RAD Conference Proceedings, vol. 1, pp. 21-25, 2016
www.rad-proceedings.org

PLATFORM DEPENDENT EFFICIENCY OF A MONTE CARLO CODE FOR TISSUE NEUTRON DOSE ASSESSMENT

Slobodan Milutinović, Filip Jeremić, Marko Mišić, Miloš Vujisić*, Predrag Marinković

School of Electrical Engineering, University of Belgrade, Belgrade, Serbia

Abstract. *Efficiency of a Monte Carlo algorithm for neutron dose calculation is compared in two implementations: a standard C++ code executed sequentially, and a CUDA C/C++ code which utilizes GPU resources for highly parallel processing. Both versions of the algorithm, developed specifically for this investigation, are based on the same physical model for the assessment of neutron dose in tissues, including lung, cortical bone and adipose tissue. The model treats emission and interaction of neutrons stochastically, utilizing cross sections for relevant interaction types. Several intentional simplifications have been introduced into the physical model used for simulations, which have allowed parts of the two codes to be related to one another in a straightforward way. A neutron's history is terminated when it leaves the outer ellipsoid (representing the human body), experiences any of the absorption interactions (inside one of the inner geometrical regions, representing tissues or organs), or if its energy falls below the cut-off limit set at 0.001 eV. The two approaches to algorithm implementation are compared according to execution speed, at various neutron source energies and for an increasing number of neutron histories. The fact that particle histories in a Monte Carlo simulation are independent from one another makes this kind of calculation suitable for implementation on parallel processing platforms. CUDA framework offers higher speeds of code execution, allowing more particle histories to be processed within a set time frame, and thus yields lower statistical uncertainty and higher reliability of the calculated neutron dose values. Appropriating standard C++ codes for CUDA is faced with specific challenges, which are described in the investigated case of neutron dose assessment. Despite the physical representation of neutron transport being somewhat simplified, comparison of both implementations to results obtained from MCNP shows good agreement in a wide range of neutron energies.*

Key words: *Neutron dose, Monte Carlo simulation, parallel processing, GPU, CUDA*

DOI: 10.21175/RadProc.2016.06

1. INTRODUCTION

Absorbed dose can be either measured or calculated. Since dose calculations offer many conveniences, especially in situations where measurements are cumbersome, unreliable, or impractical, many codes for Monte Carlo (MC) simulation of radiation transport have been developed [1]. MC simulation provides a detailed physical representation of radiation interactions, which makes these software tools superior to deterministic calculations in dealing with complex radiation fields and material structures, commonly encountered in practice. Extensive calculations required by any algorithm based on stochastic MC methods makes the advance of such codes closely related to the development of computer platforms. The goal of this paper is to investigate the possibility of using parallel processing capabilities of graphics processing units (GPUs) for speeding up calculations of absorbed dose in tissues arising from exposures to fast neutron sources.

Since they were first constructed, GPUs have long been used for specialized, intensive computations in the domain of computer graphics. Due to the high demand in high definition graphics market, they evolved into programmable, highly parallel multiprocessors with abundant raw computing power and high memory throughput. Both central processing units (CPUs) and GPUs are parallel computing systems, but their architectures differ considerably. Generally, GPUs are many-core processors with thousands of cores, while CPUs are multi-core processors with dozen of cores. On the other hand, GPU cores are simpler in structure and organized differently than CPU cores, thus making GPUs suitable for problems that exhibit high regularity and arithmetic intensity. CPU cores are task-parallel, latency-oriented processors. Caching, sophisticated flow control, and instruction level parallelism are used to hide memory latencies. On the contrary, GPUs are data-parallel and throughput-oriented processors. GPU cores are organized in several streaming multiprocessor units (SMs) of SIMD architecture. Control logic is simplified, as each core (scalar processor) in an SM executes the same instruction at the same time, but on different data. Expensive global memory accesses are hidden with extensive use of parallel, lightweight threads.

