between static GUARDYAN and MCNP.

2.3.1 Differential comparison

In roughly 2000 separate runs in spherical geometry including different isotopic composition and launching neutrons at $t = 0$ with various starting energies, we tallied the time evolution of energy dependent flux on the outer surface of a sphere. Only one time step was executed and tallies were then binned into 9 time bins. One example case can be seen in Fig 2.4. Erroneous implementation of interaction physics was identified by calculating the fraction of data points outside 1,2 and 3 $\sigma$ margin, and was corrected if large differences were observed.
Figure 2.4: Example calculation of the verification work comparing the time evolution of fluence spectrum to MCNP results. In this figure only one example is illustrated via simulation of neutrons starting with 1 MeV at t=0 at the center of a homogeneous sphere with radius of 101.105 cm containing a single isotope, $^{40}\text{K}$. The whole verification work consists of simulations on 412 spherical systems each containing different nuclides. Radius of the sphere was always adjusted to typical free flight path lengths.

After several revisions of the code, results from GUARDYAN showed good agreement within statistics with MCNP results. To confirm that, reduced chi-square statistics were calculated on data sets of GUARDYAN and MCNP by

$$\chi^2 = \frac{1}{N} \sum_{i=1}^{N} \frac{(y_{M,i} - y_{G,i})^2}{\sigma_{M,i}^2 + \sigma_{G,i}^2}$$

(2.3.1)

where $N$ is the number of data points for a single isotope and a single starting energy (e.g. $N=9\times24=216$ points are plotted in Fig. 2.4), $\sigma$ stands for the standard deviation of the MC estimates (M for MCNP, G for GUARDYAN) and $y$ is the tally result. A histogram of this measure is plotted in Fig. 2.5. Values around one imply excellent agreement; values above 2 are rare confirming that the two implementations agree.
For lower energies, in vast majority of the isotopes the dominant interaction is elastic scattering. Although, for fissionable materials, ACE laws 3, 4, 7, 9, 11, 44, 61 and 66 occur even at low energies and for higher energies, many isotopes undergo such interactions, the dominant interaction is still elastic scattering in this model. To ensure that promising $\chi^2$ values are not only caused by a properly implemented elastic scattering algorithm, we have registered the number of samplings of certain ACE laws. To assess the impact of a certain ACE law on the $\chi^2$ value we have calculated the (Pearson) correlation coefficient between $x_i$ and $y_i$ data points with $\bar{x}$ and $\bar{y}$ averages respectively:

$$corr = \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2 \sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2}}} \quad (2.3.2)$$

Table 2.1 shows the correlation of the number of a certain ACE law sampling with the $\chi^2$ values for different neutron starting energies. Near zero correlation coefficients indicate no correlation, near 1 or -1 strong linear correlation. Elastic scattering appears always with near zero coefficients with consistently negative sign. At $10^{-8}$ MeV and 18 MeV the correlation is greater but still negative; we may interpret this as elastic scattering contributing to better match of results. ACE Law 3 (inelastic level scattering), ACE Law 4 (continuous tabular distribution) and ACE Law 61 (correlated energy and angle distribution) show a clearly nonzero and often positive correlation indicating that the GPU implementation differ from MCNP. All three of these sampling processes happen with high energy neutrons, for low starting neutron energies the interactions happen after fission has occurred. Number of sampled interactions with these laws shows
that without fissile material present, none of these sampling processes are called.

<table>
<thead>
<tr>
<th></th>
<th>$10^{-8}$ MeV</th>
<th>$10^{-6}$ MeV</th>
<th>$10^{-3}$ MeV</th>
<th>1 MeV</th>
<th>18 MeV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic scattering</td>
<td>-0.21372</td>
<td>-0.01053</td>
<td>-0.05259</td>
<td>-0.07189</td>
<td>-0.26171</td>
</tr>
<tr>
<td>ACE Law 3</td>
<td>0.4743</td>
<td>0.43175</td>
<td>0.41582</td>
<td>-0.05454</td>
<td>-0.15467</td>
</tr>
<tr>
<td>ACE Law 4</td>
<td>0.30463</td>
<td>0.32031</td>
<td>0.67385</td>
<td>0.45868</td>
<td>0.17577</td>
</tr>
<tr>
<td>ACE Law 7</td>
<td>-0.00399</td>
<td>-0.00405</td>
<td>0.01045</td>
<td>-0.00808</td>
<td>-0.00166</td>
</tr>
<tr>
<td>ACE Law 9</td>
<td>-0.00872</td>
<td>-0.00649</td>
<td>0.01524</td>
<td>-0.00716</td>
<td>0.03139</td>
</tr>
<tr>
<td>ACE Law 11</td>
<td>-0.01198</td>
<td>0.00382</td>
<td>0.05435</td>
<td>0.00464</td>
<td>0.00769</td>
</tr>
<tr>
<td>ACE Law 44</td>
<td>0.00206</td>
<td>-0.00254</td>
<td>0.04434</td>
<td>0.14184</td>
<td>-0.03327</td>
</tr>
<tr>
<td>ACE Law 61</td>
<td>0.31364</td>
<td>0.21316</td>
<td>0.50553</td>
<td>-0.00556</td>
<td>-0.09562</td>
</tr>
<tr>
<td>ACE Law 66</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>-0.00607</td>
</tr>
</tbody>
</table>

Table 2.1: Correlation Coefficients of $\chi^2$ values with number of interactions sampled by a certain ACE law

ACE Law 3 and 61 for higher starting neutron energies show almost zero correlation coefficient, i.e. at energies where the ratio of sampling number to elastic scattering is the highest, yielding the conclusion that the these law sampling schemes are not suspicious. For fissile materials the simulation chain lengths vary with number of collisions in a chain following geometrical distribution yielding a less robust estimation. ACE Law 4 shows the same behavior but higher energies still show higher correlation coefficients while this sampling process is very often called for most of the isotopes. Further research will target the solution of this issue.

### 2.3.2 Integral comparison

Several criticality benchmarks were selected to test GUARDYAN by investigation of integral reactor quantities such as the effective multiplication factor $k_{eff}$. Eight benchmark scenarios were chosen from the International Handbook of Evaluated Criticality Safety Benchmark Experiments [33], including fast, intermediate energy and thermal system with various complexity. In order to obtain the $k_{eff}$ of these systems, the simulation flow of GUARDYAN was altered by tracking neutrons generation-by-generation instead of the default time step methodology. Although this change slows down simulation flow, we found that criticality calculations were also faster with GUARDYAN on the GPU than MCNP6 calculations on a similarly priced CPU, showing the advantages of using GPU architecture.

Table 2.2 shows $k_d$ and $k_{eff}$ values calculated by GUARDYAN with comparison to MCNP6 results. Although the $k_{eff}$ of sample calculations can be found in the Handbook [33] as well, MCNP6 cycles were
GUARDYAN was run on an Nvidia GeForce GTX 1080 GPU, simulating 100 generations (20 inactive and 80 active) of 2 097 152 ($2^{21}$) neutrons.

MCNP6 was run on an Intel Xeon E5-1650 v4 CPU using using hyperthreading (12 threads), simulating 12800 generations (560 inactive and 10240 active) of 16384 ($2^{14}$) neutrons.

The same cross section library, ENDF-B-VII.1, was used for both codes.

While the results summarized in Table 2.2 shows good agreement between GUARDYAN and MCNP, it is important to note that good $k_{eff}$ values does not guarantee that the time dependence of the system is correctly coded. For verifying the delayed neutron and in general the time dependence of the modeling, one step time average of the dynamic multiplication factor defined by Eq. (44) in [34] was calculated by

$$k_d = \frac{\sum_i P_i^{\Delta t}}{\sum_i L_i^{\Delta t}}$$

i.e as the fraction of total neutron production and neutron loss during a time step (the sum goes for all collisions in the time step). At $k_{eff} = 1$, $k_d$ should also be 1, meaning that the power evolution is constant.

<table>
<thead>
<tr>
<th>benchmark</th>
<th>program</th>
<th>$k_d$</th>
<th>$\sigma_c$</th>
<th>$k_{eff}$</th>
<th>$\sigma_c$</th>
<th>$t(s)$</th>
<th>$FoM(10^3 \frac{1}{s})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>heu-met-fast-001</td>
<td>MCNP6</td>
<td>–</td>
<td>–</td>
<td>0.99981</td>
<td>0.00005</td>
<td>602</td>
<td>664</td>
</tr>
<tr>
<td>GUARDYAN</td>
<td>1.000065</td>
<td>0.00023</td>
<td>0.99981</td>
<td>0.00009</td>
<td>30</td>
<td>4507</td>
<td></td>
</tr>
<tr>
<td>pu-met-fast-002</td>
<td>MCNP6</td>
<td>–</td>
<td>–</td>
<td>1.00010</td>
<td>0.00005</td>
<td>555</td>
<td>721</td>
</tr>
<tr>
<td>GUARDYAN</td>
<td>1.000036</td>
<td>0.000077</td>
<td>1.00006</td>
<td>0.00010</td>
<td>23</td>
<td>4527</td>
<td></td>
</tr>
<tr>
<td>heu-comp-inter-003</td>
<td>MCNP6</td>
<td>–</td>
<td>–</td>
<td>1.00792</td>
<td>0.00004</td>
<td>1993</td>
<td>139</td>
</tr>
<tr>
<td>GUARDYAN</td>
<td>1.001485</td>
<td>0.000099</td>
<td>1.00809</td>
<td>0.00007</td>
<td>426</td>
<td>429</td>
<td></td>
</tr>
<tr>
<td>u233-sol-inter-001</td>
<td>MCNP6</td>
<td>–</td>
<td>–</td>
<td>0.98543</td>
<td>0.00008</td>
<td>1426</td>
<td>110</td>
</tr>
<tr>
<td>GUARDYAN</td>
<td>0.999707</td>
<td>0.00058</td>
<td>0.98506</td>
<td>0.00009</td>
<td>236</td>
<td>501</td>
<td></td>
</tr>
<tr>
<td>u233-sol-therm-001</td>
<td>MCNP6</td>
<td>–</td>
<td>–</td>
<td>1.00119</td>
<td>0.00004</td>
<td>7665</td>
<td>82</td>
</tr>
<tr>
<td>GUARDYAN</td>
<td>1.000100</td>
<td>0.00013</td>
<td>1.00068</td>
<td>0.00006</td>
<td>1527</td>
<td>188</td>
<td></td>
</tr>
<tr>
<td>leu-sol-therm-001</td>
<td>MCNP6</td>
<td>–</td>
<td>–</td>
<td>1.01180</td>
<td>0.00006</td>
<td>2938</td>
<td>95</td>
</tr>
<tr>
<td>GUARDYAN</td>
<td>1.004618</td>
<td>0.00061</td>
<td>1.01103</td>
<td>0.00006</td>
<td>622</td>
<td>432</td>
<td></td>
</tr>
<tr>
<td>leu-sol-therm-002</td>
<td>MCNP6</td>
<td>–</td>
<td>–</td>
<td>0.99983</td>
<td>0.00004</td>
<td>6455</td>
<td>97</td>
</tr>
<tr>
<td>GUARDYAN</td>
<td>1.000047</td>
<td>0.0003</td>
<td>0.99954</td>
<td>0.0005</td>
<td>3881</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>leu-comp-therm-008</td>
<td>MCNP6</td>
<td>–</td>
<td>–</td>
<td>0.99972</td>
<td>0.00005</td>
<td>6311</td>
<td>63</td>
</tr>
<tr>
<td>GUARDYAN</td>
<td>1.000010</td>
<td>0.00031</td>
<td>0.99945</td>
<td>0.00006</td>
<td>1085</td>
<td>240</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Criticality benchmarks from [33] with GUARDYAN and MCNP6

GUARDYAN was run on an Nvidia GeForce GTX 1080 GPU, simulating 100 generations (20 inactive and 80 active) of 2 097 152 ($2^{21}$) neutrons.

MCNP6 was run on an Intel Xeon E5-1650 v4 CPU using using hyperthreading (12 threads), simulating 12800 generations (560 inactive and 10240 active) of 16384 ($2^{14}$) neutrons.

The same cross section library, ENDF-B-VII.1, was used for both codes.
in time. Indeed, Table 2.2 shows that when $k_{\text{eff}}$ is very close to 1, GUARDYAN also provides $k_d$ around 1. Deviations between $k_{\text{eff}}$ and $k_d$ can only be observed when the system is further from criticality (leu-sol-therm-001, u233-sol-inter-001). In general $k_d$ should not equal the effective multiplication factor, since $k_{\text{eff}}$ is calculated by renormalizing the fission source from generation to generation, in other words it is the solution to the $k$-eigenvalue equation and not the true time-dependent Boltzmann equation [35]. $k_d$ is calculated with the true dynamic flux of the system, instead of an artificial static flux. The difference can be better understood from literature (see either [34] or [35]).

Runtime $t(s)$ in Table 2.2 shows total execution time, $\sigma_r$ denotes the relative error. The efficiency measure, FoM (Figure of Merit), is calculated by

$$\text{FoM} = \frac{1}{\sigma_r^2 t}.$$  

(2.3.4)

Based on this measure, we found that GUARDYAN performed better than MCNP6, with execution times significantly lower and standard deviations in the same order of magnitude. An exception was found to be benchmark problem leu-sol-therm-002, where execution time of GUARDYAN was noticeably slower. Analysis of this benchmark revealed that $s(a,b)$ data in this problem is very detailed, thus this slowdown can be traced back to the ineffective implementation of $s(a,b)$ treatment in GUARDYAN. More efficient algorithms are currently under development, and should solve this issue.
Chapter 3

A framework for the efficient sampling of distance to collision

When it comes to particle tracking on GPUs, one may generally think of ray tracing, a 3D rendering technique in computer graphics. Most of ray tracing problems can be solved with relative ease, even slightly biased results can be acceptable when the geometry is exceptionally complicated (e.g. inhomogeneous media rendering with ray marching). In nuclear engineering however, unbiasedness is fundamental. In MC models of reactor cores, geometry dimensions are usually small compared to neutron mean free path, thus ”ray tracing”, which is rather termed surface-to-surface tracking in nuclear engineering, can become tedious as many ray-surface intersections need to be calculated between collision sites. On the GPU, surface-to-surface tracking has one more downside that is the thread divergence arising from the spatial disparity of geometric complexity, i.e. neutrons in different regions encounter different spatial dimensions in geometry detail. Thus, for GUARDYAN, we choose the Woodcock method or delta tracking instead, that allows material boundaries to be ignored, at the expense of randomly producing fictitious collisions (null-collisions). A variance reduction framework termed the Biased Woodcock (BW) framework was constructed, extending the possibilities to better utilize GPU computing power in this respect.

The application of Woodcock tracking is not unfamiliar in nuclear engineering, but implementations yet failed to recognize possibilities for continuous parametrization and thus potential for optimization. Remedying this issue, several optimization strategies are investigated in this section, based on variance analysis and the estimation of computational cost. Findings presented here are not specific to dynamic MC simulations and sometimes not even to reactor physics. This is partly due to the fact that Woodcock tracking is widely used and also, to the extent of research done on this topic reported in papers [36], [37], [38] and [39].
3.1 Introduction

Null-collisions were originally introduced to MC particle tracking methods as an alternative method to sample free path in inhomogeneous media. Based on von Neumann’s rejection technique [40], the first description of the method was published by Woodcock et al. [32] in nuclear engineering field and by Skullerud et al. [41] regarding plasma physics. The method has become popular under different names like Woodcock or delta tracking, fictitious or pseudo scatter. Several proposals have been made to improve the algorithm since its introduction, however, the original method is still frequently used despite of its drawbacks, e.g. in Serpent [42], MORET [43], MONK [44], and WARP [18] MC codes. Many of the developments focus on bypassing the rejection procedure suggesting a weighted tracking scheme instead [45], [46], and [47]; other publications also consider the use of non-majorant cross section for free path sampling, thus circumventing the heavy absorber problem (generating large amount of null-collisions due to a local extremity in cross sections) at the cost of introducing negative particle weights [48], [49], [39] and [37]. Beside nuclear engineering and plasma physics, the method found its way to other disciplines like computer graphics [50], radiative transfer methods [51] and medical physics [52], [53], largely because handling complex geometries (accessing the total cross section is required frequently and/or associated with high computational cost) becomes exceedingly simple. This fairly extensive literature and insufficient crosstalk between disciplines resulted in the reinvention of several variants of the method. In this section a comprehensive Woodcock-type particle tracking framework is presented that is considered universal as it covers all of the variants of the method published before. The framework enables MC developers to tune the algorithm via two parameters \( \Sigma_{\text{samp}} \) and \( q \), the former being referred to as the sampling cross section, and the latter denoting the probability of true collisions. Both parameters can be considered varying with the phase space variables. Possible modifications of the original Woodcock tracking algorithm proposed in previous studies can be interpreted as certain realizations of Alg. 1 (described in Section 3.2), i.e. specific choices of parameters \( \Sigma_{\text{samp}} \) and \( q \) returns the algorithms suggested before. We show, that none of these choices are optimal in terms of either variance or efficiency, and investigate how more efficient choices can be made.

3.2 The Biased Woodcock method

The pseudo-code of the suggested algorithm is shown in Alg. 1. On the right hand side, the original Woodcock method (Alg. 2) can be seen. The concept of the original algorithm is to choose a majorant \( \Sigma_m \) that is greater than or equal to the maximum of cross sections of the problem, and sample distance to a tentative collision site according to the probability density function (pdf) of Eq. 3.2.1, that is a simple
exponential distribution. Then, the algorithm accepts a tentative collision site as a true collision with probability \( \frac{\sum m}{\sum m} \), otherwise a rejection is performed, the particle suffers a null-collision, also known as delta-scatter or pseudo collision, meaning that it goes on with direction or energy unchanged. The unbiasedness of the method was proved in Ref. [54].

\[
f_W(s) = \sum_m e^{-(\sum_m s)} \quad (3.2.1)
\]

Notice that the process of sampling distance to collision relies on two quantities: the majorant cross section and the probability to accept a true collision. As it turns out, these quantities can be replaced by independent functions of phase space variables \( P = (\vec{r}, E, \vec{\Omega}) \), and account for the introduced bias with properly chosen particle weight \( w \). The new cross section replacing the majorant will be denoted by \( \Sigma_{samp}(P) \) and the new probability for acceptance will be denoted by \( q(P) \). \( \Sigma_{samp} \) must be a non-negative function, while \( q \) must take values from the interval \((0, 1)\). These changes result in the Biased Woodcock tracking, depicted by Alg. 1.

**Algorithm 1:** Biased Woodcock tracking

\[
\begin{align*}
\text{while } \neg \text{collided do} \\
& s = \text{sample}(f_{BW}); \\
& P' = (\vec{r} + \vec{\Omega}s, E, \vec{\Omega}); \\
& \text{if } \text{rand()} < q(P') \text{ then} \\
& \quad \text{collided} = \text{true}; \\
& \quad w = w \frac{\Sigma(P')}{\Sigma_{samp}(P')q(P')}; \\
& \text{else} \\
& \quad w = w \frac{1 - \Sigma(P')}{1 - q(P')}; \\
\end{align*}
\]

**Algorithm 2:** Original Woodcock tracking

\[
\begin{align*}
\text{while } \neg \text{collided do} \\
& s = \text{sample}(f_W); \\
& P' = (\vec{r} + \vec{\Omega}s, E, \vec{\Omega}); \\
& \text{if } \text{rand()} < \frac{\Sigma(P')}{\sum_m} \text{ then} \\
& \quad \text{collided} = \text{true}; \\
& \text{else} \\
& \quad \text{do nothing}; \\
\end{align*}
\]

In BW tracking, instead of Eq. 3.2.1, the probability density of Eq. 3.2.2 is to be sampled to generate a new tentative collision site. \( \Sigma_{samp} \) can be chosen constant, then again, a simple exponential distribution has to be sampled. It will be later shown however, that this would not be optimal. The sampling strategy also offers an alternative free path selection procedure in systems where the true cross section varies continuously [55], [48]. Proof that the algorithm still gives unbiased estimates can be found in Ref. [56], or can be considered as a special case in the proof given by Sec. 3.3.

\[
f_{BW}(s) = \Sigma_{samp}(\vec{r} + \vec{\Omega}s, E, \vec{\Omega})e^{\left(-\int_0^s \Sigma_{samp}(\vec{r} + \vec{\Omega}t, E, \vec{\Omega})dt\right)} \quad (3.2.2)
\]

By replacing collision acceptance probability \( \frac{\Sigma}{\sum_m} \) with \( q \), several limitations of the original algorithm are resolved. From a practical point of view, the followings are important, with particular respect to implementation on the GPU:
• The sampling cross section $\Sigma_{\text{samp}}$ no longer needs to be a majorant with respect to the true cross section $\Sigma$. The majorant criterion arises from the fact that in the original algorithm, probability $\frac{\Sigma}{\Sigma_m}$ would be greater than 1, in case of non-majorant [48], [49]. This can be quite convenient, as a major drawback associated with delta tracking (that it generates null-collisions unnecessarily often when a heavy absorber is present in the region) is thus surpassed. In GUARDYAN, where accessing the cross section at a certain collision site takes up the major part of computational time, benefits are all the more expressed. It must be noted however, that choosing a non-majorant sampling cross section may result in particle weight changing sign upon a null-collision [48].

