A GPU based direct Monte Carlo simulation of time dependence in nuclear reactors

Balazs Molnar[a,*], Gabor Tolnai[a], David Legrady[a]

[a]*Budapest University of Technology, Budapest, Muegyetem rkp. 3-9,H-1111, Hungary*

Abstract

A novel 3D Monte Carlo (MC) neutron transport code, GUARDYAN, was developed to simulate direct time dependence in nuclear reactors. GUARDYAN (GpU Assisted Reactor DYnamic ANalysis) addresses the huge computational need by exploiting massive parallelism available on modern Graphics Processing Units (GPUs). While the code is still under development, transient analysis on large scale problems is already obtainable. The implementation is verified via comparison of differential and integral quantities to MCNP6 results, including several criticality safety benchmarks. Unlike most conventional MC codes GUARDYAN is intentionally designed for time dependent calculations supporting parallel scalability on state-of-the-art high performance computing platforms. The methodology of transport simulation thus differs in many aspects: generation-by-generation tracking is replaced by a time step method; branching of neutron histories, neutron banking is eliminated by statistical weight manipulations; a robust delayed neutron treatment is implemented. These concepts, along with advanced acceleration techniques for improving the performance of point-in-cell search routine and the delta tracking method, resulted in an efficient MC tool that seems to outperform existing methods for kinetic MC simulation. Transient analysis was performed on an LWR core demonstrating that simulation of one second of a transient requires around 50 hours on a single GeForce GTX 1080 GPU. The power evolution produced by GUARDYAN during this transient was also compared to experimental data; remarkably close agreement was found despite the uncertainties in the MC model.

Highlights

- Direct time dependent Monte Carlo neutron transport tool using GPUs
- Whole-core transient analysis and comparison with kinetic measurement data
- Robust prompt and delayed neutron treatment favoring GPU characteristics
- Advanced techniques for cell search and delta-tracking
- Comparison of event-based and history-based algorithms on the GPU

Keywords: Time dependent Monte Carlo, GPU, transient analysis, delayed neutron, event-based

*Corresponding author
Email address: molnar.balazs@reak.bme.hu (Balazs Molnar)

1. Introduction

GUARDYAN (GpU Assisted Reactor DYnamic ANalysis) is a 3D continuous energy Monte Carlo (MC) code capable of simulating direct time dependence in multiplying systems using Graphics Processing Units (GPUs). Capabilities of GPUs have grown exponentially in the last years, a single commercially available GPU possesses performance of 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 [1], [2] and [3]. A comparison of CPU and GPU implementation of a criticality calculation showed 7 times speedup in favor of the GPU [4]. Due to the contrast between GPU and CPU hardware however, it would be prohibitively expensive and unreasonable to rewrite existing MC codes for GPUs, as the entire code would have to be restructured [5]. As a novel MC code, GUARDYAN was implemented with the purpose of efficient utilization of the GPU in mind, that resulted not only in structural differences compared to conventional MC codes, but various unique approaches and algorithms that may be of particular interest for MC developers.

Beside the challenges imposed by GPU hardware, the other factor calling for novel concepts is time dependence. The feasibility of kinetic MC simulation of nuclear reactors has been questioned before due to the immense computational cost required to perform these calculations. Now, state-of-the-art computing platforms are capable to satisfy the computational needs, and significant efforts have already been put into the development of dynamic modules for some production-level codes like TRIPOLI-4 [6], OPENMC [7] and SERPENT [8]. Parallel processing is now self-evident in these codes, but is achieved by implementations scaling well for multicore nodes or clusters of computers, and not GPUs. Most of these dynamic solvers are still subject to strong approximations to geometry or delayed neutron treatment. Delayed neutron treatment is still simplified in an extension of the OpenMC code [9], as is the case with the dynamic module of SERPENT 2 [10] [11]. The G4-STORK code [12] built on the Geant4 software also 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 [13] or diffusion approximations [14] are considered to reduce the computational burden. A different approach is described in an ongoing research [15], where the difference between time scales is addressed by solving the frequency domain transport equation by MC.

Unlike the above methods, the DMC method [16] by Sjenitzer and Hoogenboom describes an approximation free methodology for the direct time dependent simulation of nuclear reactors, making it the most relevant research with respect to the subject of this paper. The DMC successfully addresses several issues that are typically neglected in static MC, but are important in time dependent MC (TDMC). These concepts are fundamental for GUARDYAN as well, therefore they are summarized in the following points:

- The time shift of delayed neutrons has a significant impact on the time evolution of a neutron population. While static MC calculations neglect this delay, dynamic simulations must handle both prompt and delayed neutrons. This is non-trivial due to the several orders of magnitude difference between prompt

and delayed neutron lifetime. In DMC delayed neutron precursor sampling and forced precursor decay [17] was successfully applied to account for the time scale mismatch [18].

- Analog treatment of branching processes in multiplying systems may cause the neutron population to evolve without bounds. This calls for a population control of prompt neutrons, addressed by the improved branchless method [19] in DMC.

- Time variable must be explicitly present in time dependent simulations in order to allow system properties to vary over time. The DMC suggests the introduction of time steps to solve this problem.

- Initial conditions are needed for a time dependent simulation, both live neutron and precursor distribution must be obtained to launch the calculation. In DMC initial conditions were obtained assuming the transient starts from steady-state, thus both neutron and precursor source can be calculated.

The DMC was later coupled with thermal hydraulic feedback [20], and recently, a time dependent variant of TRIPOLI-4 was published based on the DMC method [21]. Currently, to our best knowledge, this is the only existing pure MC tool for whole core transient analysis that does not ignore delayed neutron treatment, thus the only relevant source for comparison of performance of GUARDYAN. Reported runtime figures are an order of magnitude higher than those experienced with GUARDYAN, but fairness of this comparison is very questionable and this calls for a time dependent benchmark to develop to compare both calculated values and running times. This issue will be addressed later in this paper (cf. Sec 3.4).

Several techniques are adopted from the DMC in this paper, however, many of them are revised and adjusted to the GPU methodology. These techniques include the population control with the exact conservation of the number of particles detailed in Sec. 2.1.2, the robust delayed neutron sampling of Sec. 2.1.3 and the construction of initial conditions (cf. 2.1.4). Apart from developing a unique methodology for an efficient implementation of a time dependent MC neutron transport on the GPU, the novelty of this paper lies in the introduction of acceleration methods, e.g. supervoxel delta tracking (2.2.3) and improved cell search (2.2.2), in establishing a performance analysis framework for MC kinetic calculations (3.3) and in the comparison of kinetic simulation to experimental data (3.4).

2. Methodology

2.1. Simulation flow

Calculation flow of GUARDYAN is principally different from conventional MC codes. Differences are mainly motivated by two factors: time dependence and GPU programming. In GUARDYAN the time variable is explicitly present since geometry changes and thermal feedbacks must be routinely admitted, also, the time shift of delayed neutrons must be sampled. The GPU hardware dictates many other approaches, e.g. tracking neutrons from time step to time step instead of generation-by-generation, the elimination of neutron banking, or the use of delta-tracking. To ensure peak parallel performance on the GPU memory management

and occupancy issues must be understood. Parallelization is performed by the independent working units of the GPU called threads. In contrast to CPUs, threads can access local register memory of a limited size but are more numerous, furthermore, the programmer is responsible for assigning tasks to threads. Using more register memory essentially results in less parallelization. High occupancy means that GPU resources are fully utilized, all threads are constantly working. 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 only obtained only when the slowest thread finishes. Therefore it is paramount to distribute tasks evenly, else 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 translates to that branching of neutron histories at fission, splitting/Russian roulette, and any other processes involving various computational cost for different threads are strictly forbidden, or should be minimized.

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 feedback or geometry changes. Without occasional interruption of the simulation however, these changes would either require on-the-fly calculation of time-dependent variables or more memory space for storing pre-calculated data, e.g. temperature-dependent cross sections, both option incompatible with the typical workflow and memory allocation of GPUs. The problem can be circumvented by introducing time steps and assuming system properties to be constant during a step.

GUARDYAN splits the total time span of the simulation into short intervals. A GPU function termed the kernel function is called periodically at time boundaries to simultaneously process the parts of neutron histories falling in the upcoming time step. Particles are always synchronized at boundaries. Modification of geometry or thermal feedback effects can only be considered at time boundaries. While this time step methodology is mainly motivated by the change in the physics of the system it also gives opportunity to observe and interfere with a population that is not dispersed in time or to rearrange computational resources when any changes to the system is made (see e.g. the combing method in Sec. 2.2.1)

2.1.2. Prompt neutron treatment and population control

While the reactor is in a sub– or supercritical state the neutron flux decreases/increases exponentially, prompting the number of MC samples in a simulation to be low or unmanageably high. This rapid change in neutron population under transient conditions calls for population control in Monte Carlo calculations. From the GPU perspective, it is also essential to limit the change in the number of neutrons, as the GPU is most efficient for fixed sized streams of data. 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 be unreasonable in any case due to the limited size of local memory

per thread. Without population control, a GPU based TDMC simulation would be unfeasible or at least inefficient due to register memory issue and thread divergence.

In GUARDYAN, we work with a fixed size of particle array, containing neutrons and delayed neutron precursors (see Fig. 2), using nonanalog techniques to enforce a constant population in time. Constant population is achieved by treating neutron multiplication (fission and (n, xn) reactions) and capture events with changing the statistical weight of a neutron. A simple method to ensure that a neutron history never terminates on absorption is survival biasing. Instead of simulating capture, the particle weight is multiplied with the probability of survival at each collision:

$$w' = w\left(1 - \frac{\Sigma_c}{\Sigma_t}\right). \tag{2.1}$$

To prevent branching our choice was to use the improved branchless method [16] in GUARDYAN, which treats neutron multiplication by weight change, and alters the probability of fission to yield better statistics. In this framework, GPU memory allocation issues suggested to store cross section data in a $\nu\Sigma$ format, redefining ν the following way: $\nu = 0$ for capture reaction (MT=101), $\nu = 1$ for non-multiplying reactions, $\nu = x$ for (n, xn) reactions, and $\nu = \overline{\nu}$ for fission reactions. GUARDYAN samples reaction of type i with probability of

$$P_i = \frac{(\nu\Sigma)_i}{\sum_j (\nu\Sigma)_j} \tag{2.2}$$

and the weight change at each collision is

$$w' = w\frac{\sum_j (\nu\Sigma)_j}{\Sigma_t}. \tag{2.3}$$

This collision sampling procedure is a slightly modified and generalized version of the improved branchless method which has been proved to be unbiased in [16].

