A GPU-BASED MONTE CARLO DOSE CALCULATION CODE FOR PHOTON TRANSPORT IN A VOXEL PHANTOM

Murillo Bellezzo, Eduardo do Nascimento, Helio Yoriyaz

Instituto de Pesquisas Energéticas e Nucleares - Ipen-Cnen/SP, Av. Lineu Prestes, 2242, Cidade Universitária, 05508-000, São Paulo, Brazil

mbellezzo@gmail.br, eduardon@usp.br, hyoriyaz@ipen.br

Abstract

As the most accurate method to estimate absorbed dose in radiotherapy, Monte Carlo method (MCM) has been widely used in radiotherapy treatment planning. Nevertheless, its efficiency can be improved for clinical routine applications. In this paper, we present the CUBMC code, a GPU-based MC photon transport algorithm for dose calculation under the Compute Unified Device Architecture (CUDA) platform. The simulation of physical events is based on the algorithm used in PENELOPE, and the cross section table used is the one generated by the MATERIAL routine, also present in PENELOPE code. Photons are transported in voxel-based geometries with different compositions. To demonstrate the capabilities of the algorithm developed in the present work four 128x128x128 voxel phantoms have been considered. One of them is composed by a homogeneous water-based media, the second is composed by bone, the third is composed by lung and the fourth is composed by a heterogeneous bone and vacuum geometry. Simulations were done considering a 6 MeV monoenergetic photon point source. There are two distinct approaches that were used for transport simulation. The first of them forces the photon to stop at every voxel frontier, the second one is the Woodcock method, where the photon stop in the frontier will be considered depending on the material changing across the photon travel line. Dose calculations using these methods are compared for validation with PENELOPE and MCNP5 codes. Speed-up factors are compared using a NVidia GTX 560-Ti GPU card against a 2.27 GHz Intel Xeon CPU processor.

Keywords: GPU; dosimetry; Monte Carlo simulation; radiotherapy; photon transport; PENELOPE; MCNP5

1.- INTRODUCTION

Technological evolution has doubled the number of transistors present in processors every two years for 30 years following the pattern predicted by Moore's Law (Schaller, 1997). However, the evolution of a single processing core is decreasing and the tendency is that the parallel processing is used to maintain the existing curve progression.

Clusters are supercomputers composed by hundreds or thousands processors communicating to each other, performing parallel processing. They have a high processing power, however, are not easily accessible due to its high cost. An alternative to clustering is the graphical processing units (GPUs) which are designed for graphics processing and already having an architecture designed for parallel processing. Widely used in games, the processing power of GPU has surpassed that of central processing units (CPUs), so that, new softwares are now making use of the GPU to increase performance.

There are programming languages that allow portions of the code to be executed on GPU achieving performance gain compared to the code execution using only the CPU. This gain is enhanced when the task performed by the program is highly parallelizable, which is the case of computer simulations using the Monte Carlo method.

The Monte Carlo method is a statistical method widely used in simulation of radiation transport in matter to obtain physical quantities of interest, such as, particle flux and deposited energy. Among the existing computational programs using the Monte Carlo method, stand out the MCNP (Goorley JT, 2013), PENELOPE (Salvat F., 2006) and GEANT4 (Geant4 Collaboration, 2010) codes. These programs have been developed and refined over decades of research and are now widely used in the area of medical physics research. However, they are too complex for general use demanding specific computational knowledge and experience from the users and demanding considerable processing time for the utilization in therapy planning systems in radiotherapy.

Aiming for a high performance algorithm for dose calculation studies in radiation therapy this work presents the development of the CUBMC (CUDA-Based Monte Carlo) code - an optimized Monte Carlo code to run on GPU that simulates the transport of photons in voxel-based phantoms. Physical models of photon interaction with matter used in the algorithm were extracted with appropriate simplifications from the PENELOPE code. Cross-section tables were also extracted from the MATERIAL routine of the PENELOPE code.

2.- MATERIALS AND METHODS

2.1.-Monte Carlo method

The Monte Carlo method is a statistical method usually utilized in problems where the analytical solution is too complex. The specific steps of the method vary according to the problem addressed, but basically, a problem simulated by the Monte Carlo method goes through the following steps (Brown FB, 1996):

1 – Initially, a complex problem is reduced into a sequence of simple events where the set of all possible events represent its domain.

2 - Then, an event (or sequence of events) is sampled with the aid of pseudo-random numbers (PRN) and probability density functions (PDF).