• Even the true cross section does not have to be non-negative, as it does not cause collision acceptance probability to be negative. This yields an alternative approach for particle transport in stochastic media [57], [58] or for using the concept of control variates in radiance estimation [39].

• Two independent variance reduction parameters $\Sigma_{\text{samp}}$ and $q$ are now available as inputs to the simulation. It is possible to set the expected number of collisions prior to any calculations, as the collision frequency no longer depends on the true cross sections $\Sigma$, but is determined by $\Sigma_{\text{samp}}$ and $q$. Or alternatively, one can set these sampling parameters on-the-fly, as the compensation of particle weight is carried out after each collision (pseudo or true), guaranteeing unbiasedness. Continuous parametrization provides for numerous optimization strategies and flexibility to adjust the algorithm for the better distribution of GPU resources.

• The BW tracking scheme replaces the rejection procedure with a weighted tracking scheme not completely unlike the algorithm described by Ref. [46]. As a consequence, null-collisions affect the statistics of the simulation (that is also true for Ref. [46]), instead of being merely tools for simplifying the sampling process. In fact, generating null-collisions contributes to the accurate estimation of total attenuation of particle flux. While using collision estimator, it can even be considered to take tallies on null-collisions.

The price to pay for the freedom in choosing the probability of a true collision is in the evolution of particle weights. Notice, that when a tentative collision site is accepted as true, the weight is multiplied by the ratio of ‘probabilities’ $\frac{\Sigma}{\Sigma_{\text{samp}}}$ and $q$ to avoid any bias. The same is carried out upon a null-collision but with complemernter ‘probabilities’. If $q \neq \frac{\Sigma}{\Sigma_{\text{samp}}}$, then one of the weight modification factors becomes larger than unity (at least in absolute value), thus increasing the particle weight, while the other factor will be less than unity. Larger differences between $q$ and $\frac{\Sigma}{\Sigma_{\text{samp}}}$ tend to cause larger fluctuations in particle weight, however, it should be emphasized that this does not necessarily cause the variance of the final score to break
down, as demonstrated in Section 3.6.

Extreme cases $q = 0$ and $q = 1$ may also be considered in Alg. 1, although they do not yield unbiased estimates in general. When $q \to 0^+$ (and $\Sigma_{\text{samp}}$ is fixed), null-collisions are oversampled and particle weights mostly reflect the exponential attenuation of the traversed medium. Very few particle would undergo true collisions, but they would take on very large weights. If $q = 0$, true collisions would never occur, and the nonanalog simulation would be biased, unless we wish to estimate the attenuation of particle flux along a ray (as in Section 3.6). Any other estimates would be false. When $q \to 1^-$, the opposite happens: true collisions would be oversampled at the expense of null-collisions. As pseudo-collisions are important in order to agree with exponential attenuation, the particle weight would reflect little of this information. Choosing $q = 1$ would yield biased estimates, unless the choice $\Sigma_{\text{samp}} = \Sigma$ is made, but then, we recover the analog simulation scheme.

### 3.3 Mathematical proof of correctness

To show that Alg. 1 is an unbiased way of generating collision sites, we prove two theorems. First theorem states that distortion of the probability density of distance between collisions is compensated by proper weighting. The second theorem states that also, the density of real collisions is correct, i.e. the algorithm produces the analog density $\Sigma(x) \exp\left(-\int_0^x \Sigma(x')dx'\right)$.

**Theorem 1.** Let $\Lambda$ be a random variable of the distance to the first (real) collision generated by Alg. 1. Distribution of $\Lambda$ gives the correct cumulative distribution function (cdf):

$$P(\Lambda < x) = 1 - \exp\left(-\int_0^x \Sigma(x')dx'\right). \tag{3.3.1}$$

**Proof.** Rewrite probability in Eq 3.3.1 as the expectation of the indicator variable:

$$P(\Lambda < x) = 1 - P(\Lambda > x) = 1 - E(1_{\Lambda > x}) \tag{3.3.2}$$

The expectation $E(1_{\Lambda > x})$ equals the expected weight of a particle reaching $x$ without suffering any (real) collision. The expected value of particle weight $w$ at $x$ can be expanded according to the rule of total expectation (tower rule):

$$E[w](x) = \sum_{k=0}^{\infty} E(w \mid K = k)p_K(k) \tag{3.3.3}$$

where $K$ is the random variable of the number of collisions (null-collisions and real ones) on the interval $[0, x]$. $p_K$ denotes the probability density function of $K$. Now the number of collisions is fixed on interval $[0, x]$. Let
\( X = (X_1, ..., X_k) \) denote the vector of random variables corresponding to collision sites and suppose, that the collisions took place at \( x_1, x_2, ..., x_k \). Applying the tower rule again, we get Eq. 3.3.4:

\[
E[w](x) = \sum_{k=0}^{\infty} \left[ \int ... \int E(w \mid K = k, X_1 = x_1, ..., X_k = x_k)p_{X|K}(x_1, ..., x_k)dx_1...dx_k \right] p_K(k) \quad (3.3.4)
\]

where \( p_{X|K} \) denotes the conditional joint probability density function of collision sites. Let \( V \) denote the number of null-collisions and apply the tower rule once more, conditioning for whether or not a real collision has happened at any \( x_1, ..., x_k \). Thus, we obtain:

\[
E[w](x) = \sum_{k=0}^{\infty} \left[ \int ... \int E(w \mid K = k, X_1 = x_1, ..., X_k = x_k, V = k)P(V = k \mid K = k, X_1 = x_1, ..., X_k = x_k) \right]
\]

\[
p_{X|K}(x_1, ..., x_k)dx_1...dx_k + \int ... \int E(w \mid K = k, X_1 = x_1, ..., X_k = x_k, V \neq k) \cdot P(V \neq k \mid K = k, X_1 = x_1, ..., X_k = x_k)p_{X|K}(x_1, ..., x_k)dx_1...dx_k \right] p_K(k) \quad (3.3.5)
\]

In Eq 3.3.2, we considered only the particles that has not suffered any real collisions, thus the second conditional expectation vanishes. As for the first expectation, variable \( w \mid K = k, X_1 = x_1, ..., X_k = x_k, V = k \) is not random any more, as the final weight of the particle at the end of the path is uniquely specified by the number of collisions \( k \) and collision sites \( x_1, ..., x_k \): the weight is the product of \( 1 - \frac{\Sigma(x_i)}{1 - q(x_i)} \) for \( i = 1...k \). The same way, the probability that every tentative collision is null-collision is the product of \( 1 - q(x_i) \) for \( i = 1...k \). Then Eq. 3.3.5 yields:

\[
E[w](x) = \sum_{k=0}^{\infty} \left[ \int ... \int \prod_{i=1}^{k} \frac{1 - \frac{\Sigma(x_i)}{\Sigma_{samp}(x_i)}}{1 - q(x_i)} \prod_{i=1}^{k} (1 - q(x_i))p_{X|K}(x_1, ..., x_k)dx_1...dx_k \right] p_K(k) =
\]

\[
= \sum_{k=0}^{\infty} \left[ \int ... \int \prod_{i=1}^{k} \left( 1 - \frac{\Sigma(x_i)}{\Sigma_{samp}(x_i)} \right) p_{X|K}(x_1, ..., x_k)dx_1...dx_k \right] p_K(k) \quad (3.3.6)
\]

In Eq. 3.3.6 two distributions are yet to be determined: the conditional distribution \( p_{X|K} \) and the probability density function of \( K (p_K) \). To answer these we use some basic properties of the inhomogeneous Poisson point process (HPP). That is because tentative collision sites (as random points in space) form an inhomogeneous Poisson process with rate \( \Sigma_{samp}(t) \), \( t \in [0, x] \). Proof can be given by formulating the conditional distribution of jumping farther than \( x_{i+1} \) assuming that we are currently in \( x_i \):

\[
P(X_{i+1} > x_{i+1} \mid X_i = x_i) = \exp \left( - \int_{x_i}^{x_{i+1}} \Sigma_{samp}(x')dx' \right). \quad (3.3.7)
\]

36
Eq. 3.3.7 holds simply because the pdf $\Sigma_{\text{samp}}(x_i) \exp \left(- \int_{x_{i+1}}^{x_i} \Sigma_{\text{samp}}(x') dx' \right)$ is sampled. Eq. 3.3.7 is also equivalent to the probability of having no events in an IHPP on interval $[x_i, x_{i+1}]$, thus the distance between collisions has the same distribution as inter arrival times of the IHPP. The number of collisions on two disjoint interval are clearly independent of each other and probability densities generating collision sites are the same as in an IHPP, therefore the algorithm generates an IHPP with rate $\Sigma_{\text{samp}}(t)$.

An important property of the inhomogeneous Poisson process with rate $\Sigma_{\text{samp}}(t)$ is that if the number of events is fixed on some interval $[0, x]$, then the distribution of jump epochs (collision sites) $x_1, x_2, ..., x_k$ is the same as the order statistics of $k$ independent random variables $U_1 < U_2 < ... < U_k$ with density functions $p_{U_i}(u) = \frac{\Sigma_{\text{samp}}(u)}{\int_0^x \Sigma_{\text{samp}}(x') dx'}$ for every $i$. As the order of collisions is irrelevant from the perspective of the final weight, the expectation in Eq. 3.3.6 can be calculated via replacement of variables $X_i$ for these independent, identically distributed variables $U_i$. This yields:

$$E[w](x) = \sum_{k=0}^{\infty} \left( 1 - \int_0^x \frac{\Sigma(u)}{\Sigma_{\text{samp}}(u)} \frac{\Sigma_{\text{samp}}(u)}{\int_0^x \Sigma_{\text{samp}}(x') dx'} du \right)^k p_K(k) = \sum_{k=0}^{\infty} \left( 1 - \frac{\int_0^x \Sigma(x') dx'}{\int_0^x \Sigma_{\text{samp}}(x') dx'} \right)^k p_K(k). \quad (3.3.8)$$

The number of tentative collisions ($K$) follows Poisson distribution with mean $\int_0^x \Sigma_{\text{samp}}(x') dx'$ by definition of the Poisson process. To shorten further calculations we recognize that Eq. 3.3.8 equals the probability generating function (PGF) of $K$, more precisely the expected weight equals the PGF at $1 - \frac{\int_0^x \Sigma(x') dx'}{\int_0^x \Sigma_{\text{samp}}(x') dx'}$. The PGF of the Poisson distributed variable $K$ with mean $\int_0^x \Sigma_{\text{samp}}(x') dx'$ is:

$$G(z) = \sum_{k=0}^{\infty} z^k p_K(k) = \exp \left( (z-1) \int_0^x \Sigma_{\text{samp}}(x') dx' \right). \quad (3.3.9)$$

Exploiting the PGF it is apparent that

$$E[w](x) = G \left( 1 - \frac{\int_0^x \Sigma(x') dx'}{\int_0^x \Sigma_{\text{samp}}(x') dx'} \right) = \exp \left( - \int_0^x \Sigma(x') dx' \right) \quad (3.3.10)$$

Recall Eq. 3.3.2, where the expectation of the indicator variable equals the expected weight of particles reaching $x$ without suffering real collisions:

$$P(\Lambda < x) = 1 - P(\Lambda > x) = 1 - E(1_{\Lambda > x}) = 1 - E[w](x) = 1 - \exp \left( - \int_0^x \Sigma(x') dx' \right) \quad (3.3.11)$$
Theorem 2. Alg. 1 yields the correct expected number of real collisions in $\Delta x$ around $x$ if $\Delta x \to 0$:

$$\Sigma(x) \exp \left( - \int_0^x \Sigma(x') dx' \right) \Delta x.$$ (3.3.12)

Proof. This can be easily shown based on the proof of Theorem 1, as the number of real collisions in $\Delta x$ around $x$ can be formulated:

$$\frac{\Sigma(x)}{\Sigma_{samp}(x)q(x)} P(\Lambda < x + \Delta x, \Lambda > x) = \frac{\Sigma(x)}{\Sigma_{samp}(x)q(x)} P(\Lambda < x + \Delta x | \Lambda > x) P(\Lambda > x).$$ (3.3.13)

The first factor derives from the weighting scheme. It has already been proven, that the latter probability on the right hand side is:

$$P(\Lambda > x) = \exp \left( - \int_0^x \Sigma(x') dx' \right).$$ (3.3.14)

Next we use that in an inhomogeneous Poisson process with rate $\Sigma_{samp}(x)$, the probability of having exactly 1 event in $\Delta x$ around $x$ equals $\Sigma_{samp}(x) \Delta x$ as $\Delta x \to 0$. Thus

$$\lim_{\Delta x \to 0} P(\Lambda < x + \Delta x | \Lambda > x) = q(x) \Sigma_{samp}(x) \Delta x,$$ (3.3.15)

because tentative collision sites at $x$ are accepted as real collisions with rate $q(x)$. Substituting to Eq 3.3.13 the expected number of real collisions is indeed:

$$\Sigma(x) \exp \left( - \int_0^x \Sigma(x') dx' \right) \Delta x.$$ (3.3.16)

3.4 Previous work and relation to the exponential transform

In this section, several variants of the Woodcock method is presented, showing each its respective place in the BW framework, i.e. we point out, that each variant is a certain realization (specific choice of $q$ and $\Sigma_{samp}$) of Alg. 1. The idea is to introduce these previous efforts to better understand the optimization potential allowed by continuous parametrization in the BW framework. Showing the existence of a strong relation between the BW framework and another spatial variance reduction technique, exponential transform, completes this section.

3.4.1 Work of Spanier

In Ref. [59] Spanier introduces delta-tracking as a theoretical bridge between collision and track-length estimators. A straightforward tallying method is suggested to get scores from both true and virtual collisions.
He shows that if the sampling cross section ($\Sigma_{\text{ samp}}$) tends to infinity, a collision-type estimator (recording only at collision sites) tends to the track-length estimator (which records continuously along the particle track), when tallies are also allowed upon null-collisions. In this work, biasing the collision acceptance rate ($q$) is not considered, thus the original Woodcock tracking is applied (Alg. 2) with the idea of varying $\Sigma_m$.

3.4.2 Weighted Delta Tracking (WDT)

The WDT concept was first described by Ref. [60], and later entitled as Weighted Delta Tracking in Ref. [46]. The WDT method is constructed by replacing the rejection sampling of the original null-collision concept with a weighted scheme, thus turning null-collisions into relevant interactions regarding the contribution to the final score. In fact, null-collisions are useful to estimate total attenuation of particle flux, as the weight generation rules (factors $1 - \Sigma / \Sigma_{\text{ samp}}$) ensure that the expected particle weight is in line with exponential attenuation. i.e.

$$E[w_{\text{ uncoll.}}] = \exp \left( - \int_0^x \Sigma(t) dt \right)$$  \hspace{1cm} (3.4.1)

where $w_{\text{ uncoll.}}$ is the weight of a particle at $x$ starting from 0 that suffered no true collision. This is used in Ref. [46] in a way that a particle is forced to undergo both true and virtual(null-) collision at each tentative collision site, thus all particle surviving all collisions, and always having an uncollided part of the original starting particle (along with several possible secondaries produced). In our formalism, it means that the tracking algorithm uses both $q = 0$ and $q = 1$ at the same time, the former being a tool for transmittance estimation, and the latter producing true collisions and secondaries at every possible collision site. Our algorithm is an extension to this: it is no longer obligatory to force the particle to split to a collided and an uncollided part, but its probability can be arbitrary chosen by $q \in [0, 1]$. This should be more convenient to use in the presence of multiplying media, where the WDT would generate daughter products unnecessarily often. It is confirmed in Ref. [39], where a similar method is developed, called the Double Particle Model. Varying the sampling cross section is not considered in WDT, it is chosen to be the maximum of true cross sections: $\Sigma_{\text{ samp}} = \max\{\Sigma\}$.

3.4.3 Works of Galtier et al., Carter et al., and Cramer

These developments (Refs. [49], [48] and [45]) extend the applicability of the original Woodcock tracking method by using a non-majorant cross section for path length selection. The authors suggest the choice

$$q = \frac{\Sigma}{\Sigma + |\Sigma_{\text{ samp}} - \Sigma|}$$  \hspace{1cm} (3.4.2)

which returns the Original Woodcock algorithm when $\Sigma_{\text{ samp}} > \Sigma$. [45] also proves that minimum variance reached with this choice regarding the statistics of particle weights (but not in terms of contribution to the
final score). The method presented in Ref. [48] differs from other proposals on path length selection with non-constant sampling, meaning that instead of the pdf of Eq. 3.2.2 the pdf of

\[ f_C(s) = \Sigma_{samp}(\vec{r}) \exp(-\Sigma_{samp}(\vec{r})s) \] (3.4.3)

is used.

### 3.4.4 Residual Ratio Tracking (RRT)

The method was introduced in Ref. [45], and later given the name "Residual Ratio Tracking" by Novak et al. [47]. It corresponds to Alg. 1 with collision acceptance rate being set as \( q = 0 \) and \( \Sigma_{samp} \) chosen constant. RRT gives an unbiased estimate of total attenuation of the material.

### 3.4.5 Woodcock tracking in medical physics and computer graphics

Constant settings of parameters \( \Sigma_{samp} \) and \( q \) have already been investigated in a recent study [37], variance analysis and utilization for unbiased attenuation estimation was further developed in Ref. [39]. Two versions of the algorithm have been proposed, one that tracks both the collision daughter product(s) and the uncollided part (Double Particle Model-DPM), the other one tracks only one of them based on a random decision (Single Particle Model-SPM). The DPM is essentially the extension of the WDT method to arbitrary \( \Sigma_{samp} \). The SPM suggests the choice

\[ q = \frac{a|\Sigma|}{a|\Sigma| + |\Sigma_{samp} - \Sigma|} \] (3.4.4)

where \( a \) is the albedo of the material.

### 3.4.6 Relation to the exponential transform

Exponential transform, path stretching or also known as track-length biasing replaces the transition kernel with a biased kernel

\[ T_{ET}(\vec{r} \rightarrow \vec{r}') = (1 - p)\Sigma(\vec{r}') \exp \left(- \int_0^s \Sigma(\vec{r} + \vec{\Omega}t) dt \right) \] (3.4.5)

where \( p(\vec{\Omega}) \in (0, 1) \) is some function of movement direction. To account for the bias introduced in sampling distance to collision, the particle weight is multiplied by a factor of

\[ \frac{T}{T_{ET}} = \frac{\Sigma(\vec{r}') \exp \left(- \int_0^s \Sigma(\vec{r} + \vec{\Omega}t) dt \right)}{(1 - p)\Sigma(\vec{r}') \exp \left(- \int_0^s \Sigma(\vec{r} + \vec{\Omega}t) dt \right) + \exp \left(- p \int_0^s \Sigma(\vec{r} + \vec{\Omega}t) dt \right)} = \frac{\exp \left(-p \int_0^s \Sigma(\vec{r} + \vec{\Omega}t) dt \right)}{1 - p} \] (3.4.6)

after each transition.
Now let us show the similarities and differences between exponential transform and biased Woodcock tracking (Alg. 1). In case of the biased Woodcock (BW) scheme the probability density function of the distance to the first true collision is

\[ T_{BW}(\vec{r} \rightarrow \vec{r}') = q(\vec{r}') \Sigma_{samp}(\vec{r}') \exp \left( - \int_{0}^{s} q(\vec{r} + \vec{\Omega}t) \Sigma_{samp}(\vec{r} + \vec{\Omega}t) dt \right) \]  (3.4.7)

and the appropriate weight correction factor would be

\[ \frac{T}{T_{BW}} = \frac{\Sigma(\vec{r}') \exp \left( - \int_{0}^{s} \left( \Sigma(\vec{r} + \vec{\Omega}t) - q(\vec{r} + \vec{\Omega}t) \Sigma_{samp}(\vec{r} + \vec{\Omega}t) \right) dt \right)}{q(\vec{r}') \Sigma_{samp}(\vec{r}')} \]  (3.4.8)

However, the weight generation rules do not follow this practice in case of Woodcock tracking as it is built on a different idea. In biased Woodcock tracking, if \( K \) denotes the random variable describing the number of null-collisions generated by Alg. 1 before the first true collision, the correction factor of particle weight after the first (true) collision is

\[ \frac{\Sigma}{q \Sigma_{samp}} \left( \frac{1 - q \Sigma_{samp}}{1 - q} \right)^{K} \]  (3.4.9)

where, for the sake of brevity, we assumed homogeneous medium and constant sampling parameters. If Alg. 1 is supposed to yield unbiased estimates, the expected weight correction should be equal to the ratio of analog and nonanalog kernels (Eq. 3.4.8). This is indeed true, as the expectation of Eq. 3.4.9 is:

\[ E \left[ \frac{\Sigma}{q \Sigma_{samp}} \left( \frac{1 - q \Sigma_{samp}}{1 - q} \right)^{K} \right] = \frac{\Sigma}{q \Sigma_{samp}} \exp \left( - (\Sigma - q \Sigma_{samp})s \right) = \frac{T}{T_{BW}} \]  (3.4.10)

The quickest way to compute the expectation on the left hand side is by substituting into the probability generating function of \( K \) (the probability generating function of a random variable \( K \) is defined by \( G(z) = E \left[ z^{K} \right] \)). While \( K \) is Poisson distributed on \([0, s]\) with mean \((1 - q)\Sigma_{samp}\), the probability generating function reads as \( G(z) = e^{(1 - q)\Sigma_{samp}(z - 1)} \), hence the first equality holds. There is a limiting case when the two weighting strategies (Eq. 3.4.8 and Eq. 3.4.9) become identical not only in expectation, but also in distribution. Take the limits \( \Sigma_{samp} \rightarrow \infty \) and \( q \rightarrow 0 \) while \( q \Sigma_{samp} \rightarrow c \), where \( c \) is some positive constant. The resulting algorithm generates collisions with infinite frequency while the acceptance rate tends to zero. If this kind of algorithm was feasible, it would introduce the same weight distribution as weighting with the ratio of transition kernels.