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. Therefore a variance reduction technique called the combing method [23] is used at time boundaries. Unlike the widely spread 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. Since it handles a batch of particles together, it enables the use of population characteristics to set the surviving average weight following the power evolution of the system. A detailed description of the combing method implemented in GUARDYAN will be given in Sec. 2.2.1.

Employing population control was found to be mandatory in GUARDYAN, both the improved branchless method and combing are fundamental regarding the stability of a transient simulation. Without these methods GUARDYAN was unable to produce reliable estimates even for very short time periods, demonstrated in Fig. 1. The model in this example was a subcritical homogeneous cuboid with two energy groups, in which the branchless method with analog fission probability and the improved branchless method were investigated. We see that although the analog simulation is obviously unbiased, it gives correct results only for the first

two milliseconds. Then, without indicating a serious deficit in statistics it consistently underestimates the true solution. This is a result of wild fluctuations in particle weights, closely related to the phenomenon often referred to as the critical catastrophe. In the example, neutrons with large weights have probably leaked out of the system, leaving only neutrons with small weights unable to compensate for the neutron loss. When forced fission is applied, more fission events are sampled resulting in a more even distribution of production and loss among the particles.

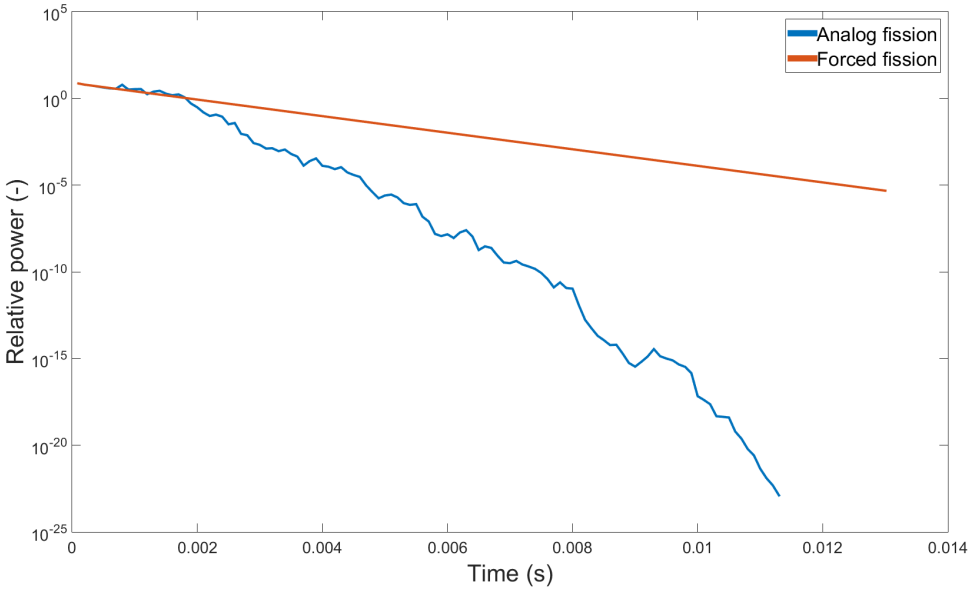

Figure 1: MC estimation of power evolution with analog fission vs. forced fission. The example model was a subcritical homogeneous cuboid with two energy groups. Combing was used for population control in both cases, but the analog simulation used the analog fission probability, while forced fission used Eqs. 2.2 and 2.3.

2.1.3. Delayed neutron treatment

While prompt neutrons have an average lifetime in the order of $10\mu s$ (for thermal systems), the expected lifetime of a delayed neutron can range from $0.1s$ to $100s$. The difference is several order of magnitudes, thus analog simulation of delayed neutrons is clearly unfeasible in a time dependent 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. 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

$$C_i(\vec{r}) = \frac{\beta_i}{\lambda_i \Lambda} n(\vec{r}) \tag{2.4}$$

where C_i stands for the concentration of precursors from the $i-th$ family, β_i and λ_i are the delayed neutron fraction and decay constant respectively, Λ is the mean generation time, and n denotes the density of live neutrons. With Λ 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 number 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, then they are combed to the average weight of prompt neutrons. In a supercritical state prompt neutron samples are numerous and can produce stable power evolution, delayed neutrons will be less well represented in this case. In a subcritical state the average weight of prompt neutrons decreases, but many delayed neutron samples are produced, ensuring that live neutrons are never undersampled. 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 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. 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 on Fig. 2. 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 interval (denoted by n'), that is determined by the distance to collision sampling routine. Precursors are denoted by C.

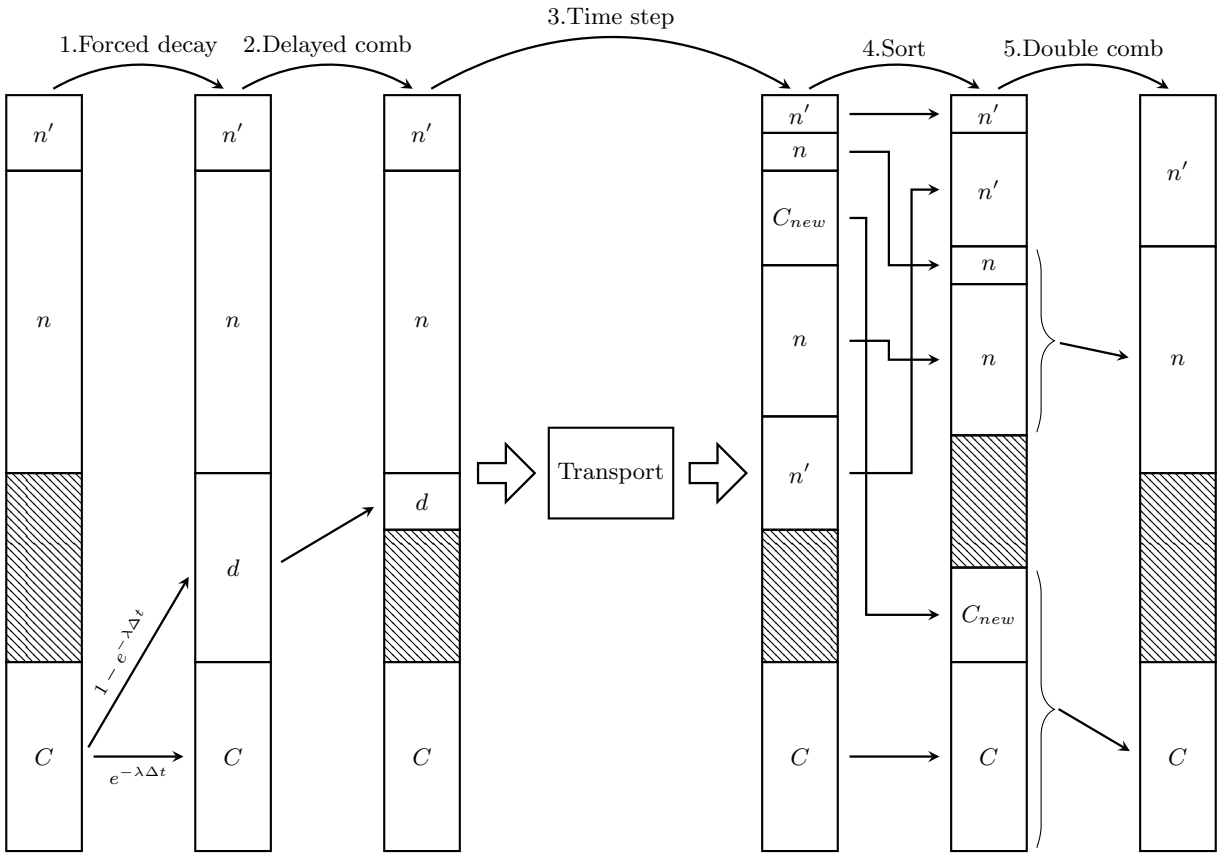

Figure 2: Operations on the GPU particle array

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 [17], filling up the empty space in the particle array. In GUARDYAN a precursor particle is always split into a decaying and a surviving part at the beginning of each time step, yielding a delayed neutron with weight $\left(1-e^{-\lambda\Delta t}\right)$ times the precursor weight, while a fraction of $e^{-\lambda\Delta t}$ of its weight continues as a precursor. Forced decay was found to be a compulsory element based on our experience, as the method always produces enough samples for both delayed neutrons for the upcoming time step, and for future contribution.
2. Delayed neutrons are combed to the average weight of prompt neutrons with the simple comb described in Sec.2.2.1.
3. Simulation of a time step is performed on live neutrons (on stacks n and d on Fig. 2). 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 β, 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 first, 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. Although the prompt source has converged, several time steps are simulated to obtain the precursor source as well. Precursor samples are generated from neutrons at the end of time steps, as long as the precursor array is incomplete. At the end of a time step, neutrons are stopped, and precursors from randomly selected families are created from neutrons located in fissile material with weight

$$w_{precursor} = w_{prompt} \cdot K \frac{\beta_i \nu \Sigma_f}{\lambda_i} v \tag{2.5}$$

where K is the number of precursor groups, β_i is the fraction of delayed neutrons for the chosen family, ν is the fission neutron yield, Σ_f is the fission cross section, v is the speed of the prompt neutron and λ_i is the decay constant of the precursor. This method is in line with the source generation of the DMC [16].

2.1.5. Woodcock (delta-) tracking

GUARDYAN uses the Woodcock method (delta tracking) for sampling distance between collisions. Surface to surface tracking is not particularly well suited for GPU architecture, the method essentially fails on two levels. First, the calculation of ray-surface intersections 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.

