Budapest University of Technology and Economics

Institute of Nuclear Techniques

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

Ph.D. thesis booklet

Balázs Molnár

Supervisor: Dr. Dávid Légrády

Budapest

2020

1 Introduction

We consider time–dependent neutron transport in multiplying media. The neutron field in such systems can be described by the time–dependent Boltzmann equations:

$$\frac{1}{v}\frac{\partial \Phi(\vec{r},\vec{\Omega},E,t)}{\partial t} = -\vec{\Omega}\cdot\vec{\nabla}\Phi(\vec{r},\vec{\Omega},E,t) - \Sigma_t(\vec{r},E,t)\Phi(\vec{r},\vec{\Omega},E,t) + Q(\vec{r},\vec{\Omega},E,t) + \sum_{i=1}^{I}\chi_i(E)\lambda_i C_i(\vec{r},t) \quad (1.1)$$

$$\frac{\partial C_i(\vec{r},t)}{\partial t} = -\lambda_i C(\vec{r},t) + Q_i(\vec{r},\vec{\Omega},E,t) \qquad \text{for} \quad i=1,2,...,I \quad (1.2)$$

where Q and Q_i incorporates the source of neutrons and delayed neutron precursors, respectively:

$$Q(\vec{r},\vec{\Omega},E,t) = \int\limits_{4\pi}\int\limits_{0}^{\infty}\Sigma_s(\vec{r},\vec{\Omega}'\to\vec{\Omega},E'\to E,t)\Phi(\vec{r},\vec{\Omega}',E',t) + $$
$$+ \chi_p(\vec{r},E,t)(1-\beta(\vec{r},E',t))\nu(\vec{r},E',t)\Sigma_f(\vec{r},E',t)\Phi(\vec{r},\vec{\Omega}',E',t)dE'd\vec{\Omega}' + S(\vec{r},\vec{\Omega},E,t) \quad (1.3)$$

$$Q_i(\vec{r},\vec{\Omega},E,t) = \int\limits_{4\pi}\int\limits_{0}^{\infty}\beta_i(\vec{r},E',t)\nu(\vec{r},E',t)\Sigma_f(\vec{r},E',t)\Phi(\vec{r},\vec{\Omega}',E',t)dE'd\vec{\Omega}' \quad (1.4)$$

and we used notations of Table 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 $i = 1,2,...,I$.

In order to describe the dynamic behavior of nuclear reactors, we wish to solve these equations in the presence of feedbacks. For reactor transients in the order of seconds, the relevant feedbacks are due to temperature changes via neutron heating. Thus the complete mathematical framework also includes the modeling of thermal-hydraulic and thermal-mechanic phenomena, for such calculations, codes with various fidelity exist, and is not the scope of this thesis. State-of-the-art dynamic solutions also rely on coupling independent, already validated tools, and the subject of this work, dynamic MC, stands for a neutronic MC tool that solves for Eq. 1.1 and Eq. 1.2, while also prepared for such coupling and updating cross sections from time to time. To address the immense computational needs associated with the calculations, an implementation of dynamic MC was investigated, using Graphics Processing Units (GPUs). This thesis is a summary of our efforts regarding the development of GUARDYAN (GpU Assisted Reactor DYnamic ANalysis), a novel 3D continuous energy MC code specifically intended for the direct simulation of time dependence in nuclear reactors.

v	-	neutron speed
$\vec{r}$	-	position
$\vec{\Omega}$	-	direction of movement
E	-	energy
t	-	time
Φ	-	neutron flux
Σ_t	-	total macroscopic cross section
Σ_s	-	scattering cross section
Σ_f	-	fission cross section
β	-	delayed neutron yield
β_i	-	delayed neutron yield of family i
ν	-	fission neutron yield
χ_p	-	prompt fission neutron spectrum
χ_i	-	delayed fission neutron spectrum of family i
λ_i	-	decay constant of family i
C_i	-	precursor concentration of family i
I	-	number of precursor families
S	-	external source

Table 1: Notations used in this thesis

2 Methodology

We visioned GUARDYAN as a MC solver dedicated for reactor dynamics, rather than as a general purpose MC code. Our primary goal was to develop a methodology that heavily supports the simulation of time-dependent reactor behavior. Several issues that do not emerge in static k_{eff} or fixed source calculations had to be addressed merely due to the explicit time dependence of dynamic MC problems:

- To allow for dynamic changes in the system, e.g. thermal feedback or rod movements, classical processing of particle histories should be abandoned as histories should be regularly synchronized.

- Time dependent tallies must be considered. The simulation should ensure sufficient amount of samples to observe time evolution of tallies.

- Time shift of delayed neutrons should not be neglected. While typical lifetime of a delayed neutron is orders of magnitudes above that of prompt neutrons, this poses a serious challenge with respect to the distribution of MC samples in time.