So far we have shown that although the weight generation rules in the biased Woodcock scheme do not follow the practice of weighting with the ratio of analog and nonanalog kernels (as in the exponentially transformed simulation), a limiting case can be found when it is yet true. This should mean that there is a theoretical bridge between biased Woodcock and exponential transform. Recall the two probability densities Eq. 3.4.5 and 3.4.7, and let \( \Sigma_{samp} \) and \( q \) be such that

\[ q(\vec{r}, E, \vec{\Omega}) \Sigma_{samp}(\vec{r}, E, \vec{\Omega}) = (1 - p(\vec{\Omega})) \Sigma(\vec{r}, E) \]  (3.4.11)
holds. Then, the probability densities of distance to collision are the same, only the weighting strategy differs. But the previous reasoning implies that biased Woodcock tracking and exponential transform become completely identical (i.e. they describe the same nonanalog simulation) in the limit $\Sigma_{samp} \to \infty$ and $q \to 0$ while $q\Sigma_{samp} \to (1 - p)\Sigma$. The significance of this result lies in the optimization possibilities of biased Woodcock tracking. As the optimal setting of exponential transform has already been investigated in many studies (e.g. Refs [61] and [62]), previous results may also help the optimization of BW. This will later be discussed in Section 3.6.

### 3.5 Error estimation and computational cost

In order to determine which variant of the Woodcock method is superior to the other and make a priori optimization possible in general, estimation of statistical error and computational cost is required. The computational cost associated with null-collision algorithms lies in the effort to generate one true collision site. Thus, one can be particularly interested in the number of collisions simulated until a collision site is accepted as a true collision. It is also true for GUARDYAN, significant part of the overall computation time is consumed by looking up cross sections at a tentative collision site. By estimating the number of instances, when this cross section look-up is required, we essentially estimate the runtime of GUARDYAN. Let $C$ denote the number of collisions until the first collision accepted as true. The computational cost is then estimated by the expectation of $C$:

$$E[C] = \int_0^{\infty} \left( \int_0^{x'} \Sigma_{samp}(y)dy \right) q(x')\Sigma_{samp}(x')exp \left( - \int_0^{x'} q(y)\Sigma_{samp}(y)dy \right) dx'$$  \hspace{1cm} (3.5.1)$$

which gives $q^{-1}$ when $q$ is constant. The constant case is trivial, because the problem simplifies to the question of the expectation of the number of Bernoulli trials needed to get one success. Eq. 3.5.1 is basically the expected value of the number of collisions according to the distribution of the first true collision (Eq. 3.4.7). Now assume, that the domain of simulation is some finite volume, and let the current position of the particle be the origin. Suppose, that the distance to escape the volume in the direction of movement is $x$. It may happen, that the particle will leave the volume without making any true collision, thus the right measure of computational cost would be the number of all collisions leading up to the leakage. The formula of Eq. 3.5.1 needs to be modified accordingly:

$$E[C_{[0,x]}] = \int_0^{x} \left( \int_0^{x'} \Sigma_{samp}(y)dy \right) q(x')\Sigma_{samp}(x')exp \left( - \int_0^{x'} q(y)\Sigma_{samp}(y)dy \right) dx' + \int_0^{x} \Sigma_{samp}(x')dx' \exp \left( - \int_0^{x} q(x')\Sigma_{samp}(x')dx' \right)$$  \hspace{1cm} (3.5.2)$$

$$+ \int_0^{x} \Sigma_{samp}(x')dx' \exp \left( - \int_0^{x} q(x')\Sigma_{samp}(x')dx' \right)$$  \hspace{1cm} (3.5.3)$$
**Theorem 3.** The general form of Eq. 3.5.3 can be reduced to

\[
E[C_{[0,x]}] = \frac{1}{q} \left( 1 - e^{\exp\left(-q \int_0^x \Sigma_{\text{samp}}(x') \, dx'\right)} \right)
\]  

(3.5.4)

if \( q \) is chosen constant on \([0, x] \).

**Proof.** Let us define the following functions:

\[
f(x) = \int_0^x \Sigma_{\text{samp}}(x') \, dx'
\]  

(3.5.5)

\[
g(x) = -e^{\exp\left(-q \int_0^x \Sigma_{\text{samp}}(x') \, dx'\right)}.
\]  

(3.5.6)

If \( q \) is constant, the above equation can be rewritten with these definitions as follows:

\[
E[C_{[0,x]}] = \int_0^x f(x') \frac{dg(x')}{dx'} \, dx' - f(x)g(x).
\]  

(3.5.7)

The first term is calculated via integration by parts:

\[
E[C_{[0,x]}] = \left[ f(x')g(x') \right]_0^x - \int_0^x \frac{df(x')}{dx'} \, g(x') \, dx' - f(x)g(x).
\]  

(3.5.8)

As \( f(0) = 0 \) and \( g(0) = 1 \), the first and third term cancel out:

\[
E[C_{[0,x]}] = -\int_0^x \frac{df(x')}{dx'} \, g(x') \, dx'.
\]  

(3.5.9)

Now we use that \( g(x) = -e^{\exp(-qf(x))} \):

\[
E[C_{[0,x]}] = \int_0^x \frac{df(x')}{dx'} \, e^{\exp(-qf(x))} \, dx'.
\]  

(3.5.10)

Evaluating the integral and simple substitution yield

\[
E[C_{[0,x]}] = \left[ -\frac{1}{q} e^{\exp(-qf(x'))} \right]_0^x = \frac{1}{q} \left( 1 - e^{\exp\left(-q \int_0^x \Sigma_{\text{samp}}(x') \, dx'\right)} \right).
\]  

(3.5.11)

Next, we seek to find some mathematical formalism to describe how variance reduction may be achieved via the biased Woodcock tracking framework. We start from the moment equations, that attempts to obtain equations for the expectation of some function of the score. In particular, the expectation and the second moment of the score may be of interest as it governs the error (variance) of the estimation. Following up on the work of Lux and Koblinger [4], an extension to null-collision techniques can be developed. There are two things to consider here: the appropriate transition and collision kernels corresponding to Alg. 1, and the weight generation rules (how particle weight is adjusted to account for the biased kernels). The biased Woodcock tracking is equal to a nonanalog simulation with transition kernel

\[
\hat{T}_\delta(P, P') = \Sigma_{\text{samp}}(P')e^{\exp\left(-\int_P^{P'} \Sigma_{\text{samp}}(\vec{r} + \vec{\Omega}t) \, dt\right)}
\]  

(3.5.12)
and collision kernel
\[
\hat{C}_\delta(P', P'') = q(P')\hat{C}(P', P'') + (1 - q(P'))\delta(P' - P'')
\] (3.5.13)

where \( \hat{C}(P', P'') \) can be an already biased collision kernel. Unlike in many variance reduction techniques this bias in these kernels is not compensated by multiplying particle weight by \( T/\hat{T}_\delta \) and \( C/\hat{C}_\delta \) (therefore it is not like an importance sampling strategy [49]), but the weight generation rules are:

\[
w'_\text{true} = \frac{w}{q^{\sum_{\text{samp}}}} \] (3.5.14)

if a true collision happens, and

\[
w'_\text{null} = w \frac{1 - \sum_{\text{samp}}}{1 - q} \] (3.5.15)

when a null-collision occurs. After a true collision, post-collision weights are calculated by:

\[
w'' = w'_\text{true} \frac{C(P', P'')}{\hat{C}(P', P'')} \] (3.5.16)

Let us use the same notations as in Ref. [4], let \( M_i(P) \) denote the \( i-th \) moment of the score due to a starter at \( P \), \( \hat{c}_a \) the probability of absorption, \( \hat{f}, \hat{f}_a \) and \( \hat{f}_s \) the pay-off functions of free flight, absorption and scatter respectively.

**Theorem 4.** Based on Eq. (5.58) in Ref. [4], the second moment of the score in the biased Woodcock framework reads as

\[
w^2 M_2(P) = \int \hat{T}_\delta(P, P') \hat{c}_\delta(P') \left[ w'_\text{true} \hat{f}(P, P') + w_a \hat{f}_a(P') \right]^2 dP' + \\
+ \sum_{r=0}^{2} \left[ \frac{2}{r} \right] \int \hat{T}_\delta(P, P') \int q(P') \hat{C}(P', P'') \left[ w'_\text{true} \hat{f}(P, P') + w'' \hat{f}_s(P', P'') \right]^{2-r} (w'')^r M_r(P'') dP'' dP' + \\
+ \sum_{r=0}^{2} \left[ \frac{2}{r} \right] \int \hat{T}_\delta(P, P')(1 - q(P')) \left[ w'_\text{null} \hat{f}(P, P') \right]^{2-r} (w'_\text{null})^r M_r(P') dP'
\] (3.5.17)

### 3.6 Optimization strategies

In this section we use the above described theory to scout optimization strategies in the BW framework. We also show that none of the previously suggested null-collision based tracking methods are optimally set in our framework. Demonstration will be given in cases when analytic solution is available, thus simulation results shall also be confirmed by variance calculation based on the moment equations and some properties of the inhomogeneous Poisson process. Three particle transport problems will be investigated:

a) Attenuation estimation along a ray in a medium with continuously varying cross section. The domain of simulation is assumed to be surrounded by a black absorber.
b) Transmission through a slab in the straight-ahead scattering model. Particles are born at the left side of the slab moving in the positive direction. The probability for straight ahead scatter is $c$, and with probability $1 - c$ the particle is absorbed. Particle history is terminated upon absorption or leakage. Contribution is assumed to be proportional to the weight of particles leaving the slab.

c) Transmission through a slab in the Fermi scattering model. This model is similar to problem b), with the distinction that particles can scatter backward as well as forward with equal probability.

3.6.1 Problem a) - a numerical optimization study

To find the solution to problem a) with most methods (be it either deterministic or stochastic) one must integrate the true cross sections all along the ray. If the integral has a closed form, there is no need for any simulation, the transmittance is given according to exponential attenuation. However, there are problems with continuously varying cross section, when the closed form is not available, then, only biased results can be achieved via numerical integration. The bias may of course be bounded and reduced arbitrary small by choosing a fine enough mesh, but computational cost will rise exponentially. Here, our goal is to find an unbiased, efficient algorithm to estimate attenuation from a few samples of the true cross sections. We use an efficiency measure following the conventional definition [63], often referred to as the Figure of Merit (FoM):

$$FoM = \frac{1}{r^2 T}$$

(3.6.1)

where $r$ stands for the relative error of the estimation (which is the ratio of standard deviation of contributions and the expected contribution), and $T$ stands for simulation time. Optimal settings will be considered as specific choices of $\Sigma_{samp}$ and $q$, such that the resulting algorithm achieves the smallest variance in a given time, or it finishes the calculation fastest with a fixed variance, i.e. it has the greatest FoM. $T$ will be assumed proportional to the number of collisions per particle. The relative error of the estimated attenuation at position $x$ on the ray is given by Eq. 3.6.2. The expected number of collisions simulated before reaching $x$ has already been discussed, recall Eq. 3.5.3. The equation for the relative error is confirmed in Ref. [39] (Single Particle Model), via a differential equation approach. Another derivation is given here.

Theorem 5.

$$r^2 = exp \left\{ \int_0^x \frac{1}{1 - q(x')} \left( q(x') \left( \Sigma_{samp}(x') - 2\Sigma(x') \right) + \frac{\Sigma^2(x')}{\Sigma_{samp}(x')} \right) dx' \right\} - 1$$

(3.6.2)

Proof. The squared relative error of transmittance is a function of moments of statistical weight $w$, which is
proved by:

\[
\begin{align*}
    r^2 &= \frac{1}{N} \sum_{n=1}^{N} w(n) \\
    &\left( E \left[ \frac{1}{N} \sum_{n=1}^{N} w(n) \right] \right)^2 \\
    &= \frac{1}{N^2} \sum_{n=1}^{N} Var \left[ w(n) \right] \\
    &= \frac{N \cdot Var[w]}{(N \cdot E[w])^2} \\
    \text{(3.6.3)}
\end{align*}
\]

In the last two step, we used that different histories are independent and weights are identically distributed random variables, thus we can simplify our notation by introducing the variance and expectation of any \( w(n) \) by omitting the history index: \( Var \left[ w(n) \right] = Var[w] \) and \( E \left[ w(n) \right] = E[w] \). Expanding the above formula, the squared relative error can be written as a function of the first and second moment of \( w \):

\[
\begin{align*}
    r^2 &= \frac{1}{N} \frac{Var[w]}{(E[w])^2} \\
    &= \frac{1}{N} \frac{E[w^2] - (E[w])^2}{(E[w])^2} \\
    &= \frac{1}{N} \left( \frac{E[w^2]}{(E[w])^2} - 1 \right) \\
    \text{(3.6.4)}
\end{align*}
\]

If we want to determine the error of our estimate, we must also calculate the second moment of transmitted particle weights. The derivation of the second moment is done the same way as the first moment’s, \( w \) is simply replaced by \( w^2 \) in the beginning. The first major difference will show when rewriting Eq. 3.3.6:

\[
\begin{align*}
    E \left[ w^2 \right] (x) &= \sum_{k=0}^{\infty} \int \cdots \int \prod_{i=1}^{k} \left( \frac{1 - \frac{\Sigma(x_i)}{\Sigma(x_i)}}{1 - q(x_i)} \right)^2 \prod_{i=1}^{k} (1 - q(x_i)) p_{X|K}(x_1, \ldots, x_k) dx_1 \cdots dx_k \\
    p_K(k) &= \\
    &= \sum_{k=0}^{\infty} \int \cdots \int \prod_{i=1}^{k} \left( \frac{1 - \frac{\Sigma(x_i)}{\Sigma(x_i)}}{1 - q(x_i)} \right)^2 p_{X|K}(x_1, \ldots, x_k) dx_1 \cdots dx_k \\
    &= \sum_{k=0}^{\infty} \int \cdots \int \prod_{i=1}^{k} p_{X|K}(x_1, \ldots, x_k) dx_1 \cdots dx_k \\
    \text{(3.6.5)}
\end{align*}
\]

where factors \( 1 - q(x_i) \) do not cancel out. In the following steps, the integrals are calculated based on the properties of the inhomogeneous Poisson process the same way as in Section 3.3. Some new terms appear after the expansion of the square, but the reasoning is the same: we use that \( K \) is Poisson distributed with mean \( \int_{0}^{x} \Sigma_{samp}(x') dx' \), and probability generating function of Eq 3.3.9 appears again.

\[
\begin{align*}
    E \left[ w^2 \right] (x) &= \sum_{k=0}^{\infty} \int_{0}^{x} \left( 1 - \frac{\Sigma(u)}{\Sigma_{samp}(u)} \right)^2 \frac{\Sigma_{samp}(u)}{\int_{0}^{x} \Sigma_{samp}(x') dx'} du \\
    &= G \left( \frac{1}{\int_{0}^{x} \Sigma_{samp}(x') dx'} \int_{0}^{x} \Sigma_{samp}(x') dx' - 2 \Sigma(x') + \frac{\Sigma_{samp}(x')}{\Sigma_{samp}(x')} dx' \right) \\
    \text{(3.6.6)}
\end{align*}
\]

By definition of the PGF:

\[
E \left[ w^2 \right] (x) = \exp \left\{ \int_{0}^{x} \frac{\Sigma_{samp}(x') - 2 \Sigma(x') + \frac{\Sigma_{samp}(x')}{\Sigma_{samp}(x')}}{1 - q(x')} dx' - \int_{0}^{x} \Sigma_{samp}(x') dx' \right\}
\]
\[ \exp \left\{ \int_0^x \frac{q(x') \Sigma_{\text{samp}}(x') - 2 \Sigma(x') + \frac{\Sigma^2(x')}{\Sigma_{\text{samp}}(x')}}{1 - q(x')} \, dx' \right\} \] (3.6.7)

For the squared relative error we get:

\[
 r^2 = \frac{1}{N} \left( E\left[ w^2 \right] - 1 \right) = \frac{1}{N} \left( \exp \left\{ \int_0^x \frac{1}{1 - q(x')} \left( q(x') \Sigma_{\text{samp}}(x') - 2 \Sigma(x') + \frac{\Sigma^2(x')}{\Sigma_{\text{samp}}(x')} \right) \, dx' \right\} - 1 \right) = \\
= \frac{1}{N} \left( \exp \left\{ \int_0^x \frac{1}{1 - q(x')} \left( q(x') \left( \Sigma_{\text{samp}}(x') - 2 \Sigma(x') \right) + \frac{\Sigma^2(x')}{\Sigma_{\text{samp}}(x')} \right) \, dx' \right\} - 1 \right) \] (3.6.8)

Efficiency based optimization requires finding extrema of the Figure of Merit, which can only be done numerically. The Figure of Merit can be calculated as a function of \( \Sigma_{\text{samp}} \) and \( q \):

\[
 FoM(q, \Sigma_{\text{samp}}) = \\
\left\{ \frac{1}{N} \left( \exp \left[ \frac{1}{1 - q} \int_0^x q(x') \left( \Sigma_{\text{samp}} - 2 \Sigma(x') \right) + \frac{\Sigma^2(x')}{\Sigma_{\text{samp}}} \, dx' \right] - 1 \right) \cdot \frac{1 - q}{q} \left( 1 - \exp \left( -q \Sigma_{\text{samp}} x \right) \right) \right\}^{-1} \] (3.6.9)

Unfortunately, global extrema of FoM can not be expressed analytically even if \( \Sigma_{\text{samp}} \) and \( q \) were considered constant, since \( \frac{\partial FoM}{\partial q} = 0 \) and \( \frac{\partial FoM}{\partial \Sigma_{\text{ samp}}} = 0 \) lead to transcendent equations. To illustrate that optimal parameters can indeed be found if parameters are assumed to be independent of spatial coordinates, Fig. 3.1 shows how the efficiency of the calculation depend on the choice of parameters \( \Sigma_{\text{ samp}} \) and \( q \).

Fig. 3.1 also shows, that with larger \( q \) than optimal, FoM quickly drops in the region around the optimal \( \Sigma_{\text{ samp}} \), FoM value is also very sensitive of altering \( \Sigma_{\text{ samp}} \) in the vicinity of the optimum. As a consequence, \( q \) must be chosen with caution, in real applications we suggest it to be an underestimation of the suspected optimal value. Then, some uncertainty of the initial information can not cause too much trouble, as our guess for the optimum is farther from the region of large gradients.