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. The original delta tracking algorithm described by Woodcock [24] 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 Σ_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 Σ_{real}/Σ_m and with the virtual material with probability $\Sigma_{virtual}/\Sigma_m$ at a collision site. Interaction with the virtual material is defined as a delta-scatter, when neither direction nor energy is altered.

Possible modifications of the algorithm have been described by [25], [26], [27], [28], and [29]. Recent research indicates that the algorithm can be optimized via tuning two independent parameters: the majorant cross section and the probability of choosing a real collision [30]. In GUARDYAN we follow the same notations as in [30]: instead of Σ_m we introduce the sampling cross section $\Sigma_{samp}(P)$ 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

$$f_{BW}(s) = \Sigma_{samp}(\vec{r} + \vec{\Omega}s, E, \vec{\Omega}) exp\left(-\int_0^s \Sigma_{samp}(\vec{r} + \vec{\Omega}t, E, \vec{\Omega})dt\right) \tag{2.6}$$

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

$$w' = w\frac{\Sigma(P')}{q(P')\Sigma_{samp}(P')} \tag{2.7}$$

if a true collision happened, and

$$w' = w\frac{1 - \frac{\Sigma(P')}{\Sigma_{samp}(P')}}{1 - q(P')} \tag{2.8}$$

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 Σ_{samp} and q can be chosen arbitrarily (taking the above restrictions into account), and the algorithm is still unbiased (for proof see [30]). Without the loss of generality we wrote $\Sigma_{samp}(P)$, but most of the times we want Σ_{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. 2.6 with the inverse cumulative method, the exact algorithm will be detailed in Sec 2.2.3.

2.1.6. Geometry, cross sections and interaction physics

The input logic of GUARDYAN is similar to the logic followed by production level codes MCNP [31], Serpent [8] or OpenMC [7]. 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 cross section data in an ACE (A Compact ENDF) format, which is generated by NJOY from the ENDF-B-VII.1 cross section library. Isotopic total and majorant cross sections are pre-calculated on a unionized energy grid. Energy grid search is implemented with a binary search. Reaction sampling begins with the selection of the interacting isotope based on the ratio of total cross sections of the isotopes present in the material. The code assumes the free-gas model for the motion of nuclei when correction to

the scattering cross section must be accounted for. Interactions are modeled similarly to MCNP, OpenMC or Serpent, post-collision energy and angle distributions are sampled according to ACE laws. $S(\alpha, \beta)$ tables are considered for scattering of thermal neutrons on bounded atoms; to deal with unresolved resonances GUARDYAN uses the table probability method.

To find the location of a certain collision site, GUARDYAN uses an improved cell search routine, which will be described in Sec. 2.2.2 due to its significance of accelerating the code.

2.2. Variance reduction and acceleration methods

2.2.1. Combing

The combing method introduced by T. E. Booth [23] provides an alternative variance reduction method to the weight window technique. 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. The fact that the output data is of a pre-determined size makes this method ideal for application on GPUs. 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

$$w' = \frac{1}{M} \sum_{i=1}^{N} w_i \tag{2.9}$$

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:

$$t_m = \rho w' + (m-1)w' \tag{2.10}$$

where ρ is a random number from $(0, 1)$ and $m = 1...M$. 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.

To improve the performance of combing we implemented the importance weighted version of the combing method. In principle 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. The weight assigned to a post-combed particle will be

$$w_i' = \frac{\sum_{i=1}^{N} w_i I_i}{M I_i}. \tag{2.11}$$

Importance of a neutron is defined by the probability of causing a fission reaction (contributing to total power). The importance function is tallied in logarithmically placed energy groups with equal spatial resolution, but disregarding directional dependence. The probability of yielding a fission event from a certain

space-energy bin is estimated by a forward MC calculation. Several neutron histories are generated which are terminated on 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. Usually, such direct calculation of the importance function would be prohibitive in computing cost. For direct time dependent MC, steady-state adjoint computing time diminishes in contrast to the computational needs of the whole time dependent calculation.

2.2.2. Improved cell search

This method targets the acceleration of finding the material type at collision sites. The computational cost of finding this material increases with the complexity of the system, and can also depend on the implementation of the input. We observed that even in the case of not so complicated geometries like the Training Reactor at Budapest University of Technology and Economics, serious effort was required to perform this calculation. 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.

2.2.3. Majorant mesh

Distance to collision sampling is realized by a modified version of the Woodcock method based on previously developed framework [30]. To improve the performance of the path length selection in GUARDYAN, super-voxels are introduced, the term adopted from the field of computer graphics [32]. A cartesian mesh is superposed on the system, partitioning the geometry to super-voxels. The local majorant is determined in each of these voxels, as being greater than any other cross sections in that particular voxel. This procedure basically constructs a majorant mesh. Now, the localized heavy absorber problem (cf. Sec. 2.1.5) affects only the voxel containing the absorber. Additionally Σ_{samp} becomes piece-wise constant, thus sampling Eq. 2.6 is extremely simple. By the inverse cumulative method the equation to be solved reads as

$$-ln(\xi) = \sum_{i=1}^{n} \Sigma_{samp}^{(i)} s^{(i)} \tag{2.12}$$

where $\Sigma_{samp}^{(i)}$ and $s^{(i)}$ are the majorant cross section and traversed path length in the $i-th$ visited voxel, and ξ is a canonical random number. $s^{(i)}$ is calculated by the 3D–DDA algorithm [33] in GUARDYAN, as a

fast way of determining ray-box intersections.

2.2.4. Event-based tracking

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. In GUARDYAN the event-based version of the code was also implemented targeting the thread divergence issue. 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 the thread divergence on this level. Preliminary results [34] showed that although the event-based strategy seems to be better suited for GPU architecture, a range of various circumstances can alter the actual performance gain over the history-based method. These circumstances include the size of neutron population (the more the better for event-based calculation), the isotopes present in the region (the variance in the complexity of corresponding sampling laws), and most trivially the implementation of certain algorithms. As an example, in GUARDYAN we found that due to the inefficient implementation of the point-in-cell search routine the event-based method performed poorer. Considering recent developments of the code, event-based tracking in GUARDYAN was revisited, the results will be given in Sec. 3.3.2.

3. Results

3.1. Verification of interaction physics - a differential comparison

Due to GUARDYAN being a novel time dependent Monte Carlo code, the correct implementation of physics was challenged by comparing GUARDYAN to the general purpose Monte Carlo code MCNP. In this code-to-code comparison differential quantities were analyzed, overall 445 000 data points were compared. 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. One example case can be seen in Fig 3. Erroneous implementation of interaction physics was identified by calculating the fraction of data points outside 1,2 and 3 σ margin, and was corrected if large differences were observed.

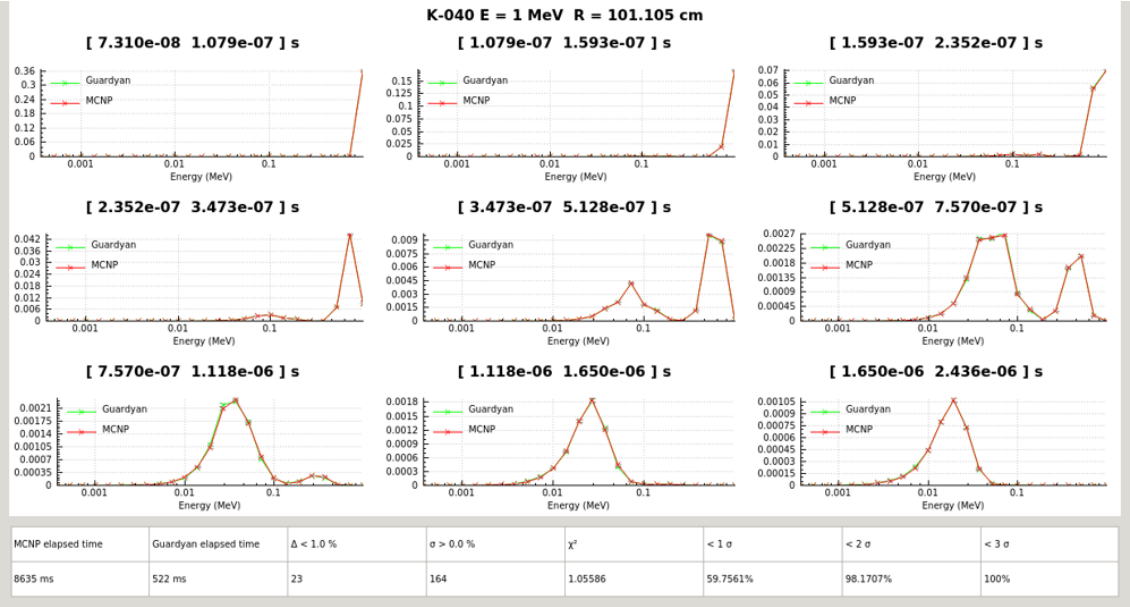

Figure 3: Example calculation of the verification work comparing the time evolution of fluence spectrum to MCNP results. In this figure only one example is illustrated by simulation results for a homogeneous sphere with radius of 101.105cm containing a single isotope of ^{40}K. The whole verification work consists of 412 figures with respect to 412 isotopes

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. A histogram of this measure is plotted in Fig. 4. Values around one imply excellent agreement; values above 2 are rare confirming that the two implementations agree.