3 - The sampled event becomes part of a statistical group and the process repeats until obtaining the desired precision related to average value of sampled event group. For sufficiently large statistical samples, the result converges towards the analytical result of the problem.

2.2.-Pseudo-random number generator

Pseudo-random number generator (PRNG) is one of the main keys in a Monte Carlo simulation. A pseudo-random number sequence is composed by pseudo-random numbers that simulate the randomness sequence of numbers, but, they are generated by an algorithm having a well-defined sequence.

There is the possibility for the utilization of two or more seeds at a time allowing generators with higher sequence numbers. When parallelizing a PRNG, one should ensure that the seed used by a generator does not appear in other generator; otherwise the two generators would then return the same number. This is avoided dividing the original sequence into several smaller sequences where each generator works. In practice, a PRNG which has a sequence of "N * M" numbers can be separated into M PRNG each one generating N pseudo-random numbers. The caution that must be taken when sizing the parallelization of a PRNG is to assure that the amount of numbers available for each generator is enough to perform the desired simulation.

The generator used in the present work is a parallelized version of RANECU (L'Ecuyer, 1988), which presents a sequence of approximately 10^{18} numbers. The method of parallelization was the same utilized in the work by Badal on the parallelization of the code PENELOPE (Badal, 2006).

The degree of parallelization is done in order to have one generator per thread. Even in the condition of largest occupation of GPU (12280 threads) we still get a sequence of about 10^{14} numbers by thread which is sufficient to simulate up to 10^{12} photons per thread.
The implemented parallel version of the generator on the GPU demonstrated to be 100 times faster than the version running on the CPU.

2.3.-Photon transport

The simulation model developed in this study does not cover other particles other than photons (KERMA approximation) which is adequate for the majority of applications in radiotherapy. Thus, secondary particles generated during the simulation of the history of a

photon will have their energy deposited locally. This approximation is valid due to the fact that the mean free path and penetration of a photon are orders of magnitude greater than that of an electron or positron of same energy. To simulate the transport of photons there are four main phenomena that are most important to the energy deposition for medical physics applications: photoelectric effect, Compton Effect, Coherent scattering and pair production. The physical models for the photon interaction were based on the models taken from PENELOPE code. The simulation of the history of a photon can be divided into the following steps (Cashwell ED, 1959):

Distance to next collision: In this step, the propagation direction and the step length carried by the photon are determined by consulting the table of total cross section of the material in which the photon propagates. If the code is using the Woodcock method, it is assumed that the particle is shifted into the largest total cross-section material, σ_t.

Boundary check: If the photon reaches any boundary in the geometry, it is stopped and a new collision distance is sampled based on the material present in the other side of the boundary where the photon is travelling. The border checking does not occur in the Woodcock method.

Interaction check: The type of interaction is sampled considering the cross-section of four types of photon interactions with matter: photoelectric, Compton scattering, pair production and Rayleigh scattering. Since only photon is considered two variables is necessary for this subroutine: the incident photon energy and flying direction. As an outcome, the emergent photon energy and new angle are sampled.
Tallies: The tallies are functions that collect data relevant to the simulation. In the current model, it returns the energy deposited in the voxels.

After performing these steps, if the photon has not reached the minimum energy or abandon the geometry a new cycle starts.

2.4.-Woodcock method

During the simulation of the transport of a particle, every time it crosses a boundary, it is forced to stop to check the next material in which it will propagate. This process can be computationally very expensive particularly in geometries comprising several interfaces, as is the case of voxel-based phantoms. Woodcock method (Woodcock, 1965) is an alternative method that performs the transport simulation ignoring the existing boundaries in the geometry.

The mean free path (λ) of a particle depends on the material in which it travels. This characteristic is related to the total cross section of the material (σ_t) so that the greater the value of σ_t smaller the value of λ. The Woodcock method consists in considering that the particle is propagated in a homogeneous medium without borders composed by phantom material with largest cross-section, σ_{max}, i.e, the material in which the particle has the smallest mean free path. In case this mean free path is greater than the average distance between the boundaries there will be a performance gain. To compensate the approximation of homogeneous medium, a probability of particle interaction with matter is introduced (without the Woodcock method it is considered a 100% chance that there is an interaction). The probability that an interaction occurs is given by the ratio between σ_t and σ_{max}.