Optimal settings of sampling parameters will depend on the true cross sections \( \Sigma \) and position \( x \). In problem a) let the total cross section be

\[
 \Sigma(y) = 0.1 + 0.05 \left( \cos(2y) + 1 \right) e^{-\frac{y}{10}} \] (3.6.10)

and suppose \( x = 10cm \). Restricting the optimization to constant parameters, the optimal settings are \( \Sigma_{\text{ samp}} = 0.14cm^{-1} \) and \( q = 0.946 \). Further details on how to find optimal settings can be found in a
Optimization of the Biased Woodcock algorithm can be carried out numerically, the maximum of FoM found at $\Sigma_{samp} = 0.14 \text{ cm}^{-1}$ and $q = 0.946$.

Figure 3.1: Figure of Merit as a function of parameters $\Sigma_{samp}$ and $q$ assuming cross sections of Eq. 3.6.10.

recent paper [38]. Note, that the optimal sampling frequency is below the maximum of cross sections ($\text{max}\{\Sigma\} = 0.2$), allowing the sign change of particle weights. This causes an increase in variance, but the save in computation time compensates for it, resulting in higher overall efficiency. Table 3.1 shows statistics of the estimated transmission values by four different methods. Also note, that the previously suggested parameters are indeed not optimal.
Table 3.1: Comparison of Woodcock-type algorithms for attenuation estimation on a ray at distance $x = 10 cm$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$r^2$</th>
<th>$T$</th>
<th>FoM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Woodcock (Ref. [32])</td>
<td>$2.770 \cdot 10^{-5}$</td>
<td>34513</td>
<td>1.046</td>
</tr>
<tr>
<td>$q = \Sigma (x)/0.2 cm^{-1}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Sigma_{samp} = 0.2 cm^{-1}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDT/RRT/DPM (Refs. [46],</td>
<td>$1.479 \cdot 10^{-5}$</td>
<td>199841</td>
<td>0.338</td>
</tr>
<tr>
<td>[47] and [39])</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q = 0$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Sigma_{samp} = 0.2 cm^{-1}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refs. [49] and [48]</td>
<td>$3.376 \cdot 10^{-5}$</td>
<td>17687</td>
<td>1.675</td>
</tr>
<tr>
<td>$q = \Sigma (x)/(2\Sigma (x) - 0.14 cm^{-1})$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Sigma_{samp} = 0.14 cm^{-1}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Biased Woodcock tracking</td>
<td>$7.586 \cdot 10^{-5}$</td>
<td>4261</td>
<td>3.094</td>
</tr>
<tr>
<td>$q = 0.946$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Sigma_{samp} = 0.14 cm^{-1}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulation results for relative error, the expected number of collisions and Figure of Merit as a function of distance along the ray is given in Fig. 3.2. Although the optimal parameters were set for $x = 10 cm$, Fig. 3.2 shows that the biased Woodcock tracking performs very well all along the ray, producing a reliable estimate of attenuation with very few samples from the true cross sections.

Residual ratio tracking [47], the Double Particle Model described in [39] and Weighted Delta Tracking [46] corresponds to using $q = 0$ in problem a). This choice proves to be the one producing the lowest variance per particle history. This is because the choice $q = 0$ yields a survival biased simulation, where all particles survive and make non-zero contribution to the final score. Naturally, the survival biased simulation takes more time to finish, therefore FoM values are smaller. However we found that by increasing the optical thickness this strategy will eventually become optimal also in terms of efficiency, therefore we have put some effort into developing the sampling process even further. Particularly, non-constant setting of $\Sigma_{samp}$ was of interest. When $q = 0$, the squared relative error and the expected number of collisions can be written as:

$$r^2(x) = \frac{1}{N} \left( \exp \left\{ \int_0^x \frac{\Sigma^2(x')}{\Sigma_{samp}(x')} dx' \right\} - 1 \right)$$  \hspace{1cm} (3.6.11)
Lowest error is reached by simulating only one particle while increasing the sampling frequency ($\Sigma_{samp}$) as much as possible. This is because the squared relative error is inversely proportional to $N$ (the number of particles), but it scales with $\exp \left( \frac{1}{\Sigma_{samp}} \right)$ with respect to the sampling frequency. Both factors ($N$ and $\Sigma_{samp}$) affect the expected number of collisions in the same way, so by simulating half as many particles with double sampling frequency always gives a lower error. Now let us find the optimal distribution of null-collision sites, i.e. the shape of the function $\Sigma_{samp}(x)$. Consider the optimal strategy which achieves the lowest variance in a given time, i.e. the number of collisions is fixed. Finding this optimal sampling strategy means to find a function $\Sigma_{samp} \in L_2$ which minimizes Eq. 3.6.11 with a constraint of Eq. 3.6.12. This is a basic variational problem with an integral constraint and can be solved with Lagrange multiplier.

**Theorem 6.** The optimal strategy in a given time uses $\Sigma_{samp}$ proportional to $\Sigma$.

**Proof.** Let us write $\Sigma_{samp}(x) = a(x)\Sigma(x)$. We will prove that the optimal sampling frequency producing the same number of collisions $C = \int_0^x \Sigma_{samp}(t)dt$ is proportional to $\Sigma(x)$, i.e. $a(x)$ is a constant function. In order to minimize the relative error, we have to find a minima of $\int_0^x \frac{\Sigma(x')}{\Sigma_{samp}(x')}dx' = \int_0^x \frac{\Sigma(x')}{a(x')}dx'$ according to $a(x)$. The Lagrangian of the variational problem considering the constraint is

$$\int_0^x \frac{\Sigma(x')}{a(x')} + ma(x')\Sigma(x')dx' - mC$$

(3.6.13)
where $m$ stands for the Lagrange multiplier. The Euler-Lagrange equation reads as

$$-rac{\Sigma(x')}{a^2(x')} dx' + m \Sigma(x') = 0$$

(3.6.14)

and it gives

$$a(x') = \frac{1}{\sqrt{m}}$$

(3.6.15)

thus $a$ is indeed constant and $\Sigma_{samp}$ is proportional to $\Sigma$.

An interesting observation is that by increasing the time available to do the calculations, the best strategy is to increase $a$ too, and eventually, with infinite amount of time, the optimal sampling frequency tends to infinity, conserving the original shape of $\Sigma$ meanwhile. This is also confirmed by [45], in the constant $\Sigma_{samp}$ case.

### 3.6.2 Problem b) - BW and exponential transform in the straight-ahead scattering model

In problem b) let us first suppose that the slab is embedded in an infinitely strong black absorber. This setup was already used in Ref. [64], and variance analysis was also provided regarding an exponentially transformed simulation. The second moment of particle weight contributing to the score was derived as:

$$S_{ET}^2 = \exp\left(-\frac{1-p^2 - \frac{c^2}{c^*} \Sigma x}{1-p \Sigma x}\right)$$

(3.6.16)

where $\Sigma$ is the total cross section and $x$ is the thickness of the slab. $p$ is the path stretching parameter defined in Sec. 3.4 and $c^*$ is the scattering probability in the transformed simulation. In the case of the biased Woodcock tracking, the second moment can also be calculated based on Theorem 4.

$$S_{BW}^2 = \exp\left(-\frac{2q\Sigma_{samp} - \Sigma_{samp}^2 q^2 - \Sigma^2 q - \Sigma^2 \frac{c^2}{(c^*)^2} (1-q)}{\Sigma_{samp} q (1-q)} x\right)$$

(3.6.17)

In both tracking strategy, choosing $c^* = 1$ (survival biasing) is obviously the best choice in terms of variance reduction. Choosing $p = 1-c$ yields a zero variance scheme for the exponential transform, i.e. all contributions are the same independent of the particle history. The same is true for the biased Woodcock algorithm if we consider the limiting case $\Sigma_{samp} \to \infty$ and $q \to 0$ while $q\Sigma_{samp} \to c\Sigma$. According to Sec. 3.4, the two strategies are then equivalent, and the variance indeed converges to zero as the second moment converges to the square of the mean:

$$S_{BW}^2 \to \exp\left(-\frac{2c\Sigma x - (c\Sigma)^2 - \Sigma^2 c^2}{c\Sigma} x\right) = \exp\left(-2(1-c)\Sigma x\right)$$

(3.6.18)
Of course, sampling with $\Sigma_{samp} \to \infty$ is not feasible in practice, thus the null-collision strategy always produces higher variance then the optimal exponential transformed simulation. However, if the same problem is considered in a geometry where the slab is a part of a semi-infinite homogeneous medium, i.e. black absorber surroundings are not assumed, the null-collision strategy still tends to a zero variance solution, whereas the exponential transformed simulation does not. The solution of the problem will obviously be the same as before, but second moments will be different,

$$S^2_{ET} = \frac{1}{1-p^2} \exp \left( -\frac{1-p^2 - c^2}{c^2 \Sigma} \Sigma x \right)$$  \hspace{1cm} (3.6.19)

in the case of exponential transform (Ref. [4]), and

$$S^2_{BW} = \left( \frac{\Sigma_{samp}^2}{\Sigma_{samp}} \right)^2 \frac{2q \Sigma_{samp} + q}{q(1-q)} \exp \left( -\frac{2q \Sigma_{samp} - \Sigma^2_{samp} q^2 - \Sigma^2 q - \Sigma^2 c^2 (1-q)}{\Sigma_{samp} q (1-q)} x \right)$$  \hspace{1cm} (3.6.20)

regarding the biased Woodcock strategy. Instead of minimizing the second moment of Eq. 3.6.20, which would give optimal settings depending on the slab thickness $x$, we will consider the optimal collision acceptance rate $q_{opt}$ as the parameter minimizing the variance in the problem with black absorber surroundings:

$$q_{opt} = \frac{c^2 - c \left| 1 - \frac{\Sigma}{\Sigma_{samp}} \right|}{c^2 - \left( 1 - \frac{\Sigma}{\Sigma_{samp}} \right)^2}$$  \hspace{1cm} (3.6.21)

Notice again, all other previously suggested null-collision techniques (all other possible choices of $q$) fail to produce optimal variance. Also, Fig 3.3 shows that optimally set exponential transform becomes inferior regarding estimation error in comparison with the biased Woodcock tracking, if the sampling frequency is chosen sufficiently high.

### 3.6.3 Problem c) BW and exponential transform in the Fermi-scattering model

For the backward-forward scattering problem Ref. [61] offers a good reference, providing optimal path stretching parameters for exponential transform. Here, we present simulation results comparing variances of the optimally set exponential transformed simulation and biased Woodcock tracking. As in Ref. [61], survival biasing will be implicitly assumed. Instead of the cumbersome optimization of the biased Woodcock algorithm regarding parameter $q$, another straightforward choice will be introduced:

$$q = \frac{(1-p \mu) \Sigma}{\Sigma_{samp}}$$  \hspace{1cm} (3.6.22)

where $\mu$ is the cosine of the direction of movement (either $-1$ or $1$ in our problem). With this choice, the distribution of true collisions is the same in the two methods, the difference lies in the weighting strategy. We show our results as a function of sampling frequency $\Sigma_{samp}$, relative to the exponential transformed
simulation with smallest variance. We investigate two cases, both with optical thickness $\Sigma x = 20$, and one with scattering probability $c = 0.9$, the other with $c = 0.5$. Optimal settings of the exponentially transformed simulation can be found in Ref. [61].

Fig. 3.4 shows estimation uncertainty as a function of sampling cross section in the biased Woodcock tracking framework. The best variance obtainable with the exponential transform is also shown. The biased Woodcock algorithm produces variances comparable to the exponential transform in the region where the sampling cross section is above $10 \text{cm}^{-1}$, even going below the best variance obtainable if the sampling frequency is sufficiently high.

### 3.7 Practical considerations and the majorant mesh

We saw, that in the case of attenuation estimation (problem a)), specific suggestions were made regarding the optimal choice of parameters. Based on Eq. 3.6.2 and Eq. 3.5.3, numerical optimization is possible, and can be used to obtain optimal parameters. The same was even true for a straight-ahead scattering model assuming a homogeneous slab setup. However, in more complex problems optimal settings are harder to find. One way to go is the tuning of parameters $\Sigma_{\text{samp}}$ and $q$ until the result is acceptable. This strategy may be supported by Theorem 4 and Eq. 3.5.3 (estimating the variance and computational cost), if any numerical solver can be utilized to solve these equations. Another way to go is to reduce the degrees of freedom by fixing $q\Sigma_{\text{samp}}$ according to Eq. 3.4.11, this guarantees that the distribution of true collision sites are the same as in an exponentially transformed simulation. The two methods become identical if the sampling
cross section tends to infinity, according to Sec. 3.4. One could take an optimal setting of path stretching parameter for the problem, and increase the sampling frequency (while Eq. 3.4.11 still holds) until variances become acceptable.

In GUARDYAN, exploiting the full potential of the framework seems even more difficult, the theory should be completed to include multiplying reactions and time-dependent tallies. Variance reduction in GUARDYAN should keep a nice population at all times while providing precise estimates as well, and Theorem 4 considers only the latter. Other approaches are currently under development, e.g. using an adjoint function to generate $\Sigma_{samp}$ and $q$ with spatial and spectral dependence. At the time of writing this thesis, the BW framework is only used as means to reduce computational cost, i.e. $\Sigma_{samp}$ is chosen to be stepwise function in space and energy, while $q$ is simply

$$q = \frac{\Sigma}{\Sigma_{samp}},$$

(3.7.1)

as in the original Woodcock method. $\Sigma_{samp}$ is generated the following way. A cartesian mesh is superposed on the system, partitioning the geometry to super-voxels. The local majorant is determined for the unionized energy grid, by sampling each super-voxel uniformly in space and choosing the greatest cross section present in that particular voxel for every energy on the grid. This procedure basically constructs a majorant mesh. Now, the localized heavy absorber problem (cf. Sec. 3.2) affects only the voxel containing the absorber. While $\Sigma_{samp}$ is piece-wise constant, sampling Eq. 3.2.2 is extremely simple. By the inverse cumulative
method the equation to be solved reads as

\[ -ln(\xi) = \sum_{i=1}^{n} \Sigma_{samp} s^{(i)} \]  \hspace{1cm} (3.7.2)

where \( \Sigma_{samp}^{(i)} \) and \( s^{(i)} \) are the majorant cross section and traversed path length in the \( i \)–th visited voxel, and \( \xi \) is a canonical random number. \( s^{(i)} \) is calculated by the 3D-DDA algorithm [65] in GUARDYAN, as a fast way of determining ray-box intersections.
Chapter 4

Variance reduction and acceleration methods

Despite exponentially increasing power of modern HPC platforms application of MC reactor kinetics is still limited by its inherent computational burden. Runtime of dynamic MC codes are thus of particular interest. Also, statistical uncertainty of MC estimates is equally important, in most applications, MC variance and runtime are inversely proportional. That is, if a calculation finishes under time $T$ with variance $\sigma^2$, $\sigma^2/2$ variance can be obtained by simply doubling the time available, $T$, e.g. simulating two times as many particle histories. Performance analysis of MC code takes both factors into account, the traditional efficiency measure, FoM (Figure of Merit), is defined by

$$FoM = \frac{1}{\sigma^2 T}. \quad (4.0.1)$$

This chapter summarizes our attempts to improve the performance of GUARDYAN. Part of these efforts include clever algorithms that target higher precision of MC estimation. These are called variance reduction techniques. The concept of variance reduction is to modify the random walk of particles to take more samples from part of the phase space we are interested in, while leaving the expected values intact and altering the variance. For example, a particle escaping the system shortly after it was born makes a poor sample, while thermal neutrons near fuel rods are highly valued samples with respect to e.g. reactor power. One must note that variance reduction is usually achieved at the expense of computation time. True variance reduction gains more by decreasing estimation error than increasing computation time, in other words, it improves the Figure of Merit. In fact, one can come up with such constructions that increase variance, but give computations a huge boost. Overall FoM can still be larger. These strange concepts can also be considered
as variance reduction, since the implementation with the greater FoM can simulate more neutron histories under the same time, eventually achieving lower variance than the one with lower FoM. Beside variance reduction methods, performance gain can also be obtained by accelerating the MC calculations. These algorithms make up the other part of this chapter. Acceleration methods increase FoM by reducing runtime while leaving the variance intact. Thus they are not variance reduction methods, although estimation error can be improved given the same simulation time, along the lines of the above logic.

With infinite computer resources, one would be able to simulate time-dependent neutron transport in an analog way, i.e. use the same probability distributions as described by physical laws to generate random walks. In Chapter 2, we derived some algorithms that make dynamic MC feasible even on a single GPU, having respectable, but by far not infinite, computer power. These concepts essentially describe how to make non-analog calculations, i.e. use biased probability distributions to generate random walks and compensate by weighting the particles. Without such variance reduction tools (biased fission probability, the branchless neutron history method, the combing and the forced decay of precursors) GUARDYAN calculations would not converge, or not at least within reasonable time, thus we consider them fundamental variance reduction techniques. The main focus of this chapter will be on other performance improving methods, that are not mandatory to apply. Targeting performance enhancement of GUARDYAN, special attention was devoted to distance-to-collision sampling. A spatial variance reduction framework was designed to better exploit GPU computing power, and was already described in Chapter 3. The practical implementation of spatial variance reduction in GUARDYAN, called the majorant mesh method, can only reduce computational burden however, and not the variance. Therefore it is considered as an acceleration method, along with the improved point-in-cell search algorithm. These algorithms target the reduction of computational cost associated with determining the material composition at a particle location. Location of a particle in GUARDYAN is searched in a universe-based combinatorial solid geometry structure, standard in other MC codes like MCNP or Serpent. The algorithm (which is usually termed point-in-cell search) decides which cell the particle is in by analyzing a tree structure of nested universes, lattices and surfaces. In our previous study [66], we identified this routine being responsible for the largest part of computational need. The improved point-in-cell search accelerates the execution of this routine, while the majorant mesh method reduces the frequency of calling the routine. Yet as another acceleration technique, the event-based version of GUARDYAN was implemented, in which the main idea is to rearrange tasks for threads to ensure a more even distribution of workload. As for true variance reduction, we implemented an improved version of the combing method by using the neutron importance as a weighting function.
4.1 Importance-based population control

In Chapter 2 we introduced the combing method as a fundamental population control tool for GUARDYAN, and as such, it is important to investigate if it can be modified to further reduce population variance. Based on Ref. [25], it is possible to implement an importance-weighted combing in GUARDYAN as well. Definition of the importance function is however not trivial considering the time dependence of the problem.

4.1.1 The importance weighted comb

The importance weighted version works similar to the simple comb, only the selection of particles is based on $Iw$, where $I$ stands for the importance of the particle, shown in Fig. 4.1. The weight assigned to a post-combed particle will be

$$w'_i = \sum_{j=1}^{N} \frac{w_j I_j}{MI_i}.$$  \hspace{1cm} (4.1.1)

The advantage of using the importance based comb is that it can improve variance by increasing the probability of making more copies of a particle with high importance, while also facilitating the termination of particles in low-importance regions such as a slow neutron bouncing in the moderator far from the fuel pins, thus saving computation time. Performance analysis of the improved combing method is given in Section 4.4.2.
Figure 4.1: The importance weighted comb (bottom) vs. the simple comb (top). The importance weighted comb keeps particle 1, while a simple comb would terminate it due to its low weight. The relatively low importance of particle 2 causes the importance weighted comb to make only one copy, whereas the simple comb would make 2.

4.1.2 Importance definition and generation

We approximate the importance of a neutron by the probability of initiating a fission chain, i.e. by the next fission probability. This probability is generated by a GUARDYAN simulation with a time cutoff to reduce computation time. Although importances will be vaguely approximated this way, we preserve our intention on keeping the computational burden of importance generation low compared to a standard dynamic MC simulation. We will show that major improvements can still be achieved this way. Other, more accurate importances can also be calculated e.g. based on the adjoint flux. The adjoint flux can be approximated by the iterated fission probability (IFP) using a forward MC calculation. That requires the simulation of many generations of neutrons until convergence to the fundamental mode is reached. The IFP was not found suitable for GUARDYAN simulations as the variance of importance estimates was proven difficult to contain within reasonable limits. Instead a deterministic solver was considered to provide an adjoint solution, up to the writing of this thesis however, such calculations were not performed.

The next fission probability is tallied in logarithmically placed energy groups with equal spatial reso-
tion, but disregarding directional dependence. GUARDYAN launches 400 starters per space-energy bin, simulating $2^{22}$ ($\approx 4$ million) neutron histories which are terminated upon fission, leakage or escaping the time boundary. The ratio of the summed weight of particles reaching fission to the total weight of starters will yield the importance of the space-energy bin. Typical length of the calculation takes about 3 hours (for the BME TR geometry), showing that estimating only the next fission probability is already very costly. An example importance map is given in Fig. 4.2.