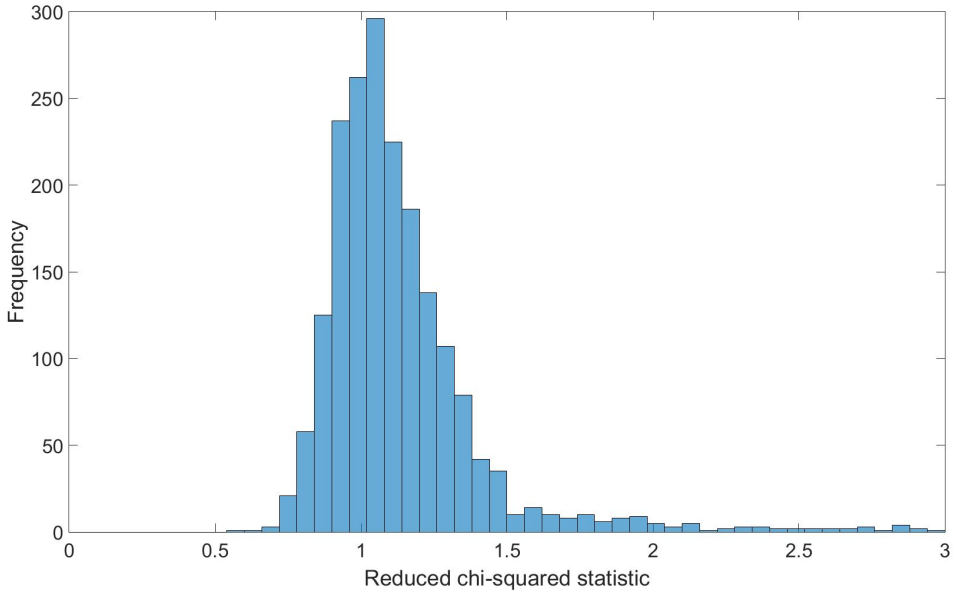

Figure 4: Verification of the interaction physics of GUARDYAN by calculating chi-square statistics [35]

3.2. Criticality benchmarks - an 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 [36], 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 1 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 [36] as well, MCNP6 cycles were regenerated using the same cross section library (ENDF/B-VII.1) that GUARDYAN uses.

benchmark	program	k_d	σ_r	k_{eff}	σ_r	$t(s)$	$FoM(10^3 \frac{1}{s})$
heu-met-fast-001	MCNP6	–	–	0.99981	0.00005	602	664
	GUARDYAN	1.000065	0.00023	0.99981	0.00009	30	4507
pu-met-fast-002	MCNP6	–	–	1.00010	0.00005	555	721
	GUARDYAN	1.000036	0.000077	1.00006	0.00010	23	4527
heu-comp-inter-003	MCNP6	–	–	1.00792	0.00006	1993	139
	GUARDYAN	1.001485	0.000099	1.00809	0.00007	426	429
u233-sol-inter-001	MCNP6	–	–	0.98543	0.00008	1426	110
	GUARDYAN	0.999707	0.00058	0.98506	0.00009	236	501
u233-sol-therm-001	MCNP6	–	–	1.00119	0.00004	7665	82
	GUARDYAN	1.000100	0.00013	1.00068	0.00006	1527	188
leu-sol-therm-001	MCNP6	–	–	1.01180	0.00006	2938	95
	GUARDYAN	1.004618	0.000061	1.01103	0.00006	622	432
leu-sol-therm-002	MCNP6	–	–	0.99983	0.00004	6455	97
	GUARDYAN	1.000047	0.0003	0.99954	0.00005	3881	92
leu-comp-therm-008	MCNP6	–	–	0.99972	0.00005	6311	63
	GUARDYAN	1.000010	0.00031	0.99945	0.00006	1085	240

Table 1: Criticality benchmarks from [36] with GUARDYAN and MCNP6
GUARDYAN was run on an Nvidia GeForce GTX 1080 GPU, with 2^{21} neutrons for 100 generations, k_{eff} was calculated for the last 80 generations
MCNP6 was run on an Intel Xeon E5-1650 CPU using 12 threads, with 2^{14} neutrons for 12800 generations, k_{eff} was calculated for the last 10240 generations
The same cross section library, ENDF-B-VII.1, was used for both codes

While the results summarized in Table 1 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, a dynamic quantity was also calculated, defined by

$$k_d = \frac{\sum_i P_i^{\Delta t}}{\sum_i L_i^{\Delta t}} \tag{3.1}$$

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 in time. Indeed, Table 1 shows that when k_{eff} is very close to 1, GUARDYAN also provides k_d around 1. Deviations between k_{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_{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 [37]. 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 [38] or [37]).

Runtime $t(s)$ in Table 1 shows total execution time, σ_r denotes the relative error. The efficiency measure, FoM (Figure of Merit), is calculated by

$$FoM = \frac{1}{\sigma_r^2 t}. \tag{3.2}$$

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.

3.3. Performance analysis of variance reduction and acceleration methods

Variance analysis of time dependent MC solutions raises fundamental questions concerning the issue of correlated samples. Let us suppose for instance, that we want to measure the deviation of MC power evolution from the true solution. Clearly, the individual tallies from consecutive time steps are not independent. Simply put, under equilibrium conditions, the probability of total power to rise, given that the power increased in the previous time step, is higher than the probability of power to drop. This is demonstrated with GUARDYAN shown in Fig. 5. The true solution (black) is estimated as the average of 18 individual runs. It makes sense to distinguish low and high frequency patterns in the time series. In Fig. 5 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 a variance analysis this means that the MC uncertainty of estimates in individual time steps tell little about the deviation of the MC result from the true solution.

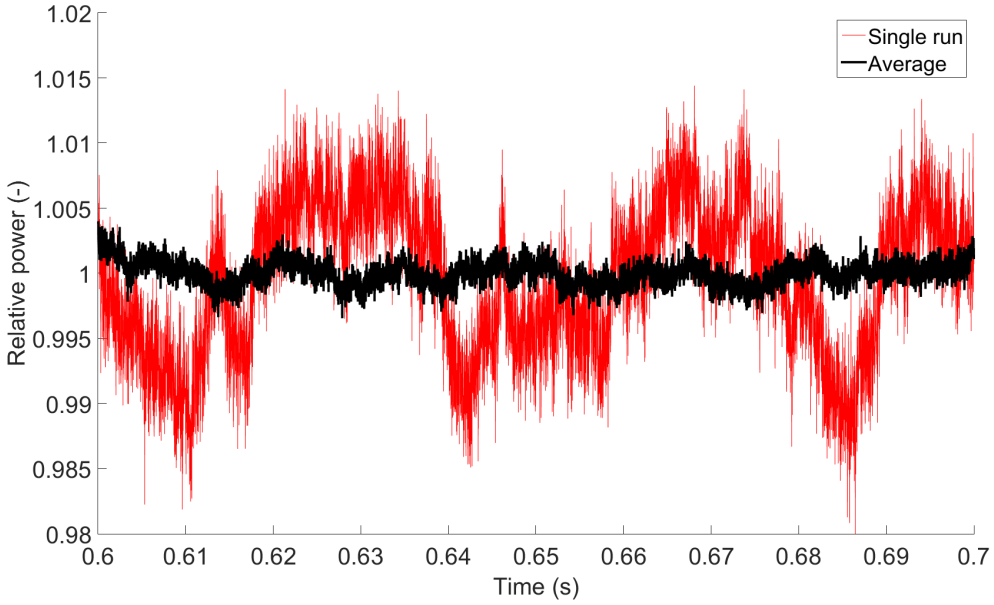

Figure 5: Power evolution of a single run vs. average power estimated from 18 independent runs

To do a variance analysis correctly, one should consider all contributions from a single starter (including contributions from its progenies) as one individual score. Then, all scores would be truly independent from each other, allowing the simulation error to be calculated without taking correlations into account. The problem is that sooner or later all live neutrons in a time step will descend from one single starter, essentially resulting in one score of reactor power for that time step. This phenomenon is well-known in criticality calculations as well, and is called clustering. Recently, more attention was devoted to this issue in the MC field. It was proved for example, that in static eigenvalue calculations using the method of successive generations, after many cycles all neutrons will be progenies of a single starter almost surely [39]. However, the effect of clustering in time dependent MC calculations is unclear, and lies out of the scope of this paper. In this section we present a workaround to extract a meaningful comparison between power evolutions estimated by various methods.

3.3.1. Importance weighted combing and time step size

Our main goal is to evaluate the efficiency of variously parametrized simulations or different variance reduction techniques based on a measure that correctly describes the deviation of MC power evolution from the true solution. We are particularly interested in how the size of time steps influences this measure, or how the importance weighted comb performs opposed to the simple comb applied in GUARDYAN. In this analysis combing (cf. Sec. 2.2.1) was only considered for neutrons and not precursors. We analyzed the effect of using different time steps in GUARDYAN, repeating the same simulation using $10^{-4}s$, $10^{-5}s$ and $10^{-6}s$ long time steps. Both combs were evaluated at all step sizes, and additionally, in three different scenarios: the Training Reactor at critical rod positions, all rods drawn out, and all rods fully inserted. Our main

interest was to determine whether the optimal step size depends strongly on the reactor state and to assess the performance gain from using a time independent importance, considering that the importance function was constructed for the critical case only. The latter information would be crucial in view of future research, as the cost of generating a time dependent adjoint function would be excessive.

Let us first discuss the matter of finding a possible way to conclusively compare different simulation methods, e.g. simple comb and importance weighted comb. We previously argued, that the deviation from the true solution is poorly estimated by the MC uncertainty of total power from individual time steps due to low frequency patterns in the power evolution affected by correlations. 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 6. In the simulation using Method #1 (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 on this particular interval. Taking another time interval, we may found that the red curve is closer to the reference solution, as is the case in the last 10 milliseconds in Fig. 6. 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, however, we chose a different approach to describe this error.