* vujsa@ikomline.net

Initially, GPUs have not been used for general-purpose computations, nor have their APIs supported such execution. Changes in architecture and programming model allowed GPUs to execute general-purpose computations written in standard, high-level programming languages, such as C, C++, and Fortran [2], especially with the development of NVIDIA CUDA (Compute Unified Device Architecture). Nowadays, GPUs are programmed through several extensions of C, C++, and Fortran programming languages, and numerous domain-specific libraries. In the past decade, GPUs have extensively used in both scientific and engineering applications, especially in computer simulations. The main goal is to produce high precision simulations with lower execution times [3].

What makes MC dose calculations suitable for being executed at highly parallel GPUs is the fact that particle histories are independent one from another. This allows a large number of threads to be run simultaneously during a simulation, each thread corresponding to a different history. Nevertheless, general complexity of an MC simulation makes the development of a GPU-based algorithm a challenging task. The example of tissue neutron dose assessment investigated herein serves to point out some of these challenges, as well as to demonstrate the platform dependent efficiency of an original MC algorithm developed for this purpose.

Simulations of neutron transport and tissue dose calculation are compared in two implementations: a standard C++ code executed sequentially, and a CUDA C/C++ code which utilizes GPU resources for highly parallel processing. The sequential version has been executed with an Intel Core i7-4770k CPU, and the parallel one with an NVIDIA GeForce GTX TITAN Black graphics processor with Compute Capability 3.5 [4],[5]. Physical model underlying the MC simulation has been validated by comparing calculated neutron dose values to corresponding results obtained from the MCNP software package [6].

2. Neutron Transport Algorithm

The most important characteristics of the algorithm for neutron transport simulation are presented in the following subsections.

2.1. Geometrical Representation of Sources and Material Regions

Our MC code uses quadratic surfaces for defining the source and the regions (i.e. tissues) exposed to the neutron field. Only spheres and ellipsoids have been used in neutron dose calculations performed for this study (see section 3), which means that in the general quadratic surface equation:

$$Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fzx + Gx + Hy + Jz + K = 0 \quad (1)$$

only constants A, B and C have positive non-zero values, K is negative, while constants in all other terms are zero.

The geometry and properties of the primary neutron source can, in principle, be arbitrary, while secondary neutron sources, i.e. nuclei on which inelastic scattering occurs, are assumed to be point isotropic sources.

2.2. Input Data

An MC simulation describes particle interactions as stochastic events. The probability for various neutron interactions and their outcomes depends on the energy of the incident neutron, as well as on the atomic number of the target nucleus. Both versions of our code expect these data to be prepared in advance as input files. Cross section data for neutron interactions were obtained from the ENDF/B-VII library [7], while data on material compositions and densities were taken from NIST reference libraries [8].

2.3. Description of Neutron Histories

Each neutron history starts at a randomly sampled point inside the source, after which it is transported along straight track segments between successive interaction points. Current neutron position (i.e. the point of emission in the source, or the position of the previous interaction) is determined by the position vector $\vec{r}_n$, its current direction by the unit vector $\vec{\Omega}_n$, while its current kinetic energy is denoted E_n. Values of $\vec{\Omega}_n$ and E_n are sampled from corresponding distributions, which depend either on the properties of the source or on the character of the interaction that a neutron has experienced. Position of the next interaction $\vec{r}_{n+1}$ is obtained as:

$$\vec{r}_{n+1} = \vec{r}_n + s\,\vec{\Omega}_n \quad (2)$$

where s is the track segment length, sampled as:

$$s = -\frac{1}{\Sigma_t}\ln\xi \quad (3)$$

where ξ is a random variable uniformly distributed over the (0, 1] interval, the values of which are obtained from a random number generator (RNG). Σ_t in equation (3) is the total macroscopic cross section for neutron interaction. For a compound material it is equal to:

$$\Sigma_t = \rho N_A \sum_{i=1}^{n} \frac{w_i}{M_i} \sum_{j=1}^{m} \sigma_{ij} \quad (4)$$

where ρ is material density, N_A is the Avogadro constant, M_i is the molar mass of the i-th element in the material, w_i is the mass fraction of the i-th element, while σ_{ij} is the partial microscopic cross section for the j-th type of interaction on the i-th element [9].