![Figure 4.2: Importance of low energy (left), medium energy (middle) and high energy (right) neutrons in the BME TR zone. Red coloring indicates high importance, while blue colors indicate low importance](image)

Fig. 4.2 gives a nice representation of the joint distribution of neutron importance in space and energy. Geometric detail has little significance for high energy neutrons, which are likely to travel through large portions of the zone without making any interactions. On the other hand, transport of lower energy neutrons is on an entirely different scale, importance generation requires pin-by-pin resolution. Importance maps further show that, if once escaped, low energy neutrons are unlikely to return to the zone in the time-frame of the simulation, while high energy neutrons also have significant importance outside the zone. We also see, that statistical uncertainty of importances are much lower for low energies, that is due to the high escape probability of medium and high energy neutrons.

### 4.2 The event-based GUARDYAN

Parallelism in MC neutron transport calculations is achieved by processing particle histories simultaneously by independent working units. On the GPU it is very straightforward to assign each thread to a neutron history and launch one big kernel function to do all the transport calculations. This is called the history-based method. Since neutron histories can be quite diverse, parallel efficiency can be undermined by
thread divergence, especially if some histories require solving simple tasks while others involve large computational cost. The event-based method first sorts neutrons into stacks according to the event they will undergo; in GUARDYAN, we sort neutrons based on the sampling procedure that needs to be applied next, be it path length selection or a certain ACE law. Then, each stack is executed separately, eliminating thread divergence on this level.

4.2.1 Previous work on event-based Monte Carlo

First successful attempts to vectorize MC calculations were made in the 1980s, following the appearance of efficient vector computers. Modification of the conventional MC algorithm was discussed in detail by [67], elaborating on both general and specific aspects of vectorization. The first example of a vectorized general purpose MC code, RACER3D, reported speedups of at least 10 for a PWR (pressurized water reactor) setup [68]. Efficiency gain of a factor of 8-22 was found when implementing the vectorized MVP continuous energy MC code [69]. In a paper from 1987 [70], Martin summarizes several successful implementations and possible variants of the method (stack-driven version, zone-selection method), but raises concerns about the level of efficiency of vectorized MC algorithms on future computing platforms. And indeed, current studies indicate that the acceleration gain through vectorization may not be as emphasized as before, its merits are unclear. This is mainly due to the versatility of advanced architectures, and that many built-in (implicit) and explicit optimization tools support the efficient execution of the task.

The event-based algorithm has already been implemented for GPU in several cases [71] [18]. However, recent studies point out that eliminating thread divergence may not be a decisive factor in the efficiency optimization of MC algorithms on the GPU. Although the occurrence of thread divergence was successfully reduced in [72], the vectorized version was found to be 10 times slower than the conventional algorithm. Memory latency issues and suboptimal arrangement of thread working load was accounted for the poor results. [73] suggests that a history-based algorithm can be significantly accelerated by only minor adjustments to the code. This is confirmed by [74], observing speedups of 2-4x over a 8-core CPU implementation. Following up on this idea, [19] reports that performance of a big-kernel history-based approach is comparable to the performance of the event-based version in a one dimensional setup with binary stochastic media. Thus the efficiency of vectorized MC simulations on GPUs is still up to debate. As the vectorization of a MC code requires nearly complete rewrite of the algorithm, production-level codes may suffer more than gain from vectorization (validation and verification work would be rendered), but new codes could efficiently exploit GPU capabilities [75]. The possibility of a hybrid simulation scheme is also worth to consider, when the event-based version is combined with the traditional history based method. Theoretical advancements on
the prediction of parallel efficiency of event-based algorithms were recently developed in [76], a comparison of history- and event-based approaches in more complex geometries has not been discussed yet, and will be remedied in Section 4.4.3.

4.2.2 Thread Divergence

When implementing a MC neutron transport code on the GPU special attention must be paid to choosing the right kernels, as the slowest working thread will determine the efficiency of parallelization. All other threads in a warp must wait for the thread finishing last. Loops, conditional and branching statements lead to thread divergence, an uneven distribution of work-load. This issue is targeted by the vectorization of the code, i.e. the event-based MC simulation. In event-based GUARDYAN this is implemented by distinguishing kernel functions for different types of events instead of just one "big" kernel (as in the history-based GUARDYAN). This is illustrated in Fig. 4.3.

![History-based and Event-based Processing of Particle Histories](image)

Figure 4.3: History- and event-based processing of particle histories including two events (red and blue)

When calling these kernels, threads of a warp executing the same operations on particles (the same event is simulated) do not branch, whereas in a history-based simulation threads may easily diverge as one particle may be in a free flight while the other scatters or induces fission. Branching statements in a warp are executed serially in CUDA: an if-else statement is executed for all warps in two cycles (both branch is executed one after the other). When the "if" branch is executed, the threads that do not satisfy the condition (would diverge to the "else" branch) are flagged and perform a NOP (no operation). This results in the degradation of parallel performance. However, it does not necessarily mean that the vectorized version of the code will execute faster. One reason was given in the previous section, regarding memory management issues. Neither should we neglect that the compiler also does some optimization to reduce the penalty due
4.3 Accelerating point-in-cell search

Location of a particle in GUARDYAN is searched in a universe-based combinatorial solid geometry structure. Point-in-cell search decides which cell the particle is in by analyzing a tree structure of nested universes, lattices and surfaces. This can consume significant computer time and the cost of finding this material increases with the complexity of the system, and can also depend on the implementation of the input. Straightforward implementation of the algorithm would include a linear search for finding the cell in question in a tree structure of nested universes, starting the search from the top universe, the root of the tree. Our main idea was to reduce the height of this tree structure by finding the common parent node of cells in a certain region. First, a cartesian mesh is defined over the whole system. Neutrons are easily tracked in this grid by ray-tracing with a simple scaling, thus the voxel containing the collision site is always known. The lowest common ancestor (LCA) of cells in this voxel is pre-calculated, thus linear search can start from this common node, which is anticipated to be much lower in hierarchy than the root. In a worst case scenario, the common node is also the root, in this case there is no speedup. The pre-processing of geometry yielding the LCA in each voxel is performed by random sampling. A number of uniformly distributed points are scattered in each voxel and the cell search is executed for these positions, recording all search paths. The last common node of these paths will be selected as the LCA.

4.4 Performance assessment and analysis

So far we described some methods that were developed for GUARDYAN to improve its performance. A framework will now be set up in order to evaluate these methods. MC performance analysis thus require both the calculation of variance and measurement of computing time. While the latter does not usually pose a challenge, variance analysis of time-dependent MC tallies is far from trivial. Generally, in MC, uncertainty of a tally is determined by the estimated relative error, i.e. the standard deviation of the sample mean divided by the sample mean. If $N$ independent starters score with the $i$−th score being $x_i$, then the variance of the sample mean $\bar{x}$ is

\[
\text{Var} [\bar{x}] = \text{Var} \left[ \frac{1}{N} \sum_{i=1}^{N} x_i \right] = \frac{1}{N^2} \sum_{i=1}^{N} \text{Var} [x_i] = \frac{1}{N^2} \sum_{i=1}^{N} \sigma^2 = \frac{1}{N} \sigma^2.
\]
The variance of the population, $\sigma^2$, is estimated by

$$s^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2 \approx \frac{1}{N} \sum_{i=1}^{N} (x_i)^2 - \left( \frac{1}{N} \sum_{i=1}^{N} x_i \right)^2$$

(4.4.2)

i.e. keeping track of the sum and the sum of squares. The squared relative error is by definition:

$$r^2 = \frac{Var[\bar{x}]}{\bar{x}^2} = \frac{1}{N} \frac{\sigma^2}{\bar{x}^2}$$

(4.4.3)

and is estimated by

$$R^2 = \frac{1}{N} \frac{s^2}{\bar{x}^2} = \frac{\sum_{i=1}^{N} (x_i)^2}{\left( \sum_{i=1}^{N} x_i \right)^2} - \frac{1}{N}$$

(4.4.4)

$x_i$ accumulate all contributions from the $i -$th particle history (and its progenies), thus scores $x_i$ are independent. For the analysis of static tallies, this does not cause any problem. In the time-dependent however, all live particle samples will eventually descend from a single starter particle almost surely (neutrons are lost to leakage, while new, independent samples are not introduced unless an external source is present).

It means that variance estimates by the above method become unreliable, as the number of scores from independent starters declines over time. One could argue that after some time, a neutron does remember its ancestor, i.e. progenies of the same neutron become non-correlated over time. In k-iterations for example, while correlations are also present they are assumed to be negligible if certain criteria on the population size and on the number of cycles are met. Also, some statistical checks are usually performed to verify that the assumption is valid. In kinetic calculations, such measures could be taken as well, but unfortunately, when one considers dynamic simulations (feedback), yet another source of correlation is added since neutron histories now affect each other through feedback. To assess the performance of kinetic calculations with GUARDYAN, we choose an alternative approach to do a MC variance analysis, working from many independent runs and use large datasets in order to evaluate the implemented variance reduction and acceleration methods.

### 4.4.1 Variance analysis of time-dependent tallies

A major concern regarding the variance analysis of time-dependent tallies was already shown as a theoretical limit of standard variance calculation, the extinction of independent tallies over time. Now let us present a practical problem if one is to ignore correlations between time steps (assumes tallies are independent in different time bins). In Fig. 4.4, a single run with GUARDYAN (red) is plotted against the true solution (black) estimated as the average of 18 individual runs. In the time series of the single run, low and high frequency patterns can be distinguished. High frequency pattern appears as white noise, that is due to the uncorrelated statistical uncertainty of estimates, on the other hand low frequency patterns are probably affected by correlations. For such a variance analysis this means that the MC uncertainty of estimates in individual time steps tells little about the deviation of the MC result from the true solution.
As an alternative, one could directly measure the distance between the produced curve and the true solution, but this raises the question of which time interval this distance should be calculated over. Comparing single runs over a certain time interval can easily lead to wrong conclusions, as illustrated in Fig 4.5. In the simulation using variance reduction tools (red curve) MC uncertainty seems to be smaller and waving is more confined, overall it shows more stability. On the other hand, it is undoubtedly further away from the reference solution than the green curve, when no variance reduction was used, on this particular interval. Taking another time interval, we may find that the red curve is closer to the reference solution, as is the case in the last 10 milliseconds in Fig. 4.5. Thus the distance of a single run from the true solution does not seem to appropriately describe the simulation error either. Other statistical measures (e.g. the difference of local minimum and maximum over a time period) could also be considered, here, we describe a method that works with large datasets from several independent runs.

Figure 4.4: Power evolution of a single run vs. average power estimated from 18 independent runs. Power was normalized to the time average of the black curve.
Figure 4.5: Illustration of the problem of comparing single runs. We see that the MC uncertainty of the red curve is far lower, but is further from the reference solution compared to the green line in this particular time period. On the long run, both estimate the total power correctly. Power was normalized to the time average of the reference solution.

Initially, the true solution is estimated by averaging individual runs with 18 different random seeds, shown with black in Fig. 4.4. Then, instead of a single run, we take the average of 5 runs with different seeds. These averages are then tested against the estimated true solution by calculating the mean squared error (MSE):

\[
MSE = \frac{1}{N} \sum_{i=1}^{N} (P(t_i) - \overline{P}(t_i))^2
\]

(4.4.5)

that is the average of squared differences between the solution of the investigated method (\(P\)) and the reference solution (\(\overline{P}\)) over \(N\) time steps.

4.4.2 Performance analysis of combing in GUARDYAN

The above variance analysis was applied to evaluate acceleration and variance reduction methods to assess the performance of GUARDYAN with various parametrization. We are particularly interested in how the size of time steps influences performance, or how the importance weighted comb performs opposed to the simple comb. Simulations were performed in the MC model of the Training Reactor (see Fig. 5.1), simulating a period of 0.1s divided into \(N = 10000\) time steps. Three scenarios were considered: a steady state reactor
operation with critical rod positions, a supercritical state with all rods drawn out, and a subcritical state with rods fully inserted. The effect of combing frequency was investigated by choosing different time step sizes: $10^{-4}s$, $10^{-5}s$ and $10^{-6}s$; and simulation was executed for both the simple and the importance weighted comb. Importances were generated in line with the description of Sec. 4.1, using an importance grid of size 0.6x0.6x5cm and 10 energy bins uniformly distributed on logarithmic scale. The importance map was constructed by a forward MC calculation that took 3 hours using 400 samples per space-energy bin. Figures 4.6, 4.7 and 4.8 show the time evolution of the total reactor power in all scenarios, these plots include all data sets of the simulations meaning that the result of each individual run is shown.

Figure 4.6: Individual runs on the critical system
Figure 4.7: Individual runs on the supercritical system

Figure 4.8: Individual runs on the subcritical system

We see that the importance weighted comb (shown with green) seems clearly better in terms of distance from the true solution (black) in all three scenarios, i.e. although the importance was constructed assuming
the geometry of the steady state, it helps to reduce variance in super- and subcritical cases nevertheless. This observation implies that the importance function is useful if constructed only once for a transient simulation. This is confirmed by Table 4.1 showing MSE values. Such a conclusion is powerful as the cost of a single importance generation is much less than the total cost of transient MC simulation in contrast to static calculations, where it may not be even worth it to compute the importance. Therefore runtime figures in Table 4.2 do not include the time to generate the importance. The Figure of Merit (FoM) values in Table 4.3 is calculated by

\[ FoM = \frac{1}{MSE \cdot T} \]  

(4.4.6)

where \( T \) stands for the runtime.

<table>
<thead>
<tr>
<th>Time step size (s)</th>
<th>( 10^{-4} ) s</th>
<th>( 10^{-5} ) s</th>
<th>( 10^{-6} ) s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple comb</td>
<td>0.01383</td>
<td>0.00795</td>
<td>0.00524</td>
</tr>
<tr>
<td>Importance weighted comb</td>
<td>0.00647</td>
<td>0.00272</td>
<td>0.00283</td>
</tr>
<tr>
<td>Subcritical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple comb</td>
<td>0.00079</td>
<td>0.00030</td>
<td>0.00028</td>
</tr>
<tr>
<td>Importance weighted comb</td>
<td>0.00031</td>
<td>0.00014</td>
<td>0.00015</td>
</tr>
<tr>
<td>Supercritical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple comb</td>
<td>0.23024</td>
<td>0.12900</td>
<td>0.12975</td>
</tr>
<tr>
<td>Importance weighted comb</td>
<td>0.12069</td>
<td>0.07615</td>
<td>0.06439</td>
</tr>
</tbody>
</table>

Table 4.1: Mean-square errors (MSE)

<table>
<thead>
<tr>
<th>Time step size (s)</th>
<th>( 10^{-4} ) s</th>
<th>( 10^{-5} ) s</th>
<th>( 10^{-6} ) s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple comb</td>
<td>3.71</td>
<td>3.89</td>
<td>9.30</td>
</tr>
<tr>
<td>Importance weighted comb</td>
<td>5.04</td>
<td>5.27</td>
<td>11.34</td>
</tr>
<tr>
<td>Subcritical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple comb</td>
<td>4.30</td>
<td>4.24</td>
<td>10.32</td>
</tr>
<tr>
<td>Importance weighted comb</td>
<td>6.06</td>
<td>5.92</td>
<td>12.57</td>
</tr>
<tr>
<td>Supercritical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple comb</td>
<td>3.30</td>
<td>3.64</td>
<td>8.55</td>
</tr>
<tr>
<td>Importance weighted comb</td>
<td>4.38</td>
<td>4.96</td>
<td>10.48</td>
</tr>
</tbody>
</table>

Table 4.2: Runtime in hours
Table 4.3: FoM in 1/h units

<table>
<thead>
<tr>
<th>Reactor State</th>
<th>Simple comb</th>
<th>Importance weighted comb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>2796</td>
<td>7465</td>
</tr>
<tr>
<td></td>
<td>5387</td>
<td>15180</td>
</tr>
<tr>
<td></td>
<td>6723</td>
<td>7892</td>
</tr>
<tr>
<td>Subcritical</td>
<td>5766</td>
<td>8684</td>
</tr>
<tr>
<td></td>
<td>19477</td>
<td>25470</td>
</tr>
<tr>
<td></td>
<td>7837</td>
<td>13432</td>
</tr>
<tr>
<td>Supercritical</td>
<td>2893</td>
<td>3988</td>
</tr>
<tr>
<td></td>
<td>5815</td>
<td>7229</td>
</tr>
<tr>
<td></td>
<td>2263</td>
<td>2493</td>
</tr>
</tbody>
</table>

Table 4.1 shows that variance is greatly reduced by decreasing time step size from $10^{-4}$s to $10^{-5}$s, but further improvement cannot be achieved by choosing an even smaller time step size of $10^{-6}$s. If we consider runtimes (Table 4.2), we see that while the change from $10^{-4}$s to $10^{-5}$s does not really affect computational cost, choosing time step length of $10^{-6}$s results in substantial slowdown. This is due to the frequent synchronization degrading parallel performance, as well as the increased cost of combing, particularly the sorting of particles during a comb. The overall effect of these factors is that the most efficient simulation uses step size around $10^{-5}$s, as FoM values are highest for this parameter as seen in Table 4.3. Also the optimal step size does not seem to be strongly influenced by the reactor state, but finer tuning of this parameter should be considered in the future.

4.4.3 Performance of the event–based GUARDYAN

Test Case #1 - Verification Setup

The implementation of interaction physics GUARDYAN was partly verified via comparison of differential quantities to MCNP6 results, which was detailed in Section 2.3.1. This verification provided a nice opportunity to also analyze the event-based GUARDYAN in a very simple setup, where meaningful conclusions are quite easy to find, particularly to unveil factors affecting the performance of the event-based structure. This simple setup assumes a monoenergetic neutron source inside a homogeneous sphere with radius adjusted to the mean-free-path of neutrons. In the simulations $4 \cdot 10^6$ neutrons were launched at energies $0.01eV$, $1eV$, $1keV$, $1MeV$ and $18MeV$, and tracked until either leaking out of the sphere or exceeding time boundary. The simulation was carried out in 412 spheres filled with different isotopes. Cross section library ENDF/B-VII.1 was used assuming temperature of $293.6K$. Regarding our investigation of event-based tracking, wall-time
was measured for both history-based \((T_H)\) and event-based \((T_E)\) simulations. In Fig. 4.9, histograms of simulation speedup are plotted for all starting energies. Speedup is simply defined by

\[
\text{Speedup} = \frac{T_E}{T_H}
\]  

(4.4.7)

i.e. the ratio of wall-times. Fig. 4.9 shows that vectorization of the code resulted in faster execution time in most cases. Typical speedup was around 1.5-2, but longer simulation time was observed mainly when starting energy is below 1\( MeV \). The efficiency loss was experienced in case of isotopes with high probability for fission around the starting energy. When the starting energy is low, neutrons released in fission take on much higher velocity than starters, thus leaking out of the system very fast. As a result, significant part of computational effort was spent on a few neutrons bouncing around in the system. Population drop caused vectorization gain to be cancelled due to the computational overhead of event-based tracking (particles need to be sorted by event type). On higher starting energies, no considerable speedup was observed in case of elements with low atomic numbers, the improvement from vectorization was more expressed when heavy elements were present. This is due to that the outgoing energy and angle of a neutron scattered on a light isotope are derived by simple laws of collision mechanics, while more complicated energy laws are applied when heavier isotopes are present [78]. In GUARDYAN, beside elastic scatter only ACE law 3 (inelastic discrete-level scattering) was used in the former case, and ACE law 4 (and 44) was additionally used in the latter. ACE law 4 represents a continuous tabular distribution, the outgoing energy is given as a probability distribution for every incoming energy [27]. This sampling procedure takes considerably more time, contributing to thread divergence, and resulting in substantial efficiency boost for event-based tracking.

**Test Case #2**

Our investigations in the verification setup pointed out that the efficiency of event-based tracking is significantly reduced when the neutron population decreases. Highest speedups were detected when the simulation used several sampling laws associated with various computational cost for calculating the outgoing energy and angle of a neutron. In test case #2, we assumed an inhomogeneous medium, depicted in Fig. 4.10. The geometry contained 61 uranium dioxide rods embedded in a light water sphere.

Table 4.4 shows execution times measured during the simulation of neutron transport in the inhomogeneous sample problem. Wall-times of history-based and event-based versions show no significant difference, the vectorized code performed slightly better.
The event-based version was tested on the geometry of the training reactor at Budapest University of Technology and Economics, shown in Fig. 5.1.