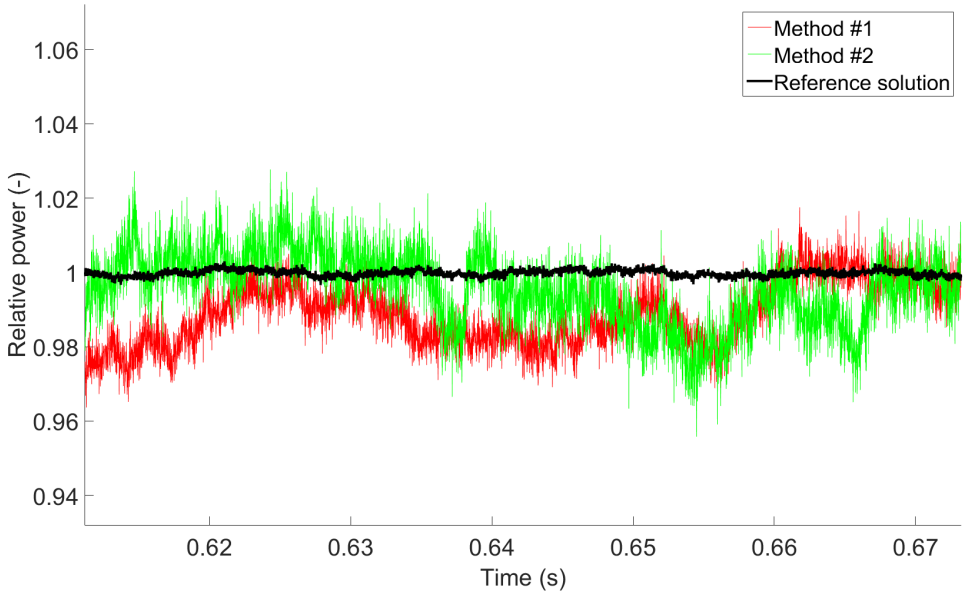

Figure 6: Illustration of the problem of comparing single runs. We see that the MC uncertainty of the red curve is far lower, but is further from the reference solution compared to the green line in this particular time period. On the long run, both estimate the total power correctly.

Initially, the true solution was estimated by averaging individual runs with 18 different random seeds, shown

with black in Fig. 5. Then, instead of a single run, the average of 5 runs with different seeds was taken in each case of the evaluated methods. These averages were than tested against the estimated true solution by calculating the mean squared error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(P(t_i) - \overline{P}(t_i)\right)^2 \tag{3.3}$$

that is the average of squared differences between the solution of the investigated method (P) and the reference solution ($\overline{P}$) over $N = 10000$ time intervals during a period of $0.1s$. Simulations were performed in the MC model of the Training Reactor, shown in Fig. 11. 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. 2.2.1, 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 7, 8 and 9 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.

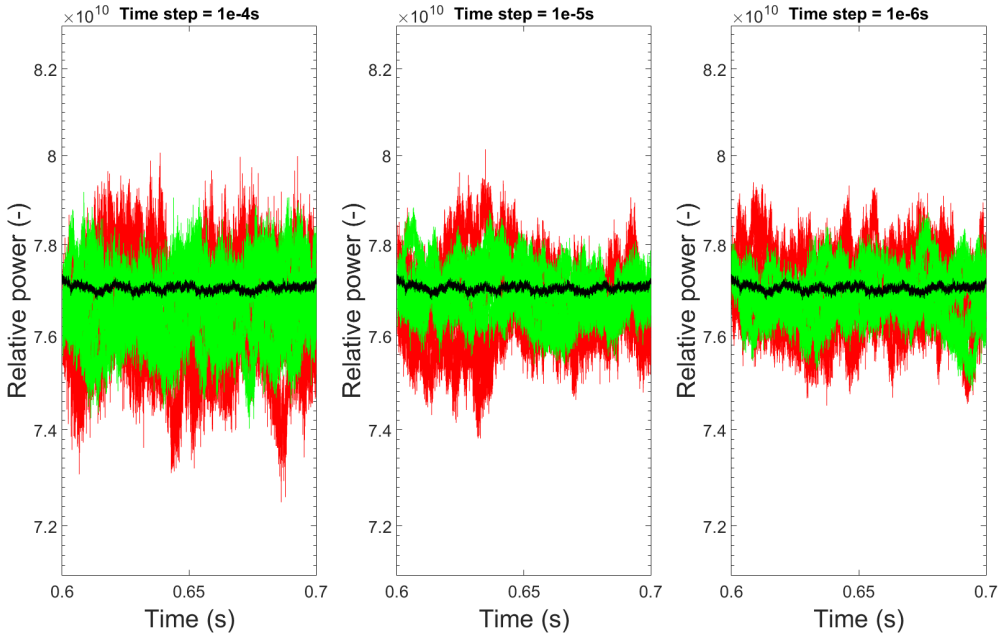

Figure 7: Individual runs on the critical system

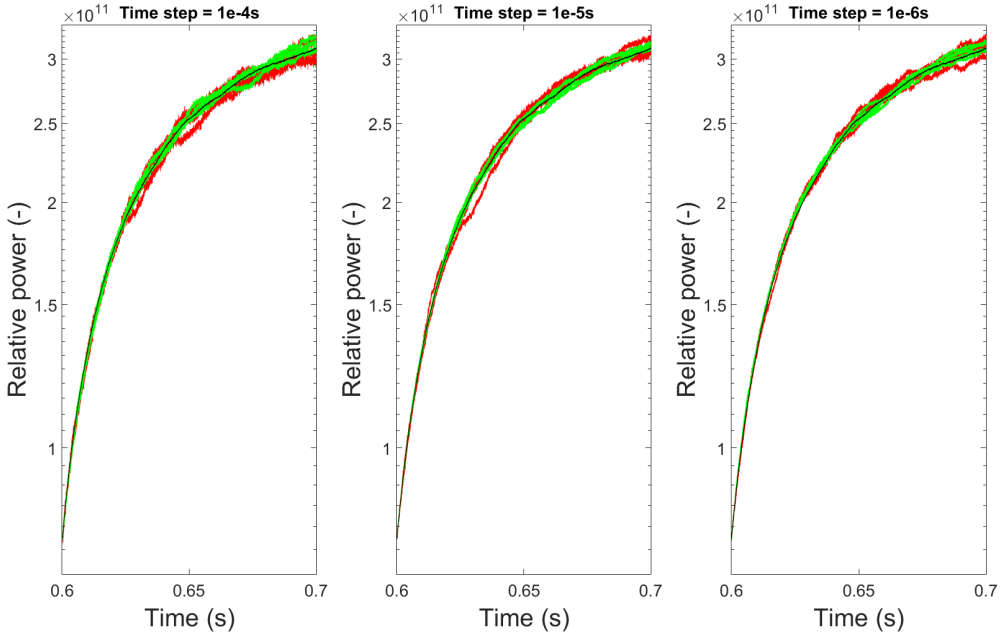

Figure 8: Individual runs on the supercritical system

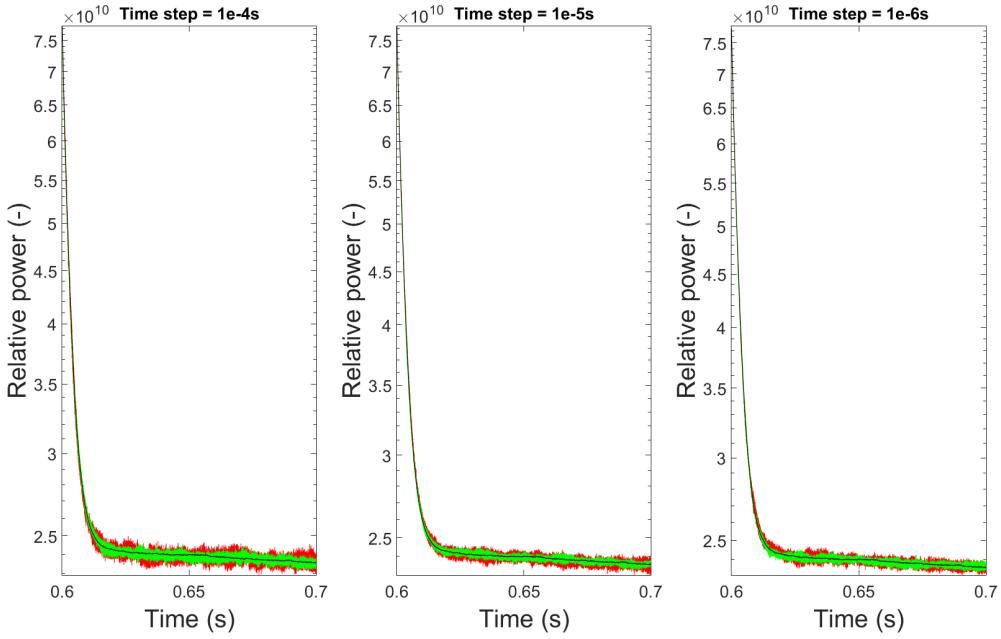

Figure 9: Individual runs on the subcritical system

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 2, where the normalized root-mean-squared-error (NRMSE) is shown which is related to the MSE via

$$NRMSE = \frac{\sqrt{MSE}}{\frac{1}{N}\sum_{i=1}^{N}\overline{P}(t_i)}. \tag{3.4}$$

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 3 do not include the time to generate the importance. The Figure of Merit (FoM) values in Table 4 is calculated by

$$FoM = \frac{1}{NRMSE^2 T} \tag{3.5}$$

where T stands for the runtime.

		Time step size (s)		
		1e-4	1e-5	1e-6
Critical	Simple comb	0.981	0.691	0.400
	Importance weighted comb	0.515	0.354	0.334
Subcritical	Simple comb	0.635	0.348	0.352
	Importance weighted comb	0.436	0.258	0.243
Supercritical	Simple comb	1.024	0.687	0.719
	Importance weighted comb	0.757	0.528	0.619

Table 2: Normalized root-mean-square error (NRMSE) in percents

		Time step size (s)		
		1e-4	1e-5	1e-6
Critical	Simple comb	3.71	3.89	9.30
	Importance weighted comb	5.04	5.27	11.34
Subcritical	Simple comb	4.30	4.24	10.32
	Importance weighted comb	6.06	5.92	12.57
Supercritical	Simple comb	3.30	3.64	8.55
	Importance weighted comb	4.38	4.96	10.48

Table 3: Runtime in hours

		Time step size (s)		
		1e-4	1e-5	1e-6
Critical	Simple comb	2796	5387	6723
	Importance weighted comb	7465	15180	7892
Subcritical	Simple comb	5766	19477	7837
	Importance weighted comb	8684	25470	13432
Supercritical	Simple comb	2893	5815	2263
	Importance weighted comb	3988	7229	2493

Table 4: FoM in 1/h units

Table 2 shows that variance is greatly reduced by decreasing time step size from $10^{-4}s$ to $10^{-5}s$, but further improvement can not be achieved by choosing an even smaller time step size of $10^{-6}s$. If we consider runtimes (Table 3), 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. 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.