In a material composed of n different elements (i.e. nuclides), decision about the nuclide (with an index k) on which the interaction occurs is made by sampling the value of ξ (from the RNG) and finding the interval for which:

$$\sum_{i=1}^{k-1} N_i\sigma_{t,i} < \xi \sum_{i=1}^{n} N_i\sigma_{t,i} \le \sum_{i=1}^{k} N_i\sigma_{t,i} \quad (5)$$

where N_i is the concentration of nuclei of the i-th element, and $\sigma_{t,i}$ is the total microscopic cross section for the i-th nuclide.

Once the nuclide on which the interaction is to occur is chosen, the type of neutron interaction is decided, again by sampling the value of a corresponding discrete random variable:

$$\sum_{i=1}^{k-1}\sigma_i < \xi\sum_{i=1}^{n}\sigma_i \le \sum_{i=1}^{k}\sigma_i \qquad (6)$$

where n is the number of all possible neutron interactions, σ_i is the microscopic cross section for the i-th interaction type (on the nuclide chosen in the previous step), and k is the index denoting the type of interaction that takes place as a result of the sampling [10].

Several simplifications have been introduced into the physical model of neutron transport, which have allowed parts of the two codes to be related to one another in a straightforward way. The various neutron interactions have been classified into just three possible outcomes: elastic scattering, inelastic scattering (irrespective of the state to which the target nucleus is excited), or absorption. The two types of neutron scattering have been modeled by detailed equations, which allow energy loss and change of direction to be sampled [9]-[11]. For all the absorption interactions, on the other hand, local deposition of all of neutron's kinetic energy at the site of interaction has been assumed, disregarding the exact type of interaction, as well as any secondary particles that may come out of it.

A neutron's history is terminated when it leaves the outer ellipsoid (representing the human body), experiences an absorption interaction (either in the ellipsoid or inside one of the inner geometrical regions), or if its kinetic energy falls below the cut-off limit set at 0.001 eV.

2.4. Output of the Simulation Algorithm

The results of a simulation run are the values of absorbed dose in geometrical regions positioned inside the water-filled ellipsoid representing tissues or organs. Repeated simulation runs differed in the number of neutron histories followed, the energy of the neutron source, and the geometrical arrangement of the source and the tissues. For the purpose of analyzing the efficiency of two algorithm implementations, durations of simulation runs have been recorded by the codes themselves and provided as part of the output data.

3. Results and Discussion

3.1. Comparison of C++ and CUDA C/C++ Codes

Sequential (C++) and parallel (CUDA C/C++) implementation of the simulation share some similarities, but their executions still differ considerably. The differences are mostly due to the programming model and the hardware used for execution. Sequential implementation executes on a CPU by iterating a given number of neutron histories. Each history completes and produces a result before the next one begins. Parallel implementation executes groups of threads simultaneously, while each thread simulates a complete history of an incident neutron. Neutron transport simulation can be organized in this way because the order of simulated histories is not relevant, since neutrons don't interact among themselves. The only cooperation between the threads is the one needed to produce simulation results. Threads in a single group produce their respective results simultaneously, and then a reduction operation is used to accumulate results for each simulated track segment.

The structure exposed to neutrons consisted of one large ellipsoid with two smaller spheres within (Fig. 1). The ellipsoid, representing the human body, was filled with water. One sphere contained lung tissue and the other bone tissue. A monoenergetic point isotropic neutron source was first placed inside one of the spheres (at (3,2,1) cm), and then outside the ellipsoid (at (−33,0,0) cm). Obtained speedups of the parallel over the sequential implementation for the two positions of the source are presented in Figs. 2 and 3. Each of the bars in these graphs corresponds to a different number of neutron histories followed in the simulation.