2.5.-Graphical processing Unity

The GPU has emerged as an auxiliary processor intended for graphics processing in order to reduce the amount of arithmetic processing managed by the central processing unit (CPU). Analyzing the architecture of a CPU and a GPU, we can observe that a CPU has a developed control area while the logic arithmetic units (ALU) are scarce. On the other hand, the GPU has only few control cores and several ALU units. This difference in architecture shows that CPUs are optimized for managing tasks in parallel but inefficient to

perform arithmetic tasks. On the contrary, GPUs have little ability with respect to the processing of multiple tasks but are highly efficient in performing arithmetic operations.
Aiming to take advantage of GPU's high capacity for parallel arithmetic processing, new programming codes capable of directing the flow of their operations to GPU's have been emerged. Among them, stands out the CUDA (Compute Unified Device Architecture) which is the programming language used in this work.

CUDA is a programming language developed by NVIDIA and includes commands and libraries in C, C++, C# and FORTRAN (NVIDIA, 2010). CUDA C is similar to C/C++ language, with the difference of having tools to establish a communication between the GPU and CPU, so that, parts of the code that are responsible for arithmetic operations can be performed on the GPU. However, for optimum performance, the code must have their parameters adjusted according to the capacity of each unity.

3.- RESULTS

Simulations were performed in four phantom types, respectively, composed by water, bone, lung and bone-vacuum media. In each case, the geometry has been described by a 64 x 64 x 64 cm^3 cube phantom composed by 128 x 128 x 128 voxels with 0.5 x 0.5 x 0.5 cm^3 in volume.

An isotropic 6 MeV point photon source was simulated at the position X = Y = Z = 32.1 cm contained in the voxel of index (64, 64, 64) (placed slightly off from the geometry center to not be defined on the boundary between voxels).

The simulations were performed by the Woodcock and stopping methods. Dose values were compared with those obtained by simulations with MCNP and PENELOPE codes. The total execution time from CUBMC was compared with the time elapsed on PENELOPE simulations for efficiency gain estimation. For dose accuracy estimation, simulations in MCNP were used as reference.

Although there is no physical boundary in a homogeneous material, its geometry is still bounded by voxels, and these boundaries force the particles to stop. This generates a significant delay in runtime, because the particle can trap hundreds of borders during its history.

The simulations using the PENELOPE code were ran on a 2.27 GHz i7 Intel Xeon processor, The MCNP simulations were performed in SGI Altix XE 340 (composed by 96 cores Intel Xeon six-core 5650 2.66 GHz), and the CUBMC simulations were ran on a GTX560Ti NVidia GPU. For dose comparison, it was sampled a row of voxels passing right above the source. The row was defined by the voxels of index Vx=[0:127], Vy=64 and Vz=96, within the middle plane between the source and the geometry edge.

Figure 1 shows the results from CUBMC simulations using Woodcock and Stopping methods in comparison with those obtained by the MCNP code. The number of particles simulated in CUBMC and MCNP are 10^{10} and 4 x 10^{8}, respectively. The MCNP statistical uncertainties are less than 0.1 %. Unfortunately, in this version, the CUBMC statistical uncertainties are not implemented yet, but from this figure one can observe a very good agreement between these two codes with differences less than 3.5%. PENELOPE results (not shown) are also in good agreement with other both codes results.

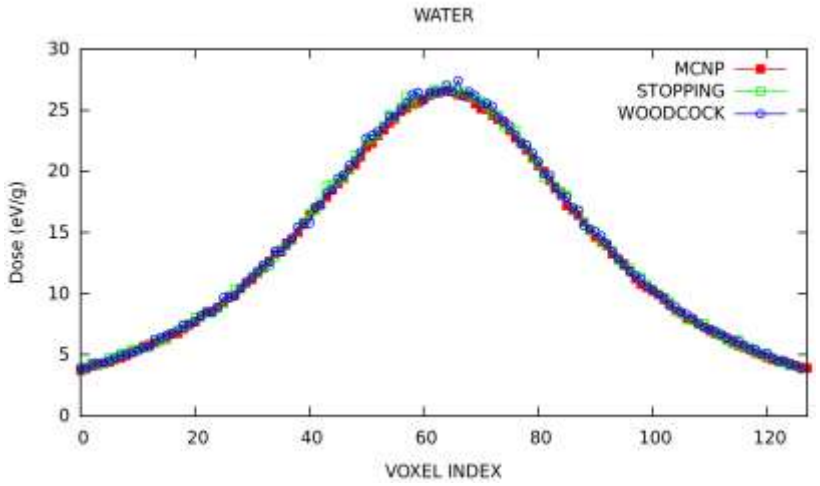

Figure 1.- Simulation for photons traveling on a water based media. There were used 10^{10} particles on CUBMC simulations, and 4x10^{8} particles on MCNP simulations.