3.3.2. Acceleration methods

Previous investigation on potential optimization of GUARDYAN identified the point-in-cell search routine and collision sampling as computational units taking up the largest part of runtime. In [34], we reported that the GPU kernel in which free path and collision sampling is implemented, was two orders of magnitude slower than any other kernel. This was concluded by profiling the application, shown in Fig. 10. Obviously, this means that acceleration of the code must target more efficient algorithms for these particular routines, as attempting to optimize any other part of the code would have little effect on overall runtime. This was also confirmed by benchmarking the event-based version of the code, i.e. improvement in runtime was not observed, despite the thread divergence issue being successfully addressed [34].

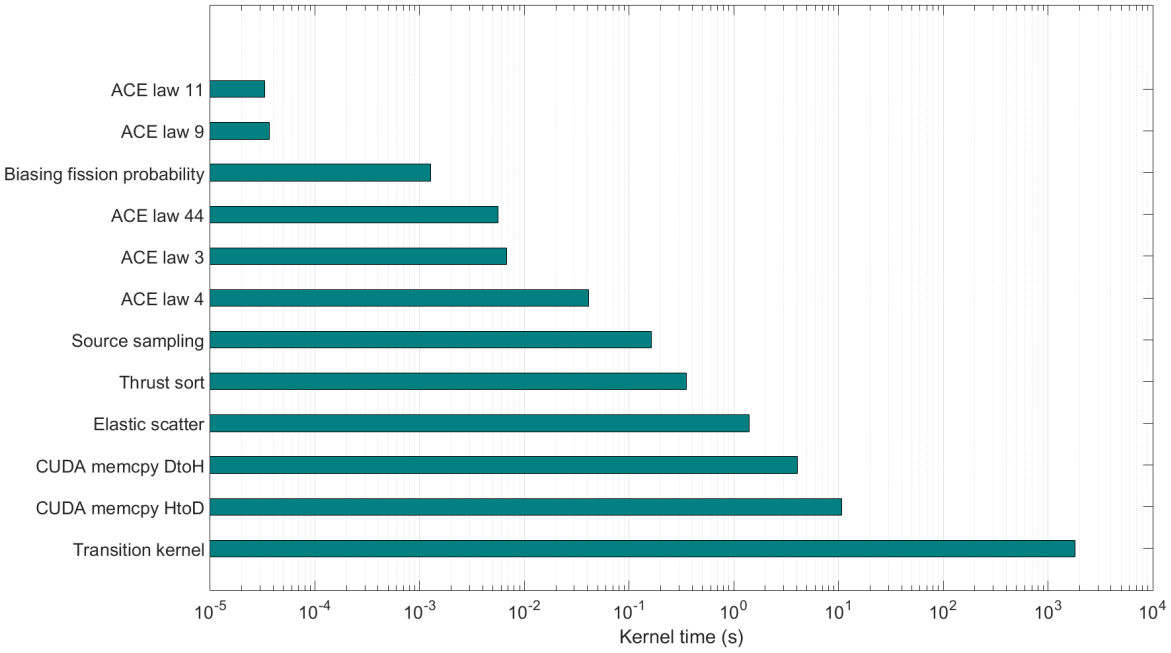

Figure 10: Profile of the application without acceleration methods [34]
Slowest GPU kernel was identified as the transition kernel = cell search + collision sampling

In this section we present two methods designed to accelerate the calculation of transporting particles between collisions. The cell search algorithm described in Sec. 2.2.2 focuses on finding the location of the current collision site more quickly, while the majorant mesh method (cf. Sec. 2.2.3) 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. 11). The majorant mesh was generated in line with the description of Sec. 2.2.3, using voxelized grid of size 0.3x0.3x0.6cm 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 5.

History-based		
	Majorant mesh OFF	Majorant mesh ON
Cell search OFF	101.98 s	2.89 s
Cell search ON	29.51 s	1.91 s

Event-based		
	Majorant mesh OFF	Majorant mesh ON
Cell search OFF	147.15 s	4.33 s
Cell search ON	44.85 s	2.82 s

Table 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.

Surprisingly, acceleration was found to be the same regardless of the execution structure. The history-based method seems to consistently outperform the event-based method by a factor of 1.5. This result is a bit counterintuitive, as we expected the event-based version to be closer to or even surpass the history-based version when acceleration is applied. Our understanding was that poor performance of the event-based method had been a consequence of the slow transition kernel as seen in Fig. 10. It may be that although serious improvements have been achieved by the majorant mesh and the acceleration of cell search routine, the majority of computational resources is still consumed by the transition kernel. As a consequence, further research should focus on developing a more efficient tree search method.

3.4. Whole-core transient analysis and testing against experimental data

Notwithstanding the comprehensive number of benchmark scenarios available for criticality calculations, many of them included in [36], it is hard to find an appropriate candidate for benchmarking the transient capabilities of GUARDYAN. Thus a transient experiment was set up in the Training Reactor at Budapest University of Technology and Economics, and GUARDYAN was challenged to reproduce the power evolution during this short transient.

3.4.1. 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 [40], [41] and [42], or assume very simplified systems such as [43] and [20]. Also, many transient benchmarks were prepared for deterministic codes with non-continuous energy cross sections, e.g. [44], or recently [45] [46] and [47]. A MC kinetic benchmark model, excluding thermal hydraulic feedback, for whole core transient analysis is very much needed in general. 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.. Due to this fact were previous studies on true dynamic Monte Carlo calculations compelled to resort to oversimplified, hypothetical models in order to verify the time-dependent neutronics module. For examples, see [16] for a simplified benchmark based on [44], or [13] for a benchmark based on [48]. 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 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.

3.4.2. Experimental setup

The experiment was carried out in the Training Reactor at the Institute of Nuclear Techniques, Budapest University of Technology and Economics. The facility operates a light water moderated and cooled reactor with several installations such as vertical irradiation channels, horizontal beam tubes, a large irradiation tunnel and pneumatic transfer systems serving research and educational purposes. An overview of the experimental assembly can be seen on Fig. 11. Transient behavior was induced by inserting a cadmium ring into the zone via the pneumatic transfer system indicated on this figure. The ring was 40mm high, with diameter of 9mm, and thickness of 0.5mm. The ring caused the local neutron absorption rate to rise especially in the thermal range, the overall reactivity insertion was around -20 cents.

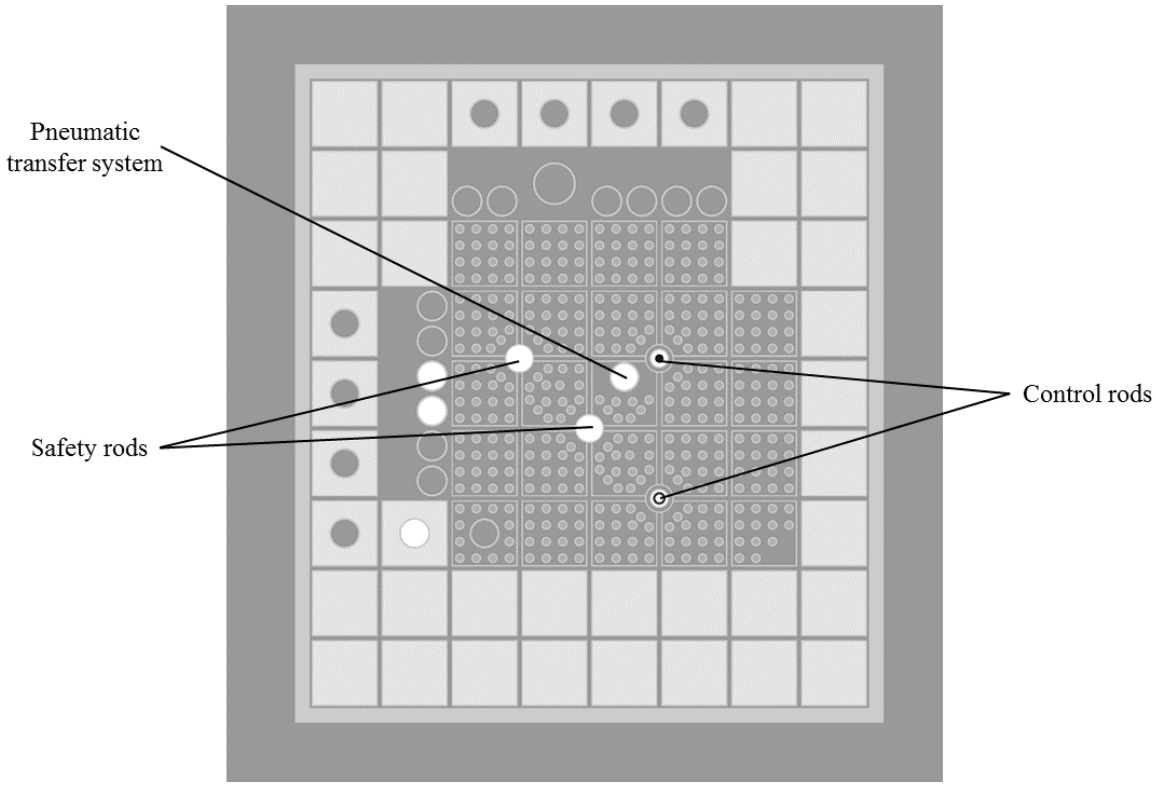

Figure 11: Geometry of the Training Reactor at Budapest University of Technology and Economics

Before the insertion the reactor was operated under critical conditions for minutes, the transient thus starting from an equilibrium state. Ignoring thermal hydraulic feedback effects was made possible by operating the reactor at low power, with initial power level set to 500W. 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. After the prompt jump, the transient was observed for a few seconds, then the cadmium ring was drawn out, causing the reactor to go back to critical state. The detector signal proportional to the total reactor power was registered by a fission chamber of type KNT–31 located in the water tank outside the zone. While this detector is part of the measurement chain I2 providing period and power level protection, the signal was directed through this channel, with TTL (transistor-transistor logic) signal accessible at the top of the reactor tank. The digital signal was then processed by an FPGA platform enabling high count rate data acquisition. The delay between rectangle signals was measured with a resolution of 25ns (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%. Measurement data after dead time correction can be seen in Fig. 12.