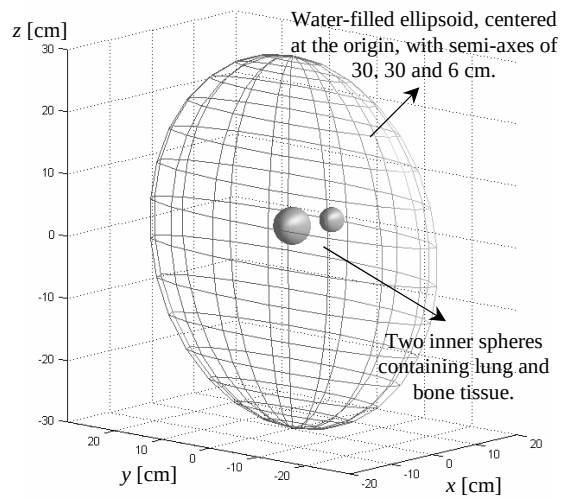

Figure 1. Structure exposed to neutrons in simulations used for comparing the two programming platforms. A point neutron source was first placed at (3,2,1) cm, and then at (−33,0,0) cm

We believe that the observed speedups of the parallel over the sequential implementation are smaller than what could be achieved with a CUDA platform, for several reasons. Most of the data is stored in the slow global memory: coordinates of simulated track segments, coordinates of intersection points of tracks with surfaces, various constants, etc. Since sequential search is a part of the algorithm, it can lead to memory bank conflicts. Also, due to the nature of the simulation, neutrons simulated by the threads don't usually take the same execution paths. This leads to branch divergence related performance slowdowns.

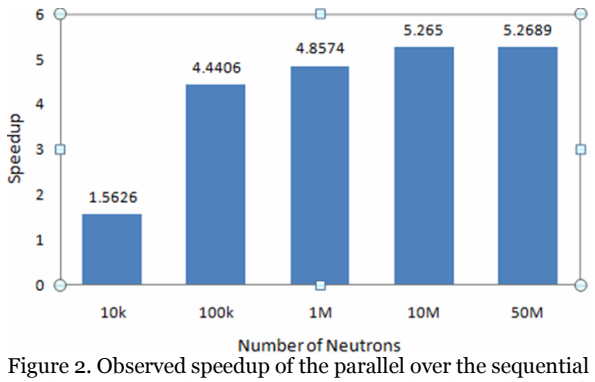

Figure 2. Observed speedup of the parallel over the sequential implementation when neutron source is placed inside one of the inner small ellipsoids

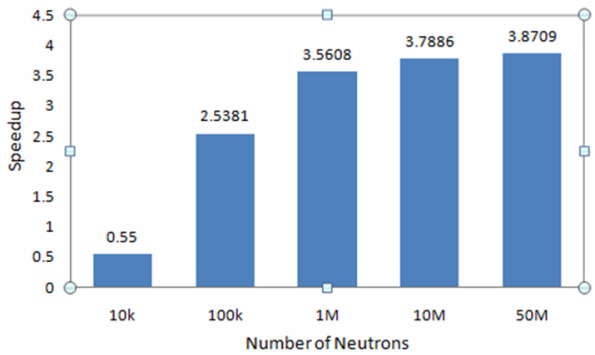

Figure 3. Observed speedup of the parallel over the sequential implementation when neutron source is placed outside the large ellipsoid

Comparison of Figures 2 and 3 shows that the performance of the parallel implementation is slightly better if the neutrons source is placed inside one of the ellipsoids. This is mostly due to reduced branch divergence, since in that case neutrons must interact with at least one segment. Also, the results show that observed speedups remain fairly constant with increasing number of neutron histories (over one million), as parallel overheads become negligible compared to overall computation time. We plan to perform further investigation and optimization of the code in our future work.

3.2. Verification of the Codes

The simplified physical model that underlies neutron transport simulations has been validated by comparing results obtained from the CUDA C/C++ code to corresponding results obtained from MCNP. Ratio of neutron absorbed dose values obtained by the CUDA C/C++ code and MCNP are presented in Tables 1 and 2. The doses have been calculated in a sphere of variable radius composed of a tissue material, with a point isotropic neutron source positioned either inside or outside the sphere. Incident neutron energy has also been varied from 0.5 MeV to 19 MeV. Table 1 gives the said ratio for various geometrical arrangements of source and sphere, and for a range of neutron source energies, when the sphere is composed of lung tissue, while Table 2 refers to the case when it is composed of cortical bone.