The same behavior was observed in the simulations for homogeneous media of lung and bone, as we can see on Figures 2 and 3.

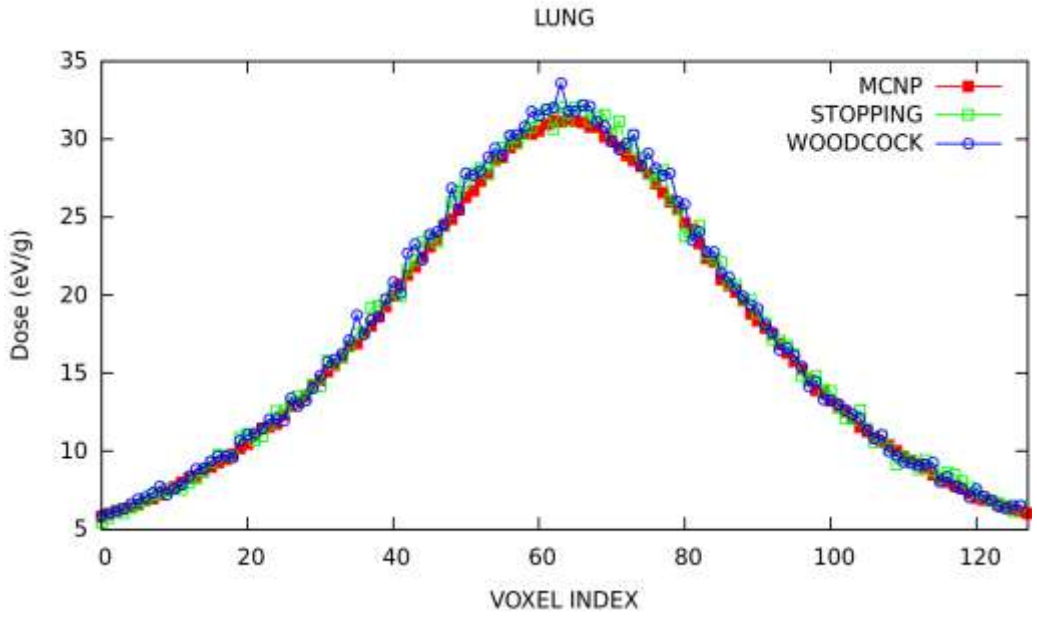

Figure 2.- Simulation for photons traveling on a lung based media. There were used 10^{10} particles on CUBMC simulations and $4x10^{8}$ particles on MCNP simulations

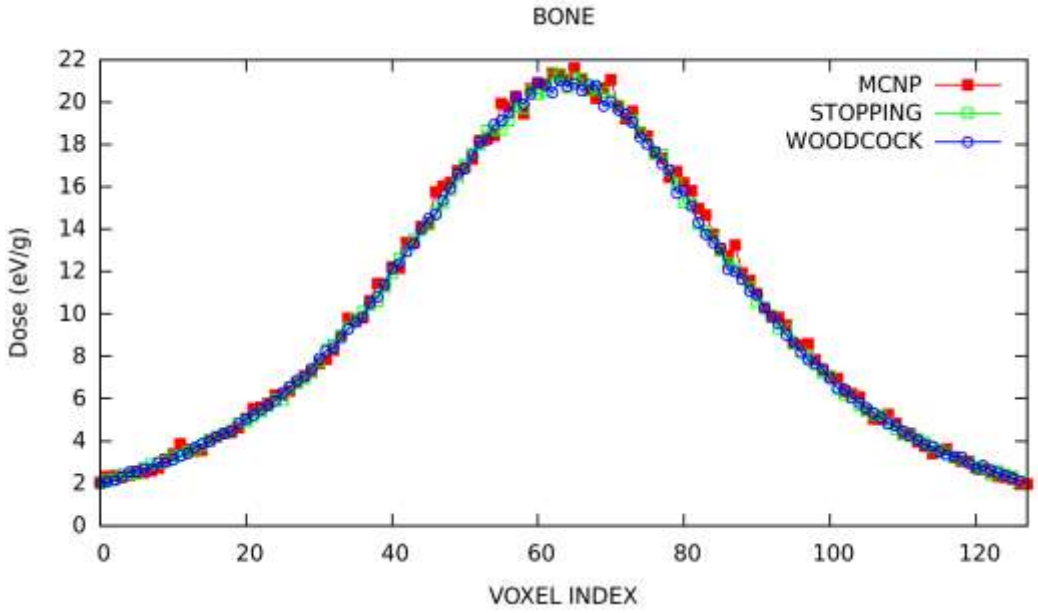

Figure 3.- Simulation for photons traveling on a bone based media. There were used 10^{10} particles on CUBMC simulations and $4x10^{8}$ particles on MCNP simulations.