3.4.3. Monte Carlo model and transient analysis with GUARDYAN

Based on the MCNP benchmark model described in [49], an input of the Training Reactor was generated for GUARDYAN. The insertion of cadmium ring was modeled assuming constant velocity of $7.5 m/s$. The withdrawal was modeled as an accelerating movement starting from rest with acceleration of $30.6 m/s^2$, 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.

The simulation was run using 2^{22} ($\approx$ 4 million) samples, in agreement with the data structure of Fig. 2. The size of a time step was $10^{-5} s$ and importance weighted combing was used. Total runtime was around $150 h$ on a commercially available GPU card (Nvidia Geforce GTX 1080), with acceleration methods of Sec. 3.3.2 turned on for the computation. In a recent study [21], 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.

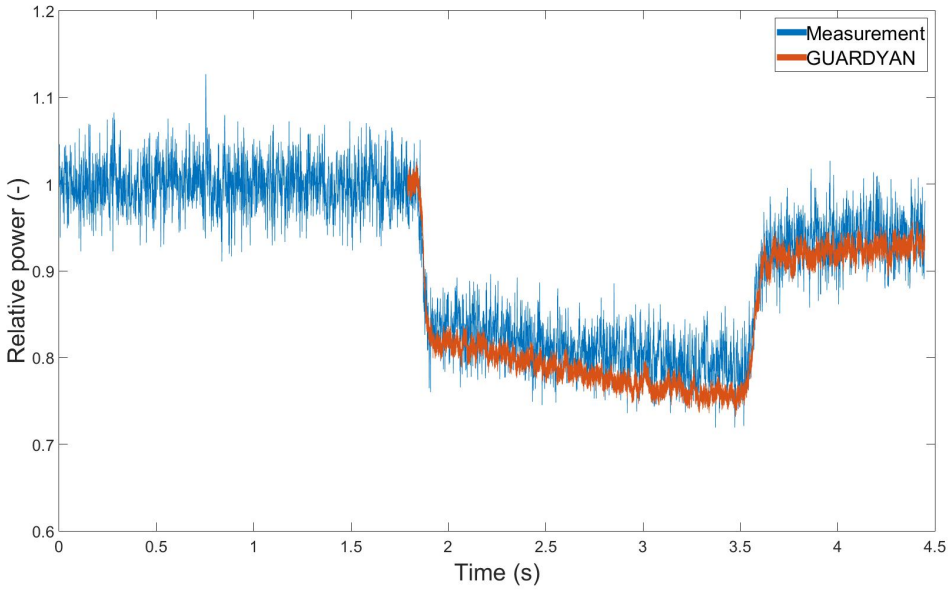

Figure 12: Testing GUARDYAN against experimental data. The two curves were normalized for matching power levels of the initial state

In Fig. 12 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 [50], with estimated dead time of $150ns$. 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. 13 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 $150ns$.

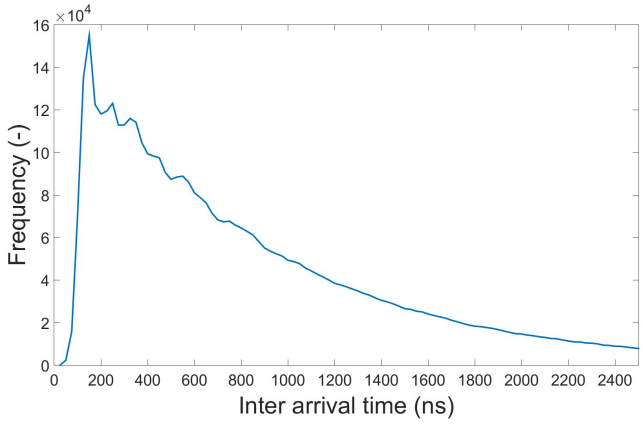

Figure 13: Time interval distribution of the measurement

Based on this observation, dead time was estimated as $150ns$. 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 subcritical period of the time series.

These promising results achieved with GUARDYAN imply that comparison to other kinetic codes like [21] 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. Future work will include the extension of the model description to allow for intercode comparison. An ongoing research has already targeted the simulation of the transient with a discrete ordinate neutron transport code PARTISN, currently being coupled with delayed neutron modeling.

4. Conclusions

An overview of GUARDYAN, a novel GPU accelerated time dependent Monte Carlo code was presented in this paper. A methodology was designed for MC simulations that treats time dependence directly and is

also efficient on state-of-the-art high performance computing platforms. The implementation of GUARDYAN on GPUs resulted in several techniques and approaches that are novel in the nuclear physics field. A robust treatment of prompt and delayed neutrons was developed addressing the timescale difference of typical lifetimes and considering GPU constraints. The combing method was successfully implemented on neutron and precursor populations, its effect on the uncertainty of time dependent power estimates was investigated. We found that by a simple definition of the importance of neutrons can improve the performance of combing method even if the importance is not time dependent. Concerns were raised about a correct variance analysis for kinetic MC tallies regarding possible correlation issues, the topic is suggested for further discussion. The event-based version of the code was also implemented targeting the thread divergence issue on the GPU. Comparison to the history-based method was accomplished, yielding a factor of 1.5 difference in favor of the history-based method.

The interaction physics of GUARDYAN was successfully verified by comparing 445000 data points to MCNP6 results. Moreover, the code was subjected to several criticality benchmarks from the International Handbook of Evaluated Criticality Safety Benchmark Experiments. Good agreement was found in both cases.

Whole core transient analysis was demonstrated with GUARDYAN and the code was tested against measurement data. An LWR transient experiment was designed and performed, setting grounds for a possible future kinetic benchmark. GUARDYAN was able to reproduce the power evolution of the transient within the uncertainties of the benchmark model, which is a novel and significant achievement among direct time dependent MC codes. Future work will include the extension of the benchmark study to allow for intercode comparison of both calculated values and runtimes.

5. Acknowledgment

This work has been carried out in the frame of VKSZ_14-1-2015-0021 Hungarian project supported by the National Research, Development and Innovation Fund.

References

[1] R. M. Bergmann, J. L. Vujić, Algorithmic choices in WARP–A framework for continuous energy Monte Carlo neutron transport in general 3D geometries on GPUs, Annals of Nuclear Energy 77 (2015) 176–193. doi:10.1016/j.anucene.2014.10.039.

[2] P. S. Brantley, R. C. Bleile, S. A. Dawson, N. A. Gentile, M. S. McKinley, M. J. OBrien, M. M. Pozulp, D. F. Richards, D. E. Stevens, J. A. Walsh, H. Childs, LLNL Monte Carlo transport research efforts for advanced computing architectures, in: Proceedings of International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering (M&C 2017), Jeju, Korea, 2017.

[3] Liu, Tianyu, Du, Xining, Ji, Wei, Xu, X. George, Brown, Forrest B., A comparative study of history-based versus vectorized Monte Carlo methods in the GPU/CUDA environment for a simple neutron

eigenvalue problem, in: Array (Ed.), SNA + MC 2013 - Joint International Conference on Supercomputing in Nuclear Applications + Monte Carlo, 2014, p. 04206. doi:10.1051/snamc/201404206.

[4] T. Okubo, T. Endo, A. Yamamoto, An efficient execution of Monte Carlo simulation based on delta-tracking method using GPUs, Journal of Nuclear Science and Technology 54 (1) (2016) 30–38. doi:10.1080/00223131.2016.1202793.

[5] F. B. Brown, Recent advances and future prospects for Monte Carlo, Progress in Nuclear Science and Technology 2 (0) (2011) 1–4. doi:10.15669/pnst.2.1.

[6] E. Brun, F. Damian, C. Diop, E. Dumonteil, F. Hugot, C. Jouanne, Y. Lee, F. Malvagi, A. Mazzolo, O. Petit, J. Trama, T. Visonneau, A. Zoia, TRIPOLI-4, CEA, EDF and AREVA reference Monte Carlo code, Annals of Nuclear Energy 82 (2015) 151 – 160, joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, SNA + MC 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms. doi:https://doi.org/10.1016/j.anucene.2014.07.053.

[7] P. K. Romano, N. E. Horelik, B. R. Herman, A. G. Nelson, B. Forget, K. Smith, OpenMC: A state-of-the-art Monte Carlo code for research and development, Annals of Nuclear Energy 82 (2015) 90 – 97, joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, SNA + MC 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms. doi:https://doi.org/10.1016/j.anucene.2014.07.048.

[8] J. Leppnen, M. Pusa, T. Viitanen, V. Valtavirta, T. Kaltiaisenaho, The Serpent Monte Carlo code: Status, development and applications in 2013, Annals of Nuclear Energy 82 (2015) 142 – 150, joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, SNA + MC 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms. doi:https://doi.org/10.1016/j.anucene.2014.08.024.

[9] A. G. Mylonakis, M. Varvayanni, D. Grigoriadis, N. Catsaros, Developing and investigating a pure Monte-Carlo module for transient neutron transport analysis, Annals of Nuclear Energy 104 (2017) 103–112. doi:10.1016/j.anucene.2016.12.039.

[10] J. Leppänen, Development of a dynamic simulation mode in Serpent 2 Monte Carlo code, Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013) (2013) 5–9.

[11] V. Valtavirta, M. Hessan, J. Leppänen, Delayed neutron emission model for time dependent simulations with the Serpent 2 Monte Carlo code-first results, in: Proceedings of the International Conference on the Physics of Reactors (PHYSOR2016), American Nuclear Society, Sun Valley, Idaho, USA, 2016.