For neutron energies in the range 0.5–4 MeV absorbed dose ratio values in Tables 1 and 2 testify to a good agreement between results obtained from the algorithm used in our study and those from MCNP, with discrepancies of less than 3% in case of lung tissue, and below 10% for bone tissue. One reason for the observed deviations of absorbed dose values is that all neutron scattering interactions in the CUDA C/C++ code are assumed to be isotropic in the center of mass system, which is only approximately true for most elements that constitute the tissues [11].

Lower discrepancy for lung tissue, which is evident when Tables 1 and 2 are compared, is attributable to the absence of high-Z elements in soft tissue. The assumption that the energy lost by neutrons in absorption interactions is deposited locally, i.e. deposited completely within the sphere, is more of an issue for elements with higher atomic numbers, because their absorption cross section is higher. This means that in bone tissue, which contains a lot of calcium ($Z = 20$), absorption reactions occur more frequently, and the assumption of local energy deposition affects the results more. The same assumption is responsible for the observed increase of absorbed dose discrepancy as neutron energy rises (scattering cross section becomes much lower than the absorption cross section), or as sphere volume is decreased (making the sphere less likely to actually hold all of the absorbed neutrons's energy).

Table 1. Ratio of neutron absorbed dose values obtained by the CUDA C/C++ code and MCNP for a sphere composed of lung tissue

Neutron energy [MeV]	Distance from the source to the center of the sphere / sphere radius			
	1.1 cm / 1 cm	3.1 cm / 3 cm	6.1 cm / 6 cm	15.1 cm / 15 cm
0.5	1.02	1.00	0.99	0.97
1	0.98	0.99	1.00	1.00
2	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00
4	1.03	1.02	1.02	1.00
6	1.02	1.02	1.02	1.02
8	1.17	1.16	1.14	1.13
10	1.27	1.26	1.25	1.22
12	1.35	1.33	1.30	1.27
15	1.48	1.48	1.43	1.36
19	1.44	1.44	1.41	1.36

Table 2. Ratio of neutron absorbed dose values obtained by the CUDA C/C++ code and MCNP for a sphere composed of cortical bone.

Neutron energy [MeV]	Distance from the source to the center of the sphere / sphere radius			
	1.1 cm / 1 cm	3.1 cm / 3 cm	6.1 cm / 6 cm	15.1 cm / 15 cm
0.5	1.03	1.00	0.99	0.93
1	0.99	0.99	1.00	0.98
2	1.02	1.02	1.02	1.00
3	1.03	1.03	1.01	1.00
4	1.10	1.08	1.06	1.02
6	1.15	1.14	1.14	1.11
8	1.44	1.40	1.37	1.32
10	1.56	1.54	1.51	1.47
12	1.64	1.61	1.60	1.52
15	1.84	1.78	1.73	1.62
19	1.74	1.72	1.69	1.61

Figures 4 and 5 show how the absorbed dose in a sphere containing lung and bone tissue, respectively, depended on the energy of incident neutrons. The sphere had a 1 cm radius, and the point neutron source was placed 1.1 cm away from its center. Results for neutron tissue dose obtained both from the CUDA C/C++ code and from MCNP are plotted in these graphs. Each simulation run followed 10^6 neutron histories. It can be noticed that there is good agreement between CUDA and MCNP curves up to the neutron energy of about 6 MeV, while at higher energies the curves diverge.

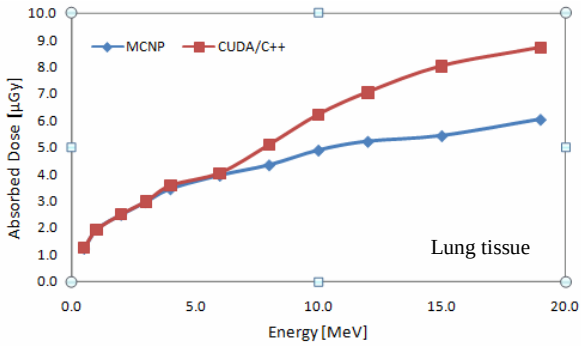

Figure 4. Dependence of the calculated absorbed dose in a sphere containing lung tissue on the energy of incident neutrons. Results obtained from both the CUDA C/C++ code and MCNP are plotted