Table 1 presents a comparison between the runtimes and speed-up factors of simulation in CUBMC and PENELOPE.

Table 1.- Comparison between runtimes on PENELOPE and CUBMC for homogeneous phantoms.

EXECUTION TIME					
	PENELOPE	CUBMC			
		Woodcock	Speed-up	Stopping	Speed-up
WATER	10051s	121s	83x	751s	13x
LUNG	9839s	210s	47x	684s	14x
BONE	36030s	478s	75x	2349s	15x

To establish a quantitative comparison between dose calculation on CUBMC and MCNP codes the maximum and average dose differences were calculated. The average dose differences were calculated as the arithmetic average of dose differences in each voxel. Table 2 presents the maximum and average dose differences obtained for each simulation mode, Woodcock and Stopping, in CUBMC code compared to those obtained with MCNP code.

Table 2.- Maximum and average dose differences between MCNP and CUBMC for homogeneous phantoms

DOSE DIFFERENCES				
	Woodcock		Stopping	
Media	max	Med	max	Med
WATER	7%	1.87 %	8%	1.89%
LUNG	11%	2.89%	9%	2.75%
BONE	14%	3.22%	14%	3.21%

The fourth case analyzed is the dose distribution in a heterogeneous phantom composed by bone-vacuum-bone-vacuum-bone intercalated slabs. The vacuum slices were on planes

Vx=[59:61] and Vx=[65:67]. Figure 4 shows the simulation results with CUBMC and MCNP.

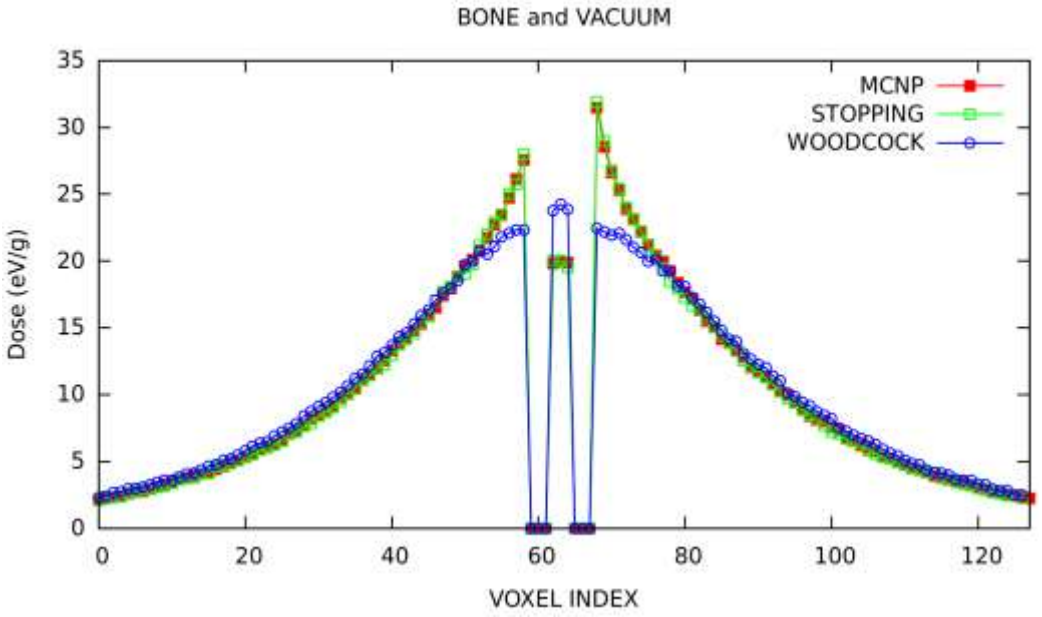

Figure 4.- Simulation for photons traveling in a bone and vacuum slab geometry. There were used 10^{10} particles on CUBMC simulations and $4x10^{8}$ particles on MCNP simulations.