We experienced, that the vectorized code ran 1.5x slower than the history-based algorithm. To better understand the underlying reasons we looked into the kernel execution times. In case of the event based version of GUARDYAN, every energy law was implemented in a separate kernel, thus an application profiling tool is able to reveal which task consumed most resources. Inspecting the profile shown in Fig. 4.12, several conclusions can be made:

- The main part of the execution time is due to calling the "transition kernel". This function transports a particle to the next collision site, and performs the selection of reaction type for that particle.
Long calculation time is most likely caused by the Woodcock method used for path length selection (a phenomenon termed the heavy absorber problem) and slow point-in-cell search algorithms implemented in GUARDYAN.

- Memory transaction costs are much greater than computational costs of simulating different reactions. The "CUDA memcpyDtoH" and "CUDA memcpyHtoD" tasks stand for the communication between host and device, taking up more simulation time than simulating elastic scatter and ACE laws.

- The "Thrust sort" kernel includes all computational overhead that is associated with event-based tracking. Note, that sorting is done two orders of magnitudes faster than memory transactions.

Fig. 4.12 indicates that history based tracking may be more effective because most of the calculation time is due to calling one kernel (called "transition kernel") which is applied to all particles before every collision. In order to execute the simulation of any type of reaction, the event based version must wait for the transition step to end for all particles. On the other hand, the history-based simulation can go on
unsynchronized, i.e. threads may diverge (one may execute a transition step while the other simulates a collision), but threads do not need to wait for others to proceed. By optimization of the transition step, event-based GUARDYAN could however outperform the conventional history-based tracking.

### 4.4.4 Evaluation of acceleration methods

Profiling of GUARDYAN, shown in Fig. 4.12, points out that the slowest working unit of the application is the transition of particles between collisions. Obviously, this means that acceleration of the code must target more efficient algorithms for this particular routine, as attempting to optimize any other part of the code would have little effect on overall runtime. This is also supported by the fact that improvement in runtime was not observed, despite the thread divergence issue being successfully addressed by the event-based version.

![Figure 4.12: Profile of the application without acceleration methods](image)

Slowest GPU kernel was identified as the transition kernel = cell search + collision sampling

In this section we present the performance of the two methods designed to accelerate the calculation of transporting particles between collisions. The cell search algorithm described in Sec. 4.3 focuses on finding the location of the current collision site more quickly, while the majorant mesh method (cf. Sec. 3.7) reduces the frequency of using this cell search routine. Application of these methods and their influence over
runtime were investigated in both the history-based and event-based versions of the code. Simulations were performed for a single time step of $10^{-5}$ s in the Training Reactor model (Fig. 5.1). The majorant mesh was generated in line with the description of Sec. 3.7, using voxelized grid of size $0.3\times0.3\times0.6\text{cm}$ and 1000 energy bins uniformly distributed on logarithmic scale. The majorant map was constructed by using 400 samples per space-energy bin. Results are summarized in Table 4.5.

<table>
<thead>
<tr>
<th></th>
<th>History-based</th>
<th>Event-based</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Majorant mesh OFF</td>
<td>Majorant mesh ON</td>
</tr>
<tr>
<td>Cell search OFF</td>
<td>101.98 s</td>
<td>29.51 s</td>
</tr>
<tr>
<td>Cell search ON</td>
<td>2.89 s</td>
<td>1.91 s</td>
</tr>
</tbody>
</table>

Table 4.5: Performance of acceleration methods in the history- and event-based GUARDYAN

The obtained kernel execution times indicate that the majorant mesh accounts for the largest part of the acceleration (speedup of 34-35x). The improved cell search routine can also improve runtime by a factor of 3, but the performance gain is less when the majorant mesh is turned on, in this case the acceleration is merely 50%. The same is true backwards, the standalone application of majorant mesh yields a speedup of 34-35x, but the acceleration gain solely due to the majorant mesh is only 15x when the improved cell search routine is turned on as well. The observable performance gain strongly depends on several factors, including the complexity of the system, and the majorant cross section. In the Training Reactor model, the majorant mesh method proved to be a very successful technique, presumably due to cadmium being present in the control rods, causing the majorant cross section to be huge at lower neutron energies.

Acceleration was found to be the same regardless of the execution structure, the same speedups are experienced with event-based an history-based method. This means that conclusions made in the previous section still hold, i.e. the transition of particles between collisions still consumes most of the runtime. This is not at all surprising if we take a look at Fig. 4.12, kernel execution time of the transition step is more than three orders of magnitude longer than runtime requirement of any other kernel function (memory transaction costs are expected to be close to equal in the two cases). Since acceleration methods could account for speedups of 50-60, however promising these figures may seem, they are not good enough to allow
the event-based GUARDYAN to show the advantages of vectorization.
Transient analysis with GUARDYAN

GUARDYAN was challenged to simulate the transient behavior of a nuclear reactor. Transient analyses were for now restricted to kinetics only, TH feedback effects were not considered. Beside having an impression about the performance of kinetic MC calculations on the GPU in real scenarios, such analyses are also able to validate the code before moving on to TH coupling. We decided to test GUARDYAN against experimental data, since, up to the writing of this thesis, benchmarks for kinetic calculations using continuous energy neutron data were not available. The Training Reactor at Budapest University of Technology and Economics provided an excellent opportunity for carrying out such an experiment. A reactivity initiated transient was monitored via fission chambers and high count rate data acquisition units, and the transient was simulated over 3-4 seconds with GUARDYAN.

5.1 Benchmarking a kinetic Monte Carlo solver

Existing transient benchmarks are either proposed for coupled neutronics/thermal hydraulics codes with no true time-dependent MC modules, but iterative solutions like in [79], [80] and [81], or assume very simplified systems such as [82] and [83]. Also, many transient benchmarks were prepared for deterministic codes with non-continuous energy cross sections, e.g. [84], or recently [85] [86] and [87]. A MC kinetic benchmark model, excluding thermal hydraulic feedback, for whole core transient analysis is very much needed in general 1. Separating feedback effects from the benchmark would be advantageous in view of testing only the time-dependent neutronics implementation, which is far from trivial, considering the treatment of delayed neutrons, the time factor, population control, etc.. So far attempts on dynamic MC calculations resorted to oversimplified, hypothetical models to verify the correct implementation of neutronics. For examples, see [2]

---

1 A benchmark model of a TMI-1 3x3 minicore was created [88] at the time of writing this thesis.
for a simplified benchmark based on [84], or [9] for a benchmark based on [89]. Besides, while it is especially necessary for novel codes like GUARDYAN, a kinetic benchmark would also benefit the verification work of otherwise thoroughly tested general purpose tools being extended with a time-dependent module.

Our motivation in regard to testing GUARDYAN against experiment is twofold: first, we wish to check the implementation for any discrepancy in a transient scenario before moving on with coupling the code to thermal hydraulic calculations, second, we intend to demonstrate the transient capabilities of GUARDYAN and get an impression about the performance of whole core transient analysis. We also wish to set grounds for a future MC kinetic benchmark study that would provide opportunity for code-to-code comparison of time-dependent MC codes.

5.2 Experimental setup

The Training Reactor of the Budapest University of Technology and Economics (BME TR) is a pool-type, water moderated nuclear reactor. The facility has been operating since 1971, using EK-10 type fuel assemblies with 10% $^{235}$U enrichment. The nominal maximum thermal power of the reactor is 100 kW, and the maximum thermal neutron flux is $2.7 \cdot 10^{12} \text{n/cm}^2\text{s}$. The reactor core is located in a cylinder-shaped tank of 1400 mm diameter and 20 mm thickness, made of Al-alloy, which is filled with demineralized water. The fuel in the reactor is UO$_2$ in magnesium matrix and it is placed in cylinder-shaped Al-alloy cladding with a thickness of 1.5 mm. The fuel pins are 10 mm in diameter and 500 mm in active length. At each end of the fuel pins, there are inactive parts with the length of 45 mm. The current core of the BME TR (see in Fig. 5.1) contains a total number of 24 fuel assemblies that contain a total of 369 fuel pins. Besides the 9 regular fuel assemblies with 16 pins in a 17 mm pitch regular square lattice, 15 fuel assemblies of 5 different designs with missing pins or distorted lattice are used in order to form space for control rods and irradiation positions. The core also contains 41 solid graphite blocks with aluminum cladding and a holder element, which serve as the reflector around the core. Some of the graphite blocks are penetrated by aluminum elements in the cut of the horizontal beam tubes to increase the beam intensity. In the corner assemblies, graphite blocks are cut diagonally, the remaining space is filled by air. The irradiation position of the thermal rabbit system is located in the reflector surrounded by water to maximize the thermal flux. The fast rabbit system is located in the middle of the core replacing a half fuel assembly and is surrounded by air in order to increase the fast flux in the irradiation position. During the experiments the pneumatic rabbit system was used to insert a small absorber into the core, and observe the change in reactor power. The absorber used in the experiments was a ring made of metallic cadmium. The height of the sample was 40 mm, the outer diameter was 9 mm and the thickness was 0.5 mm. The ring was placed in a polyethylene
casing which is regularly used with the pneumatic rabbit system. The rabbit may be considered as a cylinder of 17 mm outer diameter, 3 mm thickness and 58 mm height - including 4 mm and 5 mm solid caps at its top and bottom, respectively.

At the start of each measurement, the reactor operated at critical state. In order to avoid thermal feedback effects, all experiments were carried out at low power with a maximum of 500 W. This was a result of a compromise between expecting relatively good statistics for counts captured under a millisecond and reducing systematic error due to neglecting feedback mechanisms. Time resolution of a millisecond was necessary, as the transient test was intended to demonstrate that GUARDYAN is able to follow rapid changes such as a prompt jump, typically occurring during a few average prompt neutron lifetimes.

The absorber ring was inserted into the core using the 'fast' section (see in Fig. 5.1) of the pneumatic rabbit system and induced a transient behavior. The bottom of the cadmium ring was 2 cm below the midplane of the core when inserted. The ring caused the local neutron absorption rate to rise especially in the thermal range, the overall reactivity insertion was around -20 cents. The reactor power started to decrease exponentially in time after a prompt jump. The transient was observed for a few seconds before the absorber was removed and the reactor returned to a critical state at lower power than at the initial state. The process was repeated for a number of times at different initial power states.

During the transient TTL-format signals of CFUL08 fission chambers in ex-core detector tubes "I1" and "I2" were recorded in time-stamp mode by a field-programmable gate array (FPGA)-based NI-MyRIO device [90], enabling high count rate data acquisition. The delay between rectangle signals was measured with a resolution of 25 ns (clock rate of 40 MHz) by the device, essentially affixing time stamps to individual detector counts. Timestamped data were binned into intervals with length of one millisecond, resulting in detector counts around 1000/bin with statistical uncertainty about 3%. In order to correct for the deadtime effect, the time-interval distribution of the counts was plotted, which showed \( \tau = 150 \text{ ns} \) deadtime, and paralysable deadtime model was used [91]. Measurement data after dead time correction can be seen in Fig. 5.2.
5.3 Benchmark model

A set of experiments were performed at the Training Reactor of the BME in order to produce reactivity transients for the validation of reactor kinetic codes. The experiments utilized the pneumatic rabbit system of the facility to insert and withdraw a piece of Cadmium absorber to and from the core, while the reactor power was recorded based on ex-core fission chambers. This section describes the BME TR, the performed experiment and its benchmark model. All details necessary for modeling can be found in the technical report [92].

The currently available and installed instruments do not make possible to record or to determine the exact position or speed of the rabbits in the pneumatic system. Therefore some assumptions were made based on the measured power curve in order to make a suitable estimation for the position of the rabbit.

Linear functions were fitted over different regions of the normalized count-rate function and then the intersections were determined. The start and end times of these selected regions can be found in Table 5.1. The intersection of the linear functions were assumed to be the departure or arrival time of the sample from or to the given position which are presented in Table 5.2. Constant velocity values were assumed and
determined based on the traveling time and the traveled distance.

Table 5.1: Start and end times of the regions of the different stages

<table>
<thead>
<tr>
<th>Stage</th>
<th>Start [s]</th>
<th>End [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000</td>
<td>1.768</td>
</tr>
<tr>
<td>2</td>
<td>1.841</td>
<td>1.903</td>
</tr>
<tr>
<td>3</td>
<td>1.948</td>
<td>3.462</td>
</tr>
<tr>
<td>4</td>
<td>3.507</td>
<td>3.657</td>
</tr>
<tr>
<td>5</td>
<td>3.752</td>
<td>4.448</td>
</tr>
</tbody>
</table>

Table 5.2: Corresponding positions and times of the Cd-ring. The position refers to the vertical distance between the bottom of the ring and the midplane of the core

<table>
<thead>
<tr>
<th>Position [cm]</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>1.8409</td>
</tr>
<tr>
<td>-2</td>
<td>1.8930</td>
</tr>
<tr>
<td>-2</td>
<td>3.5218</td>
</tr>
<tr>
<td>30</td>
<td>3.6421</td>
</tr>
</tbody>
</table>

Based on the MCNP benchmark model described in [92], an input of the Training Reactor was generated for GUARDYAN. The insertion of cadmium ring was modeled assuming constant velocity of $7.5 \text{m/s}$. The withdrawal was modeled as an accelerating movement starting from rest with acceleration of $30.6 \text{m/s}^2$, based on the lifting force of the vacuum pump. This was necessary since instantaneous insertion and withdrawal model introduced substantial differences from the experimental data.

5.4 Transient simulations

The simulation was run using $2^{22}$ ($\approx 4$ million) samples, in agreement with the data structure of Fig. 2.3. The size of a time step was $10^{-5} \text{s}$ and importance weighted combing was used. Total runtime was around $150 \text{h}$ on a commercially available GPU card (Nvidia GeForce GTX 1080), with acceleration methods of Sec. 4.4.3 turned on for the computation. In a recent study [14], authors report runtime of 3000 CPU hours for a simulation of 10 seconds with the TRIPOLI-4 MC code. This means an order of magnitude difference in
favor of GUARDYAN, however, it remains unknown how comparable these codes may be for realistic models with complexity allowing for e.g. reproduction of measurement and whether the experienced one order of magnitude running time difference in favor of GUARDYAN is indeed a valid observation.

Figure 5.2: Testing GUARDYAN against experimental data. The two curves were normalized for matching power levels of the initial state

In Fig. 5.2 we see that GUARDYAN is able to simulate a reactor transient caused by the insertion of a small cadmium ring, which would be difficult even for deterministic solvers. Changes occurring under tenth of milliseconds are easily followed with GUARDYAN, both the insertion and withdrawal of the cadmium ring is nicely simulated. The agreement between experimental and simulation results is not perfect however. Error sources include the difference between MC model and the actual reactor core, as well as the approximation used at modeling the insertion and withdrawal of the cadmium ring. Another source of error is the uncertainty of detector dead time. Dead time correction was done using the paralyzable model [91], with estimated dead time of $150\,\text{ns}$. This dead period is mainly accounted for by the detector, as other signal processing units in the chain are faster (e.g. time stamps can be generated at a theoretical maximum of 40MHz). However, the detector, as part of the measurement chain I2, is a fixed component of the reactor with no possible way to set up measurements such as dead time determination with a reference source. Thus dead time was determined by analyzing the time–interval–distribution (TID), i.e. the distribution of elapsed time between counts. Ideally, TID would show an exponential behavior, but due to detection dead time, this distribution is somewhat distorted. Fig. 5.3 shows the TID of the transient measurement. Due to the Poisson statistics
of the true counting rate, the distribution is similar to the exponential distribution, but is truncated on the first 150\(\text{ns}\).

![Time interval distribution of the measurement](image)

Figure 5.3: Time interval distribution of the measurement

Based on this observation, dead time was estimated as 150\(\text{ns}\). However, we experienced that dead time strongly influences the goodness of the match between GUARDYAN and measurement data. Assuming longer dead time yielded an almost perfect match between the two data sets, especially affecting the sub-critical period of the time series.

These promising results achieved with GUARDYAN imply that comparison to other dynamic MC codes like [14] would be highly valuable. Since kinetic benchmarks for MC whole core transient analysis are hard to find, we propose the above study to be considered as a possible benchmark problem. Some efforts were already made to extend the benchmark description [93] and the simulation of the BME TR transient was targeted by a research coupling the discrete ordinate neutron transport code PARTISN to the transient driver SEnTRi [94] developed also at BME NTI. Measurement to code comparisons yielded good agreement in the sense that global kinetics processes were found to be accurately modeled despite the very complicated, highly heterogeneous geometry. The major deviation of the deterministic solution from the measurement was due to PARTISN overestimating the reactivity worth of the Cd ring by 1-2\%. Such small discrepancies are however quite easy to notice when looking at time-dependent results as seen in Fig. 5.4. Also, the imprecise timing information of the reactivity insertion/withdrawal left some discrepancies present, shown in Fig. 5.5.
Figure 5.4: Comparison of the measured and simulated relative power during the complete transient [94]

Figure 5.5: Comparison of the measured and simulated relative power during the insertion and removal of the absorber [94]

Due to limitations in instrumentation, kinetic measurements that are outside the validity of point-kinetics or quasi-static approximations can not be carried out in the BME TR. The global power evolution of the above transient with the Cd ring insertion can easily be reproduced by point-kinetics. This was observed in [94], thus in order to test the capabilities of the deterministic code and GUARDYAN for cases where a considerable change exists in the flux shape during a BME TR transient, a hypothetical case was defined in this particular paper. The hypothetical scenario includes an instantaneous insertion of a 20 cm long cadmium sample next to the core into the thermal pneumatic transfer system. The calculated reactivity worth of the sample can be seen in Table 5.3. Fission power was calculated in several fuel assembly quarters (see in Fig.
5.6) in a 10cm long section around the midplane to obtain spatial resolution.

Table 5.3: Reactivity worth determined by Monte Carlo and deterministic calculation [94]

<table>
<thead>
<tr>
<th>Code</th>
<th>Reactivity worth [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCNP6</td>
<td>09.152</td>
</tr>
<tr>
<td>PARTISN</td>
<td>11.012</td>
</tr>
</tbody>
</table>

Figure 5.6: The fuel regions where fission power was compared. The sample is inserted into the thermal pneumatic rabbit system. [94]

The result of the deterministic calculations can be seen in Fig. 5.7. In the first few hundred $\mu$s, power evolution in the cells shows very different behavior which proves that our original goal of creating a fast local transient was achieved. Around 500 $\mu$s the curves have the same slopes, which indicates that the flux shape does not change anymore. A comparison between the deterministic and GUARDYAN calculations was performed for the first 300 $\mu$s showing strong spatial/temporal dependence.

The tallied power changes at the selected locations were in the order of a few percents and had to be locally estimated. Both properties pose a challenge for a Monte Carlo code especially if direct time dependence is concerned as the statistical uncertainty scales inversely with the number of fissions occurring in a time interval in the specified spatial region. The GPU implementation restricts the number of starting neutrons by the physical memory of the GPU, therefore, the simulation of the 300 $\mu$s long interval was repeatedly simulated starting from the same source distribution but using different random seeds and the obtained 700 curves were averaged. In the deterministic simulation, 1 $\mu$s time steps and $S_{12}$ approximation were applied. Due to the small time steps, the calculation converges only with at least such fine angular resolution.

The results from both simulations are compared in Fig. 5.8. The figures suggest good agreement as the two estimations agrees within the statistics of the Monte Carlo results. One may observe that according to the deterministic calculation the power in the 2nd fuel region drops only after 40 $\mu$s, which is most probably due to the moderator (poliethylene casing) inserted together with the absorber. The Monte Carlo results appear to confirm this finding, although the statistics is not good enough to provide a solid proof.
Figure 5.7: Deterministic calculation for the power in the observed fuel assembly quarters. All curves are normalized to the value at 700 µs [94].

Figure 5.8: Relative power in the four fuel assembly quarters during the first 300 µs. The values normalized to the powers at 300 µs [94].
An important conclusion from this exercise is that the sudden insertion of about 10 cents negative reactivity next to the core causes almost 10% deviation in the flux shape in the nearest fuel pins and less then 1% in the middle of the core. These deviations decay in about 0.5 ms and after that the global flux shape in the core does not change and the point-kinetics approximation is fully valid. These observations indicate the required spatial and temporal resolution for an experiment that could provide a validation base for kinetics codes with full spatial dependence.
Chapter 6

Conclusion and future work

In this thesis we considered a MC simulation for reactor dynamics using Graphics Processing Units. Making the calculations feasible, some methodological developments were presented. The concepts were implemented in a novel GPU-based reactor dynamics code GUARDYAN. The code was put through a verification and validation procedure, Code-to-code comparisons, including criticality benchmarks and the comparison of differential quantities, indicated the correct implementation of interaction physics. The proper treatment of time dependence was challenged for whole-core transient analysis and by the reproduction of experimental data. We suggested several variance reduction and acceleration tools to improve the performance of GUARDYAN, many of them were eventually tested and got adopted in the code. To capture the impact of these tools and to obtain some kind of performance analysis in general, we developed a variance analysis framework for time-dependent MC tallies.