- Population of MC samples should be controlled under transient conditions. If a reactor is in supercritical state, the number of samples could easily grow to be unmanageably high, while in a subcritical state the population may decline close to extinction yielding unreliable estimates.
- Initial conditions must be accounted for, from where the simulation starts.
- Variance reduction if applied should be flexible for dynamic changes

Many of the above points were already addressed in previous studies mostly dating from the early 2010's. Following the pioneering work of Sjenitzer and Hoogenboom [1], serious efforts are being made to develop dynamic modules to existing general purpose MC codes like Serpent [2] [3] [4] [5] OpenMC [6] or GEANT–4 [7] as well as to develop novel time–dependent MC codes [8] [9] [10]. A most significant step towards whole–core transient analyis was the integration of the Dynamic Monte Carlo method to TRIPOLI-4 [11] achieving decent stability even on large scale problems like a SPERT III E-core. Computing time is still a limiting factor for these applications however, reported runtimes include 300 CPU hours for one second of a transient. In the last decade of HPC in nuclear engineering, two distinct approaches emerged for parallelization: one is the application of CPU clusters, the other is general purpose GPU programming [12]. There is yet no decisive argument for the superiority of one over the other regarding MC modeling of reactor neutronics [13] [14] [15] [16] [17]. However, it is widely agreed, that GPU codes require different implementation of the same problem in order to compete with CPU codes. The ingenuity of the implementation is what determines the success of such codes.

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. 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. 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 [18]. 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. To achieve efficient parallelization, the methodology of GUARDYAN admits the following points:

- To overcome the thread divergence issue posed by the statistical nature of neutron chain lengths, GUARDYAN splits the simulation of neutron histories into time steps. Between time steps, the neutron population is kept constant. Population control is only considered at time boundaries, when neutron histories are synchronized.
- Tallying, point–in-cell search and distance–to–collision sampling methods are adjusted to the GPU in order to avoid significant performance loss.

A schematic view of the simulation flow of GUARDYAN is presented in Fig. 1. Simulation starts with the initial distribution of particles given by the user or, as an alternative, critical initial conditions can be constructed. Initial prompt neutron samples are obtained simply as surviving neutrons at the end of a time step while initial precursor samples must be generated from prompt neutrons at several time boundaries based on prompt–precursor balance at criticality. 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. 1. 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. Distance to collision is sampled by an improved Woodcock tracking routine with majorants calculated over a voxelized grid. 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.