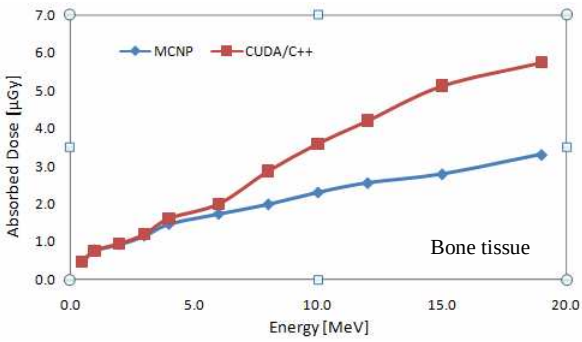

Figure 5. Dependence of the calculated absorbed dose in a sphere containing bone tissue on the energy of incident neutrons. Results obtained from both the CUDA C/C++ code and MCNP are plotted

4. Conclusion

Two implementations of a Monte Carlo algorithm for neutron tissue dose calculation have been compared: a standard C++ code executed sequentially, and a CUDA C/C++ code which utilizes GPU resources for highly parallel processing. Speedups achieved with the CUDA code, even if lower than theoretically achievable, present a significant contribution to execution efficiency of the simulations. Faster execution of the code means that more particle histories can be followed in the same amount of time, thus yielding lower statistical uncertainty and higher reliability of the calculated neutron dose values. Even though the physical model used for simulating neutron interactions was somewhat simplified, results obtained with the CUDA C/C++ code showed good agreement with values produced by MCNP, especially for neutron energies below 6 MeV. Further investigation will be directed towards optimization of the MC simulation algorithm, so that the advantages offered by the CUDA platform could be harvested to their fullest.

Acknowledgement: *We would like to thank our colleagues from the Radiation and Environmental Protection Department of the Vinca Institute of Nuclear Sciences for their help with simulations performed in MCNP.*

References

1. H. Nikjoo, S. Uehara, D. Emfietzoglou and F.A. Cucinotta, "Track-Structure Codes in Radiation Research", *Radiat. Meas.*, vol. 41, no. 9-10, pp. 1052-1074, Oct.-Nov. 2006
2. M. Mišić, Đ. Đurđević and M. Tomašević, "Evolution and Trends in GPU Computing", in *Proceed MIPRO*, Opatija, Croatia, May 2012 pp. 289-294
3. D.B. Kirk and W.W. Hwu, *Programming Massively Parallel Processors*, 2nd ed., Waltham (MA), USA: Morgan Kaufmann, 2013
4. Intel Core i7-4770K Processor (8M Cache, up to 3.90 GHz), Intel Corp., Santa Clara (CA), USA Retrieved from: http://www.intel.com/content/www/us/en/company-overview/contact-us.html
5. *GeForce GTX TITAN Black: Specifications*, Nvidia Corp., Santa clara (CA), USA Retrieved from: http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan-black/specifications
6. T.E. Booth et al. (X-5 Monte Carlo Team), "MCNP — A General Monte Carlo N-Particle Transport Code", ver. 5, vol. I: Overview and Theory, Los Alamos National Laboratory, DOE, Univ. of California, (NM) USA, 2003
7. *T-2 Nuclear Information Service: ENDF/B-VII.1 Incident-Neutron Data*, Los Alamos National Laboratory, Univ. of California, (NM) USA Retrieved from: http://t2.lanl.gov/nis/data/endf/endfvii.1-n.html
8. *NIST PML Physical Reference Data: X-Ray Mass Attenuation Coefficients*, NIST, Gaithersburg (MD), USA Retrieved from: http://physics.nist.gov/PhysRefData/XrayMassCoef/tab2.html
9. J. Spanier & E. M. Gelbard, *Monte Carlo Principles and Neutron Transport Problems*, Mineola (NY), USA: Dover Publ., 2008
10. I. Lux and L. Koblinger, *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*, Bosa Roca (FL), USA: CRC Press, 1991
11. Paul Reuss, *Neutron Physics*, Paris, France: EDP Sciences, France, 2008