Novel contributions to the MC reactor physics field can be summarized in the following thesis points:

1. I have first ever shown, that Graphics Processing Units are now capable to simulate the temporal evolution of prompt neutron fission chains using the Monte Carlo method within reasonable time. I devised a methodology overcoming the population control issue in time-dependent transport problems, via a branchless neutron history method and population-based variance reduction using micro time steps [P1] [P2] [P3] [P4].

2. I addressed the timescale mismatch of delayed and prompt neutrons by the non-analog sampling of delayed neutron precursors. I developed a robust treatment of prompt and delayed neutrons that routinely adjusts neutron population characteristics to follow the time evolution of the system via merging prompt and delayed Monte Carlo samples. I showed that the construction is stable even under heavy transient conditions [P1] [P3].
3. I have devised a null–collision type particle tracking framework, termed the Biased Woodcock framework, allowing efficient sampling of distance to collision in inhomogeneous media. I have revisited previously developed techniques based on the original method described by Woodcock et al. [32], and showed that each can be interpreted as limiting cases of the Biased Woodcock framework. I showed that the framework also allows to prove that neither is optimal in terms of variance or efficiency. Scouting optimal sampling schemes, I have first ever found a theoretical bridge between null–collision type algorithms and exponential transform. [P5] [P6] [P7] [P8].

4. Via restructuring neutron history processing on the GPU, I have provided comparisons between the recently popularized event-based and the traditional history-based methods. I have first ever shown, that the event-based strategy is not superior when considering problems with high geometric detail. I have identified path length selection as the main contributor to performance loss, undermining the thread divergence reduction efforts of the event-based strategy. [P4].

5. I have first ever tried a time-dependent Monte Carlo code for whole–core transient analysis and against measurement data. I designed a transient experiment at the Training Reactor of Budapest University of Technology and Economics and showed that the Monte Carlo simulation is able to reproduce the power evolution of the transient within the uncertainties of the benchmark model. I have found runtime figures to measure up to, and possibly even surpass, those produced by existing references [P3] [P1].

Next step in the development of GUARDYAN is the coupling with a thermal hydraulic code. In this respect, perhaps the most intriguing question is convergence, i.e. the propagation of MC uncertainty in the coupling. It is important to have a proper tool for estimating this uncertainty. Currently, using independent runs is the best practice, but more sophisticated methods based on the Markov property are being investigated.
Acknowledgment

The author would like to thank members of the GUARDYAN developer team, and in particular, Gábor Tolnai, for the enormous work he put in, essentially building GUARDYAN from scratch. I would also like to thank my supervisor, Dr. Dávid Légrády, my colleagues at the Institute of Nuclear Techniques for their useful ideas and comments and Dr. Máté Szieberth for his detailed review on the manuscript of this thesis.

This work was supported by the Higher Education Excellence Program of the Ministry of Human Capacities in the frame of Biotechnology research area of Budapest University of Technology and Economics (BME FIKP-BIO).
Appendix

The geometry descriptor of GUARDYAN

Surfaces are described by second order equations can be defined such as spheres, planes, cylinders, cones and toruses. Possible definitions of surfaces are summarized in Table 6.1.

<table>
<thead>
<tr>
<th>Surface</th>
<th>Notation</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane normal to x-axis</td>
<td>px</td>
<td>$x - x_0 = 0$</td>
</tr>
<tr>
<td>Plane normal to y-axis</td>
<td>py</td>
<td>$y - y_0 = 0$</td>
</tr>
<tr>
<td>Plane normal to z-axis</td>
<td>pz</td>
<td>$z - z_0 = 0$</td>
</tr>
<tr>
<td>Plane with arbitrary orientation</td>
<td>p</td>
<td>$Ax + By + Cz - D = 0$</td>
</tr>
<tr>
<td>Cylinder on the x-axis</td>
<td>cx</td>
<td>$y^2 + z^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Cylinder on the y-axis</td>
<td>cy</td>
<td>$x^2 + z^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Cylinder on the z-axis</td>
<td>cz</td>
<td>$x^2 + y^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Cylinder parallel to the x-axis</td>
<td>c/x</td>
<td>$(y - y_0)^2 + (z - z_0)^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Cylinder parallel to the y-axis</td>
<td>c/y</td>
<td>$(x - x_0)^2 + (z - z_0)^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Cylinder parallel to the z-axis</td>
<td>c/z</td>
<td>$(x - x_0)^2 + (y - y_0)^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Sphere centered at origin</td>
<td>so</td>
<td>$x^2 + y^2 + z^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Sphere centered on x-axis</td>
<td>sx</td>
<td>$(x - x_0)^2 + y^2 + z^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Sphere centered on y-axis</td>
<td>sy</td>
<td>$x^2 + (y - y_0)^2 + z^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Sphere centered on z-axis</td>
<td>sz</td>
<td>$x^2 + y^2 + (z - z_0)^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Sphere – general</td>
<td>s</td>
<td>$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0$</td>
</tr>
<tr>
<td>Ellipsoid, hyperboloid,</td>
<td>sq</td>
<td>$A(x - x_0)^2 + B(y - y_0)^2 + C(z - z_0)^2 + 2D(x - x_0) + 2E(y - y_0) + 2F(z - z_0) + G = 0$</td>
</tr>
<tr>
<td>paraboloid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cone on x-axis</td>
<td>kx</td>
<td>$\sqrt{y^2 + z^2} - t(x - x_0) = 0$</td>
</tr>
<tr>
<td>Cone on y-axis</td>
<td>ky</td>
<td>$\sqrt{x^2 + z^2} - t(y - y_0) = 0$</td>
</tr>
</tbody>
</table>
GUARDYAN assigns a general notation to each surface, treating them as a general quadratic surface described by the following equation:

\[ Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fzx + Gx + Hy + Jz + K = 0 \]  

(6.0.1)

An exception to this is the torus, which cannot be written in this form. As a result, a torus cannot be transformed in the current version. In case of cones, an additional parameter is needed to describe the orientation of the aperture of the cone.

Cells are defined by bounding surfaces and boolean operators. If a cell is inside a surface, then the surface must be written with negative sign, otherwise a positive sign must be used. AND, OR and NOT relations between parts of space can be given by a colon(;), white space( ) and a hash(#) respectively. Parentheses are also allowed. Cells are assumed to be homogeneous, the ID of the material filling the cell needs to be given at the cell definition, or else terms void or dead can be used, meaning that the particle travels through that cell without interaction or is terminated upon entering, respectively.

Universes can be defined consisting of one or more cells, and can be used to fill other cells or arrange them in a lattice. In GUARDYAN, rectangular and hexagonal lattices can be defined. Transformation of surfaces and cells can be given by translation vectors and a rotation matrices.

An example geometry input reads as:

```xml
<geometry>
  <surface id="1" type="cz" transform="301" coeffs="40.000"/>
  <surface id="2" type="c/z" coeffs="5.000 17.32 10.0"/>
  <cell id="2" material="13" surfaces="1" universe="53"/>
  <cell id="3" material="dead" surfaces="-2 -1" universe="58"/>
</geometry>
```
<cell id="4" surfaces="-1" fill="58" transform="4"/>
<transformation id="4" displacement="0.2 0.2 0.2"
rotation="300 210 -90 390 300 -90 90 90 0"/>
<transformation id="0" displacement="0 0 0"/>
<lattice id="58" type="hexx" dimension="9 9" pitch="0.1 0.1"
universes ="
58 58 58 58 58 58 58 58 58
58 58 58 53 53 53 53 58 58
58 58 53 53 53 53 53 53 58
58 53 53 53 53 53 53 53 58
58 53 53 53 53 53 53 53 58
58 53 53 53 53 53 53 53 58
58 53 53 53 53 53 53 53 58
58 53 53 53 53 53 53 53 58
58 53 53 53 53 53 53 53 58
transf ="
0 0 0 0 0 0 0 0 0
0 0 0 4 4 0 0 0 0
0 0 0 4 4 0 0 0 0
0 0 4 4 0 0 0 4 0
0 4 4 0 0 0 4 4 0
0 4 0 0 0 4 4 0 0
0 0 0 0 4 4 0 0 0
0 0 0 4 4 0 0 0 0
0 0 0 0 0 0 0 0 0"/>
</geometry>
Figure 6.1: Layout of the example input describing a hexagonal lattice of cylinders in a hexagonal lattice

Representation of nuclear data in GUARDYAN

GUARDYAN uses nuclear data of ENDF-B-VII.1 [30] library. An ACE-format data [27] is generated using NJOY [31] nuclear data processing system. Table 6.2 lists the considered neutron interactions.

<table>
<thead>
<tr>
<th>MT number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Elastic scattering cross section.</td>
</tr>
<tr>
<td>5</td>
<td>Sum of all reactions not given explicitly in another MT number.</td>
</tr>
<tr>
<td>11</td>
<td>Production of two neutrons and a deuteran, plus a residual.</td>
</tr>
<tr>
<td>16</td>
<td>Production of two neutrons, plus a residual.</td>
</tr>
<tr>
<td>17</td>
<td>Production of three neutrons, plus a residual.</td>
</tr>
<tr>
<td>18</td>
<td>Total fission.</td>
</tr>
<tr>
<td>19</td>
<td>First-chance fission.</td>
</tr>
<tr>
<td>20</td>
<td>Second-chance fission.</td>
</tr>
<tr>
<td>21</td>
<td>Third-chance fission.</td>
</tr>
<tr>
<td>22</td>
<td>Production of a neutron and alpha particle, plus a residual.</td>
</tr>
<tr>
<td>23</td>
<td>Production of a neutron and three alpha particles, plus a residual.</td>
</tr>
<tr>
<td>24</td>
<td>Production of two neutrons and an alpha particle, plus a residual.</td>
</tr>
<tr>
<td>25</td>
<td>Production of three neutrons and an alpha particle, plus a residual.</td>
</tr>
<tr>
<td>28</td>
<td>Production of a neutron and a proton, plus a residual.</td>
</tr>
<tr>
<td></td>
<td>Neutron Interaction Description</td>
</tr>
<tr>
<td>---</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>29</td>
<td>Production of a neutron and two alpha particles, plus a residual.</td>
</tr>
<tr>
<td>30</td>
<td>Production of two neutrons and two alpha particles, plus a residual.</td>
</tr>
<tr>
<td>32</td>
<td>Production of a neutron and a deuteron, plus a residual.</td>
</tr>
<tr>
<td>33</td>
<td>Production of a neutron and a triton, plus a residual.</td>
</tr>
<tr>
<td>34</td>
<td>Production of a neutron and a 3He particle, plus a residual.</td>
</tr>
<tr>
<td>35</td>
<td>Production of a neutron, a deuteron, and two alpha particles, plus a residual.</td>
</tr>
<tr>
<td>36</td>
<td>Production of a neutron, a triton, and two alpha particles, plus a residual.</td>
</tr>
<tr>
<td>37</td>
<td>Production of four neutrons, plus a residual.</td>
</tr>
<tr>
<td>38</td>
<td>Fourth-chance fission.</td>
</tr>
<tr>
<td>41</td>
<td>Production of two neutrons and a proton, plus a residual.</td>
</tr>
<tr>
<td>42</td>
<td>Production of three neutrons and a proton, plus a residual.</td>
</tr>
<tr>
<td>44</td>
<td>Production of a neutron and two protons, plus a residual.</td>
</tr>
<tr>
<td>45</td>
<td>Production of a neutron, a proton, and an alpha particle, plus a residual.</td>
</tr>
<tr>
<td>51</td>
<td>Production of a neutron, nucleus in the first excited state.</td>
</tr>
<tr>
<td>52</td>
<td>Production of a neutron, nucleus in the second excited state.</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>90</td>
<td>Production of a neutron, nucleus in the 40th excited state.</td>
</tr>
<tr>
<td>91</td>
<td>Production of a neutron in the continuum.</td>
</tr>
</tbody>
</table>

Table 6.2: Neutron interactions considered in GUARDYAN

Materials are defined by isotopic composition (atomic ratio or mass fraction) and density.

A sample material input reads as:

```xml
<materials>
  <material id="1" density="19" units="g/cm3" fraction="atomic"
    dataset="80c">
    <isotope name="U-235" fraction="0.9736"/>
    <isotope name="Fe-Nat" fraction="0.0275"/>
  </material>
  <material id="2" density="1.0" units="atom/b-cm" fraction="weight"
    dataset="80c">
    <isotope name="H-1" fraction="2.0"/>
  </material>
</materials>
```
Sampling angular distributions

ACE–data contains angular distributions as a function of the scattering cosine \( \mu_{cm} = \cos(\theta_{cm}) \) in the central mass frame. Thus when the sampled \( \mu_{cm} \) is obtained, a transformation to laboratory frame is needed:

\[
\mu = \mu_{cm} \sqrt{\frac{E'_{cm}}{E'}} + \frac{1}{A+1} \sqrt{\frac{E}{E'}}
\]

(6.0.2)

where \( E'_{cm} \) is the post–collision energy in the central mass frame, \( E' \) is the post–collision energy in the laboratory frame, \( E \) is the energy of the incoming neutron and \( A \) is the mass number of the target nucleus. Obtaining \( E'_{cm} \) and \( E' \) will be discussed in the next section. An azimuthal angle \( \phi \) is sampled uniformly on \([0, 2\pi)\), and the coordinates of the new direction of the neutron \( \vec{\Omega}' = (u', v', w') \) are calculated as

\[
u' = \mu v + \sqrt{1 - \mu^2} (v w \cos \phi + u \sin \phi)
\]

(6.0.3)

\[
w' = \mu w - \sqrt{1 - \mu^2} \sqrt{1 - w^2} \cos \phi.
\]

(6.0.4)

In the ACE–format the scattering cosine in the central mass frame can be given in three ways

- isotropic angular distribution
- scattering in equally probable 32 angular bins
- tabular distribution

By calculating the probability density functions (PDF) and cumulative density functions (CDF) the second representation is merged into the third in GUARDYAN, essentially simplifying the algorithm.

Isotropic scattering

When a reaction is associated with an isotropic angular distribution, the scattering cosine is simply

\[
\mu = 2\xi - 1
\]

(6.0.6)

where \( \xi \) is a random number uniformly sampled on \([0, 1)\).
Tabular distributions

In this case both PDFs and CDFs are given in a table at discrete $\mu$ values. Several tables are available corresponding to $E_1, E_2, ...$ incident neutron energies. First we use a stochastic interpolation based on the energy of the incoming neutron for deciding which table to use. With probability

$$\frac{E - E_i}{E_{i+1} - E_i}$$

the $i$-th table is selected, otherwise the $i + 1$-th is used, assuming that $E_i < E < E_{i+1}$. Suppose that the $i$-th table was selected, then we search the CDF values $c_{i,1}, c_{i,2}, ...$ for which

$$c_{i,j} < \xi < c_{i,j+1}$$

holds, where $\xi$ is a canonical random number. Thus the sampled $\mu$ is between $\mu_{i,j}$ and $\mu_{i,j+1}$, but the final value depends on what kind of interpolation is used. In GUARDYAN, the PDF is either approximated as a piece-wise constant function or linear interpolation is used between tabulated $p_{i,1}, p_{i,2}, ...$ values. If the PDF is piece-wise constant, the sampled $\mu$ is:

$$\mu = \mu_{i,j} + \frac{\xi - c_{i,j}}{p_{i,j}}.$$  \hspace{1cm} (6.0.9)

If linear interpolation is used, to obtain the scattering cosine the following equation must be used:

$$\mu = \mu_{i,j} + \frac{1}{m} \left( p_{i,j}^2 + 2m(\xi - c_{i,j}) - p_{i,j} \right)$$

where

$$m = \frac{p_{i,j+1} - p_{i,j}}{\mu_{i,j+1} - \mu_{i,j}}.$$ \hspace{1cm} (6.0.11)

Sampling energy distributions

Similar to sampling the scattering cosine, the outgoing energy distributions are given in the central mass frame. The transformation to obtain the energy of the secondary neutron in the laboratory frame is

$$E' = E'_\text{cm} + \frac{E + 2\mu_{\text{cm}}(A + 1)}{(A + 1)^2} \sqrt{E E'_\text{cm}}.$$ \hspace{1cm} (6.0.12)

where $E'_{\text{cm}}$ is the outgoing energy in the central mass frame, $\mu_{\text{cm}}$ is the scattering cosine in the central mass frame, $E'$ is the outgoing energy in the laboratory frame and $E$ is the incident neutron energy. Except for elastic scatter when the outgoing energy is the same as the incident energy in the central mass frame, the outgoing energy is determined from either tabular distributions or parametric data. These are referred to as ACE–laws. The following ACE–laws are applied in GUARDYAN.
ACE Law 3: Inelastic scatter from nuclear levels

ACE Law 4: Tabular distribution

ACE Law 7: Simple Maxwell fission spectrum

ACE Law 9: Evaporation spectrum

ACE Law 11: Energy dependent Watt spectrum

ACE Law 44: Kalbach-Mann correlated scattering

ACE Law 61: Correlated energy-angle scattering with tabulated angular distribution

ACE Law 66: N-body phase space distribution

ACE Law 3 – Inelastic discrete–level scattering

For lower excitation energies discrete nuclear levels can be identified by experiments in which a nucleus is left after inelastic scattering. With $Q$ being the excitation energy, the energy of the outgoing neutron can be derived by simple collision mechanics [78]:

$$E' = \left(\frac{A}{A+1}\right)^2 \left(E - \frac{A+1}{A}Q\right).$$ (6.0.13)

When the excitation energy is high, inelastic continuum scattering is assumed, i.e. nuclear levels are considered to be continuous with a certain level density distribution.

ACE Law 4 – Continuous tabular distribution

Similarly to the sampling method applied for angular tabular distributions in Section 6, the outgoing energy tables $E'_{i,1}, E'_{i,2}, ..., E'_{i,J}$ are given for certain values of incident neutron energies $E_i$ with $i = 1, ..., I$. Both probability distribution $p_{i,j}$ and cumulative distribution $c_{i,j}$ tables are available, where, again, $i$ corresponds to the incident energy and $j$ to the outgoing energy bin. First, a stochastic interpolation is performed between tables corresponding to incident energies; with probability

$$f = \frac{E - E_i}{E_{i+1} - E_i}$$ (6.0.14)

the $i-th$ table is selected, otherwise the $i+1$-th is used, assuming that $E_i < E < E_{i+1}$. Here, special attention is required to avoid the violation of governing physics [95], as using any of the tables may result
in an outgoing energy outside the range permitted by the incident energy \( E \). Therefore, the minimal and maximal possible energy is calculated as

\[
E'_{\text{min}} = E'_{i,1} + f(E'_{i,1+1} - E'_{i,1}) \quad (6.0.15)
\]

\[
E'_{\text{max}} = E'_{i,J} + f(E'_{i,J} - E'_{i,J}) \quad (6.0.16)
\]

and the final sampled energy is scaled to these limits.