[12] L. Russell, A. Buijs, G. Jonkmans, G4-STORK: A Monte Carlo reactor kinetics simulation code, Nuclear Science and Engineering 176 (3) (2014) 370–375. doi:10.13182/nse13-8.

[13] N. Shaukat, M. Ryu, H. J. Shim, Dynamic Monte Carlo transient analysis for the organization for economic co-operation and development nuclear energy agency (OECD/NEA) C5G7-TD benchmark, Nuclear Engineering and Technology 49 (5) (2017) 920–927. doi:10.1016/j.net.2017.04.008.

[14] A. Srivastava, K. Singh, S. Degweker, Monte Carlo methods for reactor kinetic simulations, Nuclear Science and Engineering (2017) 1–19doi:10.1080/00295639.2017.1388091.

[15] T. Yamamoto, H. Sakamoto, Dynamic Monte Carlo calculation method by solving frequency domain transport equation using the complex-valued weight Monte Carlo method, Annals of Nuclear Energy 85 (2015) 426–433. doi:10.1016/j.anucene.2015.04.041.

[16] B. L. Sjenitzer, J. E. Hoogenboom, Dynamic Monte Carlo method for nuclear reactor kinetics calculations, Nuclear Science and Engineering 175 (1) (2013) 94–107. doi:10.13182/nse12-44.

[17] D. Legrady, J. E. Hoogenboom, Scouting the feasibility of Monte Carlo reactor dynamics simulations, in: Proceedings of the International Conference on the Physics of Reactors 2008 (PHYSOR08), Paul Scherrer Institut, Interlaken, Switzerland, 2008.

[18] B. L. Sjenitzer, J. E. Hoogenboom, A Monte Carlo method for calculation of the dynamic behaviour of nuclear reactors, Progress in Nuclear Science and Technology 2 (2011) 716. doi:10.15669/pnst.2.716.

[19] B. L. Sjenitzer, J. E. Hoogenboom, Variance reduction for fixed-source Monte Carlo calculations in multiplying systems by improving chain-length statistics, Annals of Nuclear Energy 38 (10) (2011) 2195–2203. doi:10.1016/j.anucene.2011.06.013.

[20] B. L. Sjenitzer, J. E. Hoogenboom, J. J. Escalante, V. S. Espinoza, Coupling of dynamic Monte Carlo with thermal-hydraulic feedback, Annals of Nuclear Energy 76 (2015) 27–39. doi:10.1016/j.anucene.2014.09.018.

[21] M. Faucher, D. Mancusi, A. Zoia, New kinetic simulation capabilities for Tripoli-4®: Methods and applications, Annals of Nuclear Energy 120 (2018) 74–88. doi:10.1016/j.anucene.2018.05.030.

[22] P. Bialas, A. Strzelecki, Benchmarking the cost of thread divergence in CUDA, in: International Conference on Parallel Processing and Applied Mathematics, Springer, 2015, pp. 570–579. doi:10.1007/978-3-319-32149-3_53.

[23] T. Booth, A weight (charge) conserving importance-weighted comb for Monte Carlo, Tech. rep., Los Alamos National Lab., NM (United States) (1996).

[24] E. Woodcock, T. Murphy, P. Hemmings, S. Longworth, Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry, in: Proc. Conf. Applications of Computing Methods to Reactor Problems, Vol. 557, 1965, p. 2.

[25] L. L. Carter, E. D. Cashwell, W. M. Taylor, Monte Carlo sampling with continuously varying cross sections along flight paths, Nuclear Science and Engineering 48 (4) (1972) 403–411. doi:10.13182/nse72-1.

[26] S. Cramer, Application of the fictitious scattering radiation transport model for deep-penetration Monte Carlo calculations, Nuclear Science and Engineering 65 (2) (1978) 237–253. doi:10.2172/5257977.

[27] M. Galtier, S. Blanco, C. Caliot, C. Coustet, J. Dauchet, M. E. Hafi, V. Eymet, R. Fournier, J. Gautrais, A. Khuong, B. Piaud, G. Terrée, Integral formulation of null-collision Monte Carlo algorithms, Journal of Quantitative Spectroscopy and Radiative Transfer 125 (2013) 57–68. doi:10.1016/j.jqsrt.2013.04.001.

[28] L. Morgan, D. Kotlyar, Weighted-delta-tracking for Monte Carlo particle transport, Annals of Nuclear Energy 85 (2015) 1184–1188. doi:10.1016/j.anucene.2015.07.038.

[29] D. Legrady, B. Molnar, M. Klausz, T. Major, Woodcock tracking with arbitrary sampling cross section using negative weights, Annals of Nuclear Energy 102 (2017) 116–123. doi:10.1016/j.anucene.2016.12.003.

[30] B. Molnar, G. Tolnai, D. Legrady, Variance reduction and optimization strategies in a biased Woodcock particle tracking framework, Nuclear Science and Engineering 190 (1) (2018) 56–72. doi:10.1080/00295639.2017.1413876.

[31] J. F. Briesmeister, et al., MCNP-A general Monte Carlo N-particle transport code, Version 4C, LA-13709-M, Los Alamos National Laboratory (2000) 2.

[32] L. Szirmay-Kalos, B. Tóth, M. Magdics, Free path sampling in high resolution inhomogeneous participating media, in: Computer Graphics Forum, Vol. 30, Wiley-Blackwell, 2011, pp. 85–97. doi:10.1111/j.1467-8659.2010.01831.x.

[33] J. Amanatides, A. Woo, et al., A fast voxel traversal algorithm for ray tracing, in: Eurographics, Vol. 87, 1987, pp. 3–10.

[34] B. Molnar, G. Tolnai, D. Legrady, M. Szieberth, Vectorized Monte Carlo for GUARDYAN – a GPU accelerated reactor dynamics code, in: PHYSOR 2018: Reactor Physics paving the way towards more efficient systems, 2018.

[35] D. Legrady, A. Claret, B. Molnar, G. Tolnai, Validation of the interaction physics of GUARDYAN a novel GPU-based Monte Carlo code for short time scale reactor transients, in: PHYSOR 2018: Reactor Physics paving the way towards more efficient systems, 2018.

[36] J. D. Bess, T. Ivanova, International Handbook of Evaluated Criticality Safety Benchmark Experiments, NEA Nuclear Science Committee, 2016.

[37] D. E. Cullen, C. J. Clouse, R. Procassini, R. C. Little, Static and dynamic criticality: are they different?, Tech. rep., Lawrence Livermore National Lab., Livermore, CA (US) (2003). doi:10.2172/15009756.

[38] E. E. Gross, J. Marable, Static and dynamic multiplication factors and their relation to the inhour equation, Nuclear Science and Engineering 7 (4) (1960) 281–291. doi:10.13182/nse60-a25718.

[39] T. M. Sutton, A. Mittal, Neutron clustering in Monte Carlo iterated-source calculations, Nuclear Engineering and Technology 49 (6) (2017) 1211 – 1218, Special Issue on International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering 2017 (M&C 2017). doi:https://doi.org/10.1016/j.net.2017.07.008.

[40] M. Daeubler, A. Ivanov, B. L. Sjenitzer, V. Sanchez, R. Stieglitz, R. Macian-Juan, High-fidelity coupled Monte Carlo neutron transport and thermal-hydraulic simulations using Serpent 2/SUBCHANFLOW, Annals of Nuclear Energy 83 (2015) 352–375. doi:10.1016/j.anucene.2015.03.040.

[41] W. Syed, K. S. Chaudri, M. Sohail, Development of coupled neutron physics/thermal hydraulics system using OpenMC and RELAP5, in: International Conference on Emerging Technologies (ICET), IEEE, 2016, pp. 1–4. doi:10.1109/icet.2016.7813274.

[42] R. Henry, I. Tiselj, L. Snoj, CFD/Monte Carlo neutron transport coupling scheme, application to TRIGA reactor, Annals of Nuclear Energy 110 (2017) 36–47. doi:10.1016/j.anucene.2017.06.018.

[43] M. Aufiero, C. Fiorina, A. Laureau, P. Rubiolo, V. Valtavirta, Serpent–OpenFOAM coupling in transient mode: simulation of a Godiva prompt critical burst, Proceedings of ANS MC2015 - Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method (2015) 19–23.

[44] T. Kozlowski, T. J. Downar, PWR MOX/UO2 core transient benchmark final report, Nuclear Energy Agency, Organization for Economic Co-operation and Development, US Nuclear Regulatory Commission.

[45] U. Rohde, S. Kliem, U. Grundmann, S. Baier, Y. Bilodid, S. Duerigen, E. Fridman, A. Gommlich, A. Grahn, L. Holt, et al., The reactor dynamics code DYN3D – models, validation and applications, Progress in Nuclear Energy 89 (2016) 170–190. doi:10.1016/j.pnucene.2016.02.013.

[46] M. Knebel, L. Mercatali, V. Sanchez, R. Stieglitz, R. Macian-Juan, Validation of the Serpent 2-DYNSUB code sequence using the special power excursion reactor test III (SPERT III), Annals of Nuclear Energy 91 (2016) 79–91. doi:10.1016/j.anucene.2016.01.005.

[47] Q. Shen, Y. Wang, D. Jabaay, B. Kochunas, T. Downar, Transient analysis of C5G7-TD benchmark with MPACT, Annals of Nuclear Energy 125 (2019) 107 – 120. doi:https://doi.org/10.1016/j.anucene.2018.10.049.

[48] V. Boyarinov, P. Fomichenko, J. Hou, K. Ivanov, A. Aures, W. Zwermann, K. Velkov, Deterministic time-dependent neutron transport benchmark without spatial homogenization (C5G7-TD), Nuclear Energy Agency Organisation for Economic Co-operation and Development (NEA-OECD).

[49] G. Klujber, M. Szieberth, Benchmark description of the BME Training Reactor, Tech. rep., Budapest University of Technology and Economics, Institute of Nuclear Techniques, Budapest, Hungary (2018).

[50] G. F. Knoll, Radiation detection and measurement, 2nd Edition, John Wiley & Sons, 1989.