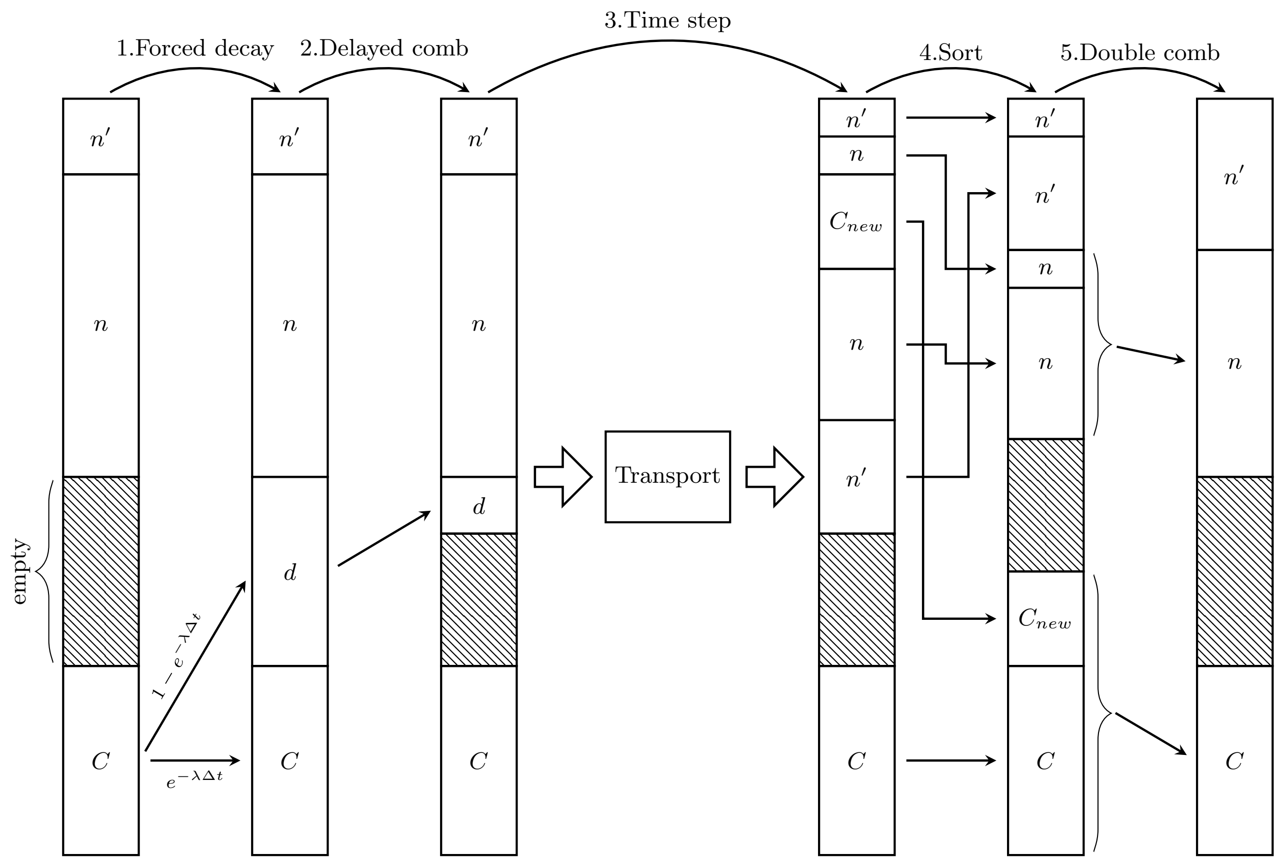

Figure 1: Operations on the GPU particle array

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 [19], 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 [20].

3. Simulation of a time step is performed on live neutrons (on stacks n and d in Fig. 1). During the simulation of neutron transport, history branching is prohibited and compensated by changing the statistical weight of the neutron, also, fission probability is biased for the better control of neutron weights. 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 [20] 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.

3 Thesis statements

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. [21], 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].

List of publications

[P1] Balazs Molnar, Gabor Tolnai, and David Legrady. A GPU-based direct Monte Carlo simulation of time dependence in nuclear reactors. *Annals of Nuclear Energy*, 132:46 – 63, 2019.

[P2] D. Legrady, A. Claret, B. Molnar, and 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.

[P3] B. Molnar, G. Tolnai, B. Toth, and D. Legrady. Guardyan – a novel GPU–based Monte Carlo code for simulating reactor transients (in Hungarian). *Nukleon*, 2019.

[P4] B. Molnar, G. Tolnai, D. Legrady, and 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.

[P5] B. Molnar, G. Tolnai, and D. Legrady. Variance reduction and optimization strategies in a biased Woodcock particle tracking framework. *Nuclear Science and Engineering*, 190(1):56–72, 2018.

[P6] B. Molnar and D. Legrady. Variance analysis of Woodcock type tracking. In *M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering*, 2017.

[P7] D. Legrady, B. Molnar, M. Klausz, and T. Major. Woodcock tracking with arbitrary sampling cross section using negative weights. *Annals of Nuclear Energy*, 102:116–123, 2017.

[P8] László Szirmay-Kalos, Iliyan Georgiev, Milán Magdics, Balázs Molnár, and Dávid Légrády. Unbiased light transport estimators for inhomogeneous participating media. In *Computer Graphics Forum*, volume 36, pages 9–19. Wiley Online Library, 2017.

References

[1] 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.

[2] 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.

[3] V. Valtavirta, T. Ikonen, T. Viitanen, J. Leppänen, Simulating fast transients with fuel behavior feedback using the Serpent 2 Monte Carlo, in: Proceedings of the International Conference on the Physics of Reactors (PHYSOR2014), Japan Atomic Energy Agency, Kyoto, Japan, 2015.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Q. Xu, G. Yu, K. Wang, Neutron generation based method for Monte Carlo three-dimensional reactor time-dependent simulation, in: 2012 20th International Conference on Nuclear Engineering and the

ASME 2012 Power Conference, American Society of Mechanical Engineers, 2012, pp. 481–487. `doi:10.1115/icone20-power2012-54929`.

[11] 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`.

[12] 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`.

[13] 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`.

[14] P. S. Brantley, R. C. Bleile, S. A. Dawson, N. A. Gentile, M. S. McKinley, M. J. O'Brien, 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.

[15] X. Du, T. Liu, W. Ji, X. G. Xu, F. B. Brown, Evaluation of vectorized monte carlo algorithms on gpus for a neutron eigenvalue problem, in: Proceedings of International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2013), Sun Valley, Idaho, USA, 2013, pp. 2513–2522.

[16] S. P. Hamilton, T. M. Evans, S. R. Slattery, Gpu acceleration of history-based multigroup monte carlo, Trans. Am. Nucl. Soc. 115 (2016) 527–530.

[17] A. Scuderio, Monte Carlo neutron transport: Simulating nuclear reactions one neutron at a time, in: GPU Technology Conference, San Jose, California, May, 2014, 2014.

[18] 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`.

[19] 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.

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

[21] 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.