To obtain a sample \( E' \), we search the CDF values \( c_{i,1}, c_{i,2}, \ldots \) for which

\[
c_{i,j} < \xi < c_{i,j+1} \quad (6.0.17)
\]

holds, where \( \xi \) is a canonical random number. \( E' \) is between \( E'_{i,j} \) and \( E'_{i,j+1} \), but the final value depends on the interpolation applied. In GUARDYAN, the PDF is either approximated as a piece–wise constant function or linear interpolation is used between tabulated \( p_{i,1}, p_{i,2}, \ldots \) values. If the PDF is piece–wise constant, the sampled \( E' \) is:

\[
E' = E'_{i,j} + \frac{\xi - c_{i,j}}{p_{i,j}}. \quad (6.0.18)
\]

If linear interpolation is used, to obtain the outgoing energy the following equation must be used:

\[
E' = E'_{i,j} + \frac{E'_{i,j+1} - E'_{i,j}}{p_{i,j+1} - p_{i,j}} \left( \sqrt{p_{i,j+1}^2 + 2 p_{i,j+1} - p_{i,j}} (\xi - c_{i,j}) - p_{i,j} \right). \quad (6.0.19)
\]

The sampled energy is finally scaled to the energy limits \( E_{\text{max}} \) and \( E_{\text{min}} \):

\[
E'_{\text{scaled}} = E'_{\text{min}} + \frac{E' - E'_{i,1}}{E'_{i,J} - E'_{i,1}} (E_{\text{max}} - E_{\text{min}}). \quad (6.0.20)
\]

**ACE Law 7 – Maxwell fission spectrum**

To obtain the energy of a fission neutron one can sample the Maxwell-Boltzmann distribution defined as

\[
p_{M}(E') = c \sqrt{E'} \exp \left( -\frac{E'}{T(E)} \right) \quad (6.0.21)
\]

where \( E \) is the incident neutron energy, \( T \) is the nuclear temperature, which is the tabulated function of incident energy. Based on rule \( C64 \) in the Monte Carlo Sampler [96] a sample from this distribution can be obtained by

\[
E' = -T \left[ \log(\xi_1) + \log(\xi_2) \cos^2 \left( \frac{\pi \xi_3}{2} \right) \right] \quad (6.0.22)
\]

where \( \xi_1, \xi_2, \xi_3 \) are canonical random numbers. \( E' \) is then subjected to a rejection procedure, i.e. Eq. 6.0.22 must be recalculated if

\[
0 \leq E' \leq E - U \quad (6.0.23)
\]

does not hold, where \( U \) is called the restriction energy included in the ACE data.
ACE Law 9 – Evaporation spectrum

Evaporation spectrum can be assumed when a secondary neutron has sufficient energy to "evaporate" from a compound nucleus. The distribution to be sampled reads as

\[ p_e(E') = cE' \exp \left( \frac{-E'}{T(E)} \right) \] (6.0.24)

where \( E \) is the incident neutron energy, \( T \) is the nuclear temperature, which is the tabulated function of incident energy. For sampling, GUARDYAN uses the same algorithm which was implemented in MCNP6 based on report LA-UR-14-27694 [97]:

\[ E' = -T \log((1 - g\xi_1)(1 - g\xi_2)) \] (6.0.25)

where \( g = 1 - e^{-w}, \ w = (E - U)/T, \ U \) is the restriction energy, and \( \xi_1, \xi_2 \) are canonical random numbers. If

\[ 0 \leq E' \leq E - U \] (6.0.26)

does not hold, then \( E' \) is rejected, and the sampling must be repeated until the condition is satisfied.

ACE Law 11 – Energy dependent Watt spectrum

The Watt spectrum is defined by

\[ p_w(E') = c \exp \left( -\frac{E'}{a(E)} \right) \sinh \left( \sqrt{\frac{b(E)}{E'}} \right) \] (6.0.27)

where \( a \) and \( b \) are tabulated functions of incident energy. This distribution is sampled analog to the algorithm in the MCNP5 documentation [98]. The sample is calculated as

\[ E' = ag \ln \xi_1 \] (6.0.28)

where

\[ g = \sqrt{\left(1 + \frac{ab}{8}\right)^2 - 1 + \frac{ab}{8}} \] (6.0.29)

and \( \xi_1, \xi_2 \) are canonical random numbers. The sampled energy is rejected and the sampling is repeated if either

\[ [(1 - g)(1 - \ln \xi_1) - \ln \xi_2]^2 \leq bE' \] (6.0.30)

or

\[ 0 \leq E' \leq E - U \] (6.0.31)

does not hold.
ACE Law 66 – N–Body Phase Space Distribution

N–body phase space distribution can be best applied if a reaction produces more than two secondaries with similar masses. The PDF of outgoing energy of the \(i\)–th product is:

\[ p_i(E') = C_n \sqrt{E'(E'^{\text{max}}_i - E')}^{\frac{3}{2} - 4} \]  

(6.0.32)

where \(n\) is the number of secondaries, \(C_n\) is the normalization constant, \(E'^{\text{max}}_i\) is the maximal energy of the \(i\)–th product in the central mass frame, and \(E'\) is the outgoing energy. Obtaining a sample from this distribution is done by using rules \(R28, C45\) from [96]. The maximal energy is first calculated as

\[ E'^{\text{max}}_i = \frac{A_p - 1}{A_p} \left( \frac{A}{A + 1} E + Q \right) \]  

(6.0.33)

where \(A_p\) is the mass number of the reaction product, \(A\) is the mass number of the target, and \(Q\) is the Q-value of the reaction. Then, a sample for the outgoing energy can be found by:

\[ E' = \frac{x}{x + y} E'^{\text{max}}_i \]  

(6.0.34)

where \(x\) is a sample from the Maxwell–Boltzmann distribution (cf. Ace law 7) and \(y\) is determined as a second sample from the Maxwell–Boltzmann distribution if \(n = 3\); from

\[ y = - \ln(\xi_1 \xi_2 \xi_3) \]  

(6.0.35)

if \(n = 4\); and from

\[ y = - \ln(\xi_1 \xi_2 \xi_3 \xi_4) - \ln(\xi_5) \cos^2 \left( \frac{\pi}{2} \xi_6 \right) \]  

(6.0.36)

if \(n = 5\). \(\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6\) denote independent, canonical random numbers.

Sampling joint energy–angle distributions

Some reactions have correlated outgoing energy and angle distributions, which cannot be sampled independently as in Section 6 and Section 6. The following algorithms sample joint distributions of scattering cosine and outgoing energy in the central mass frame. In order to transform these phase space coordinates to laboratory frame transformations detailed in Section 6 and Section 6 are needed.

ACE Law 44 – Kalbach-Mann correlated scattering

The energy of the outgoing neutron is sampled according to ACE law 4 described in Section 6. The scattering cosine is then sampled from the conditional probability density function:

\[ p(\mu|E_{i,j} \leq E' \leq E_{i,j+1}) = \frac{A(i,j)}{2 \sinh(A(i,j))} \left[ \cosh(A(i,j)\mu) + R(i,j) \sinh(A(i,j)\mu) \right]. \]  

(6.0.37)
where $A$ and $R$ are parameters defined by which table was used based on the incident energy ($i$), and which outgoing energy bin was selected ($j$). These parameters are also obtained from tables $A_{i,j}$ and $R_{i,j}$. If no interpolation is used then simply:

$$R(i, j) = R_{i,j}$$
$$A(i, j) = A_{i,j}.$$  \hfill (6.0.38)

Using linear interpolation, the parameters read as:

$$R(i, j) = R_{i,j} + \frac{E' - E_{i,j}}{E_{i,j+1} - E_{i,j}} (R_{i,j+1} - R_{i,j})$$ \hfill (6.0.40)

$$A(i, j) = A_{i,j} + \frac{E' - E_{i,j}}{E_{i,j+1} - E_{i,j}} (A_{i,j+1} - A_{i,j})$$ \hfill (6.0.41)

To sample from the conditional PDF we use rules $C39$ and $C40$ from [96]. If a canonical random number $\xi_1 > R(i, j)$ then the sample for the scattering cosine reads as:

$$\mu = \frac{1}{A} \ln \left( T + \sqrt{T^2 + 1} \right)$$ \hfill (6.0.42)

otherwise:

$$\mu = \frac{1}{A} \ln \left( \xi_2 e^A + (1 - \xi_2) e^{-A} \right).$$ \hfill (6.0.43)

where $\xi_2$ is another canonical random number and

$$T = (2\xi_2 - 1) \sinh(A).$$ \hfill (6.0.44)

ACE Law 61 – Correlated energy–angle scattering with tabulated angular distribution

In this case the outgoing energy and scattering cosine is also not independent, but there is no parametric data available, only tabular distributions are given. First, the outgoing energy is sampled according to ACE law 4 in Section 6. Assume, that the $i$–th probability table was selected and the energy was sampled from the $j$–th bin. If the PDF was approximated as a step-wise constant function, then, for the scattering cosine, we use probability tables corresponding to ($i,j$) indices. If linear interpolation was used we simply use the nearest bin $j$ or $j+1$ to the randomly sampled CDF point. From the selected table, the scattering cosine is sampled according to the algorithm described in Section 6 (“sampling tabular distributions”).

Thermal motion correction

Thermal motion of atoms can affect both the probability of neutron interactions and the energy–angle distribution of possible outgoing secondaries. The former is assessed by NJOY [31] as cross sections of
elastic scatter and absorption reactions are already Doppler broadened when they are read by GUARDYAN. Inelastic scatter reactions are almost always threshold reactions, and thermal motion can be neglected as the energy threshold is much higher than \( kT \), with \( k \) being the Boltzmann constant, and \( T \) the temperature. Inelastic scatter cross sections are thus not Doppler broadened by NJOY. Energy–angle distribution of secondaries produced in fission are weakly affected by thermal motion of the target, thus the only relevant reaction when these distributions require correction is elastic scatter. In this case the velocity of the target nucleus must be sampled to allow for correction. Target motion is taken into account if the incident neutron energy is lower than 400 \( kT \) and/or the target is \(^1H\).

**Thermal motion effects on elastic scatter**

Let us assume that a sample obtained for the target velocity is \( v_T \). How one can obtain a sample will be discussed later in this section. In order to take the target motion into account when sampling the outgoing energy–angle distribution, first, transformation of incident energy and angle to central mass frame is required. This is simply because distributions are given in the central mass frame. In fact sampling energy distributions are not of question here since in elastic scatter the neutron does not lose energy in the central mass frame. Angle distribution is usually isotropic, however, some anisotropy can be observed at high neutron energies. Then, the distribution of scattering cosine is given as a tabulated function of incident energy, thus the following procedure can not be bypassed. First, the incident neutron velocity in the central mass frame is calculated as:

\[
v_{cm} = v - \frac{v + Av_T}{A + 1}
\]

where the second term is the velocity of the center of mass, \( v_{cm} \) and \( v \) is the neutron velocity in the central mass frame and laboratory frame respectively, \( v_T \) is the target velocity and \( A \) the atomic mass number. This yields the incident neutron energy:

\[
E_{cm} = \frac{1}{2} m|v_{cm}|^2
\]

and the direction of movement:

\[
\Omega_{cm} = \frac{v_{cm}}{|v_{cm}|}
\]

in the central mass frame. Next we sample the scattering cosine according to Section 6, except the scattering cosine is not yet transformed to the laboratory frame, i.e. Eq. 6.0.2 is not used since it does not hold when the target is not at rest. The post-scattering direction \( \Omega'_{cm} \) is calculated still in the central mass frame according to Eq. 6.0.3. Since the speed of the neutron does not change in the central mass frame, the post–collision velocity reads as:

\[
v'_{cm} = |v_{cm}|\Omega'_{cm}.
\]
Finally the post-collision velocity calculated by
\[ v' = v'_{cm} + \frac{v + A v_T}{A + 1} \] (6.0.49)
and the post-collision energy and angle is determined by
\[ E' = \frac{1}{2} m |v'|^2 \] (6.0.50)
and
\[ \Omega' = \frac{v'}{|v'|}. \] (6.0.51)

**Target velocity sampling and DBRC**

In order to correctly sample target velocity, the thermally-averaged reaction rate must be preserved:
\[ \int R(v_T) dv_T = \int |v - v_T| \sigma(|v - v_T|) p_M(v_T) dv_T. \] (6.0.52)
where \( \sigma \) is the cross section and \( p_M \) is the Maxwell-Boltzmann distribution. Thus the PDF to be sampled reads as
\[ p(v_T) = \frac{R(v_T)}{\int R(v_T) dv_T} = \frac{1}{C} |v - v_T| \sigma(|v - v_T|) p_M(v_T) \] (6.0.53)
where the normalization constant was denoted by \( C \). The Maxwell-Boltzmann distribution for the target velocity reads as
\[ p_M(v_T) = \left( \frac{m}{2\pi kT} \right)^{3/2} \exp \left( -\frac{m|v_T|^2}{2kT} \right), \] (6.0.54)
where \( m \) is the mass of the target and \( k \) the Boltzmann constant. The distribution of target speed \( v_T = |v_T| \) can be derived by changing to spherical coordinates \( (v_T, \mu, \phi) \):
\[ p_M(v_T) dv_T^3 = p_M(v_T) v_T^2 dv_T d\mu d\phi \] (6.0.55)
and then integrating over direction cosine \( \mu \) and azimuthal angle \( \phi \):
\[ M(v_T) = \int_{-1}^{1} \int_{0}^{2\pi} p_M(v_T)v_T^2 d\phi d\mu = \sqrt{2} \left( \frac{m}{kT} \right)^{3/2} v_T^2 \exp \left( -\frac{mv_T^2}{2kT} \right) = \frac{4}{\sqrt{\pi}} \beta^3 v_T^2 \exp \left( -\beta^2 v_T^2 \right) \] (6.0.56)
where we applied the following notation:
\[ \beta = \sqrt{\frac{m}{2kT}}. \] (6.0.57)

Using
\[ |v - v_T| = \sqrt{v^2 + v_T^2 - 2v v_T \mu} \] (6.0.58)
the PDF to be sampled can be written as
\[ p(v_T, \mu) = \frac{4}{\sqrt{\pi} C} \sigma(|v - v_T|) \sqrt{v^2 + v_T^2 - 2v v_T \mu} \beta^3 v_T^2 \exp \left( -\beta^2 v_T^2 \right) \] (6.0.59)
Let us first assume that \( \sigma (|v - v_T|) = \sigma_s \), is constant. A MC sample can be obtained via rejection sampling: instead of directly sampling \( p(v_T, \mu) \), we take a sample from

\[
f_1(v_T) = \frac{1}{C'} (v_n + v_T) \beta^3 v_T^2 \exp (-\beta^2 v_T^2)
\]

and accept it with probability proportional to

\[
f_2(v_T, \mu) = C'' \left( \frac{4 \sigma_s \sqrt{v_n^2 + v_T^2 - 2v_n v_T \mu}}{v_n + v_T} \right)
\]

Note that this method works because \( p(v_T, \mu) = f_1(v_T) f_2(v_T, \mu) \), and \( f_2 \) is bounded and everywhere non-negative function. The sample from \( f_1 \) should be accepted if

\[
\xi < \frac{f_2}{\text{max}\{f_2\}} = \frac{\sqrt{v_n^2 + v_T^2 - 2v_n v_T \mu}}{v_n + v_T}.
\]

where \( \xi \) is a canonical random number and \( \mu \) is sampled uniformly from \([-1, 1]\). The constant \( C'' \) is chosen such that \( f_1 \) is a proper PDF, that is if

\[
C'' = \int_0^\infty (v_n + v_T) \beta^3 v_T^2 \exp (-\beta^2 v_T^2) \, dv_T = \frac{1}{4\beta} \left( \sqrt{\pi} \beta v_n + 2 \right).
\]

\( f_1 \) can be rewritten as

\[
f_1(v_T) = \left( \frac{4\beta^4 v_n v_T^2}{\sqrt{\pi} \beta v_n + 2} + \frac{4\beta^4 v_T^2}{\sqrt{\pi} \beta v_n + 2} \right) \exp (-\beta^2 v_T^2).
\]

Let us apply the following variable changes:

\[
x = \beta v_T \tag{6.0.65}
\]

\[
y = \beta v_n. \tag{6.0.66}
\]

The PDF of transformed variable \( x, f(x) \), reads as:

\[
f(x) = \left[ \left( \frac{\sqrt{\pi} y}{\sqrt{\pi} y + 2} \right)^4 \pi^2 e^{-x^2} + \left( \frac{2}{\sqrt{\pi} y + 2} \right) 2x^3 e^{-x^2} \right].
\]

\( f \) is thus written as a weighted sum of two PDFs: \( \frac{4}{\sqrt{\pi}} x^2 e^{-x^2} \) and \( 2x^3 e^{-x^2} \), where the weights add up to 1. These PDFs can be easily sampled. If

\[
\xi_1 < \frac{2}{\sqrt{\pi} y + 2} \tag{6.0.68}
\]

then we sample \( 2x^3 e^{-x^2} \) using rule \( C49 \) [96]:

\[
x = \sqrt{-\ln (\xi_2 \xi_3)} \tag{6.0.69}
\]

Otherwise we sample \( \frac{4}{\sqrt{\pi}} x^2 e^{-x^2} \) using rule \( C61 \) from [96]:

\[
x = \sqrt{- \ln (\xi_2) - \ln (\xi_3) \cos^2 \left( \frac{\pi}{2} \xi_4 \right)} \tag{6.0.70}
\]
where $\xi_1, \xi_2, \xi_3, \xi_4$ are canonical random numbers. $v_T$ is obtained by simply dividing $x$ by $\beta$ according to the definition of $x$. Earlier we assumed that $\sigma (|v - v_T|)$ is constant. In some cases this approximation was observed to lead to serious bias, typically in LWRs due to the resonance region of $^{238}\text{U}$. A common way to deal with this matter is the Doppler Broadening Rejection Correction (DBRC) method [99], which simply accepts samples obtained the above constant cross section approximation with probability

$$\frac{\sigma (|v - v_T|)}{\sigma_{s,\text{max}}},$$

where $\sigma_{s,\text{max}}$ is the cross section at 0 K. The DBRC can be used on the interval $[\nu_n - 4/\beta, \nu_n + 4\beta]$.

$S(\alpha, \beta, T)$ tables

Scattering kinematics of neutrons with thermal energies can not be treated analytically in general, due to chemical bounding and crystalline effects of target materials. Scattering laws are described by $S(\alpha, \beta, T)$ tables instead. Scattering can be treated as either elastic or inelastic with respect to the scattering system. This means that this distinction is not based on whether a nucleus is left in an excited state, but whether phonons are produced or whether a molecule is excited into higher rotational or vibrational mode. Furthermore, coherent and incoherent models of both elastic and inelastic scattering are distinguished. Coherent elastic data describes neutron scatter on Bragg edges, e.g. on graphite, while incoherent elastic data is used when scattering on solids containing hydrogen, e.g. polyethylene. To sample probability tables yielding the effective cross section, the outgoing energy and scattering cosine, we use the same algorithms presented to sample tabular distributions in Sec. 6 and Sec. 6. In all cases, linear interpolation is used. One exception is coherent elastic scatter, when the effective cross section and the scattering cosine can be derived from parametric data. Given parameters $D_k$, the cross section of scattering on the $k$ – th Bragg edge reads as

$$\sigma_{el}(E) = \frac{D_k}{E},$$

if $E_{Bk} \leq E < E_{Bk+1}$ with $E_{Bk}$ being the energies of Bragg edges. If the incident neutron energy does not exceed $E_{B1}$ then $\sigma_{el}(E) = 0$. The $i$ – th Bragg edge is selected with probability

$$P_i = \frac{D_i}{D_k},$$

and the scattering cosine is then calculated by

$$\mu = 1 - \frac{2E_{B1}}{E}.$$
Unresolved resonances

Some resonances are close enough so that resolving by experiment is not possible. Therefore, maintaining accuracy even in systems where the flux is high in this energy region is achieved via sampling probability tables. The correction only affects cross sections and not secondary energy-angle distributions. Input tables from evaluated nuclear data include cumulative distributions for cross section bands and cross sections or correction factors in each bands. First we choose probability table \( i \) based on incident energy \( E_i < E < E_{i+1} \). Cumulative distribution of this table is sampled yielding cross section band \( j \). Assuming linear interpolation the cross sections are calculated as:

\[
\sigma = \sigma_{i,j} + f(\sigma_{i+1,j} - \sigma_{i,j})
\] (6.0.75)

where \( \sigma_{i,j} \) is the \( j \) – \( th \) band cross section corresponding to incident energy \( E_i \) and

\[
f = \frac{E - E_i}{E_{i+1} - E_i}.
\]

If logarithmic interpolation is used the formula is

\[
\sigma = e^{\log \sigma_{i,j} + f \log \frac{\sigma_{i+1,j}}{\sigma_{i,j}}}
\] (6.0.76)

where

\[
f = \frac{\log \frac{E}{E_i}}{\log \frac{E_{i+1}}{E_i}}.
\]
List of publications


Bibliography


[17] [link].
URL http://awing.reak.bme.hu/GUARDYAN


[47] J. Novák, A. Selle, W. Jarosz, Residual ratio tracking for estimating attenuation in participating media,


A. Khuong, B. Piaud, G. Terrée, Integral formulation of null-collision Monte Carlo algorithms, Journal


[52] J. Sempau, S. J. Wilderman, A. F. Bielajew, DPM, a fast, accurate Monte Carlo code optimized for
photon and electron radiotherapy treatment planning dose calculations, Physics in Medicine & Biology

IEEE Nuclear Science Symposium Conference Record (NSS/MIC), Institute of Electrical and Electronics
Engineers (IEEE), 2009, pp. 4081–4084. doi:10.1109/nssmic.2009.5402382.

[54] W. A. Coleman, Mathematical verification of a certain Monte Carlo sampling technique and applications

[55] F. B. Brown, W. R. Martin, Direct sampling of Monte Carlo flight paths in media with continuously


URL https://doi.org/10.13182/NSE73-A26575