As we can see, there is a good agreement between the results from the MCNP simulation and the Stopping method from CUBMC. However, the Woodcock method has a deviation from expected value dose when near to the edge between materials.

4.- DISCUSSION

The CUBMC code is still under development, but so far, it demonstrated to be in very good agreement with reference codes PENELOPE and MCNP. It contains a number of simplifications that will be corrected by a more complete and detailed models in the future versions of the code.

In general, there was a significant performance gain (Table 1) compared to the same simulations by PENELOPE, but what could be observed is that the performance gain is highly dependent on the material. In the worst case, it was achieved a performance gain of 50 times with the Woodcock method and more than 10 times using the Stopping method. Although this runtime comparison is done with different processors, it demonstrates that a typical simulation case taking 167 minutes in a 2.27 GHz intel Xeon processor with the PENELOPE code can be performed in less than 3 minutes in a modest GPU unit.

Although the routine for calculating the statistical accuracy of the dose is not yet implemented, the standard deviation for CUBMC results is estimated to be near 0.1%.

The average dose difference between CUBMC and MCNP results in the cases studied is lower than 3.5% and several other case studies are under way. Maximum voxel dose differences for water, lung and bone are 7%, 11% and 14%, respectively. Part of these differences is due to several simplifications used on the actual physics of CUBMC code and they should be reduced significantly after the implementation of more detailed physical models. Another known reason for those discrepancies is due to different cross-section libraries used in both codes.

Performance gain comparison with MCNP was not possible up to now and is under analysis using several other simulation cases. It was also observed that the simulations performed in MCNP have a much lower statistical fluctuation than simulations containing the same number of particles in PENELOPE or CUBMC. The reason for this behavior will be studied in a future work.

By introducing a heterogeneous medium containing vacuum, we can observe an average difference of 1.5 % between the MCNP code and the method of stops from CUBMC. However, the Woodcock method from CUBMC features a shift in the dose calculation for voxels near the boundary between materials, obtaining a deviation up to 28% for these voxels and the reason for such behavior is under review.

5.- CONCLUSIONS

Although CUBMC code is not yet fully functional, it was possible to observe that, the dose calculation for homogeneous media presents an average difference smaller than 3.5% from MCNP, which is acceptable by now due to the simplifications present on the current version of CUBMC code. The speed-up factors obtained on these simulations when compared with the same simulations on PENELOPE were of 50 to 100 times with the Woodcock method and of 10 to 15 times with the method of stops using a low cost GPU from NVIDIA, GTX560Ti.

The CUBMC code still needs to be improved to achieve more accurate results for simulations on heterogeneous media, and more detailed physical models for photons should be implemented for more realistic dose calculations. Although the code is still on development the results presented here already demonstrate its potential to become an option for fast Monte Carlo simulations for photons with practical utilization in treatment planning system in radiotherapy in the near future.

REFERENCES

Badal A; Sempau J. (2006). *A package of Linux scripts for the parallelization of Monte Carlo simulations*, Computer Physics Communications **175**: 440-450.

Brown FB; Sutton TM. (1996). *Monte Carlo fundamentals.* Knolls Atomic Power Laboratory Technical Report KAPL-4823.

Cashwell, ED; Everett CJ; Rechard OW. (1957). *A practical manual on the Monte Carlo method for random walk problems*. Los Alamos Scientific Laboratory report LA-2120.

Geant4 Collaboration. (2010). *Geant4 user's guide for application developers*. *CERN*.

Goorley JT; James MR; Booth TE; Brown FB; Bull JS; Cox LJ; Zukaitis AJ. (2013). *Initial MCNP6 release overview-MCNP6 version 1.0*.. Los Alamos National Laboratory report LA-UR-13-22934.

L'Ecuyer P. (1988). *Efficient and portable combined random number generators*. Commun. *ACM*31, 6 (June 1988), 742-751. DOI=10.1145/62959.62969.

NVIDIA. (2010). *CUDA Programming Guide- Version 3.0*.

Salvat F; Fernández-Varea JM; Sempau J. (2006). *PENELOPE-2006: A code system for Monte Carlo simulation of electron and photon transport*. In: Workshop Proceedings. 2006. **4**, p. 7.

Woodcock E; Murphy T; Hmmings P; Longworth S. (1965). *Techniques used in the GEM code for Monte Carlo neutronics calculation*. Proccedings of the Conference Applications of Computing Methods to Reactors, ANL-7050.