Electron tracks simulation in water: Performance comparison between GPU CPU and the EUMED grid installation

Edgard Seif, Ziad El Bitar, Sébastien Incerti, Mario Bernal, Ziad Francis

Electron tracks simulation in water: performance comparison between GPU CPU and the EUMED grid installation

Edgard Seif[a], Ziad El Bitar[b], Sébastien Incerti[c], Mario A Bernal[d], Ziad Francis[a]

[a]Saint Joseph University, UR Mathématiques et Modélisation, Beirut, Lebanon
[b]IPHC, UMR 7178—CNRS/IN2P3, Strasbourg, France
[c]CNRS, IN2P3, CENBG, UMR 5797, Gradignan, France
[d]Instituto de Fisica Gleb Wataghin, Universidade Estadual de Campinas, Campinas, SP, Brazil

Abstract

Purpose

We explored different technologies to minimize simulation time of the Monte-Carlo method for track generation following the Geant4-DNA processes for electrons in water.

Methods

A GPU software tool is developed for electron track simulations. A similar CPU version is also developed using the same collision models. CPU simulations were carried out on a single user desktop computer and on the computing grid France Grilles using 10 and 100 computing nodes. Computing time results for CPU, GPU, and grid simulations are compared with those using Geant4-DNA processes.

Results

The CPU simulations better performs when the number of electrons is less than 10^4 with 100eV initial energy, this number decreases as the energy increases. The GPU simulations gives better results when the number of electrons is more than 10^4 with initial energy of 100eV, this number decreases to 10^3 for electrons with 10KeV and increases back with higher energy. The use of the grid introduces an additional queuing time which slows down the overall simulation performance. Thus, the Grid gives better performance when the number of electrons is over 10^5 with initial energy of 10KeV, and this number decreases as the energy increases.

Conclusions

The CPU is best suited for small numbers of primary incident electrons. The GPU is best suited when the number of primary incident particles occupies sufficient resources on GPU card in order to get an important computing power. The grid is best suited for simulations with high number of primary incident electrons with high initial energy.

Key words: GPU, Monte-Carlo, electron, Geant4-DNA

Introduction

Monte Carlo simulations are widely used in many scientific fields especially in ionizing particle track structure and radiation physics simulations. The Geant4 Monte-Carlo simulation using the Geant4 toolkit [1-3] and the Geant4-DNA processes on CPU, hereafter referred to as Geant4-DNA processes, offers a wide range of applications from high energy physics to low energies and Geant4-DNA processes [4-7] were developed specifically for micrometric scale simulations, particularly useful in radiobiology and microdosimetry fields [8-11].

Studying the effects of radiation on living cells is important for the understanding of damage induction processes and this is a topic of interest in a wide range of applications, e.g., radiation therapy, medical imaging dosimetry, radiobiology, and radiation protection and risk assessment for space missions. The macroscopic results are easily identified, like living tissue alteration and cell survival quantifications. However, other stochastic effects are still nowadays an open field of research, like DNA scale damage, risks of mutation, mitochondrial damage and other radio induced end-effects problems. The challenge is that these radiation small-scale effects are difficult to quantify by experiment. Therefore, Monte-Carlo (MC) simulations are used to generate particle tracks in specific volume shapes representing the irradiated targets. Using simulations, we can assess the detailed energy deposition distribution in the irradiated medium with a high level of details.

In the last decade, many MC tools were developed for radiobiology purposes, e.g., Geant4-DNA[4], Partrac [12, 13] , and RITRACKS NASA software [14]. These types of simulations are computationally intensive and consequently time consuming. To improve the computational efficiency, parallel computing is a viable solution. Many available technologies provide different ways for parallelization like Grid computers and the use of Graphical Processing Units known as GPUs. This implies that users should have access to relatively large computing facilities and should have strong programming skills in order to carry out complex technical simulations. At the same time, it is difficult to predict the best technology to use for a simulation, as this would depend on many parameters and mainly on the solution's algorithm. In a specific scenario, one technology can achieve a better result; however, this might vary for different configurations.

Many research groups ported simulations on GPUs and proved a significant speedup over CPU [15-18]. In our case we are interested in comparing the computing performance between CPU, GPU, and Grid computing using *France Grilles* [19]. In this study, we focus on electrons tracks simulation, using the MC method for simulating the transport and energy deposition in liquid water. We followed a step-by-step tracking approach, since this method is convenient for nm-scale radiation biology simulations. Electrons should be taken into account in any ionizing radiation modelling since they are the most abundant secondary particles issued from charged particles collisions.

For the sake of comparison, we developed a CPU version of the code using the C++ language, and we validated the results with those obtained with Geant4-DNA. The same CPU code is compatible with *France Grilles* computing grid and thus can be used for performance comparisons.

We also ported our code on GPU using the CUDA C language [20] which allowed us to extend our comparison of computing time results. Therefore, we are comparing our CPU code running on a desktop machine and on computing grids, the GPU version and the Geant4-DNA processes (on CPU).

In order to make our tool easy to use, we developed a graphical user interface using the framework Qt [21]. This framework offers cross-platform solution for different programming languages like C++, Python and can be interfaced with other languages like CUDA.

Materials and methods

Hardware Specifications

The characteristics of the CPU used in our simulations are presented in table 1 and the GPU characteristics are presented in table 2.

Family	Model number	Frequency	Number of Cores	Number of threads	Memory Types
Intel Core i-9	i9-7940X	3.1 GHz	14	28	DDR4-2666

Table 1 : CPU characteristics

Name	Memory size	Memory Bandwidth	Nvidia CUDA Cores
Nvidia Quadro P4000	8GB	Up to 243 GB/s	1792

Table 2 : GPU characteristics

The grid is heterogenous by nature and nodes can be added, removed or modified at any time, therefore the hardware specifications are not constant and therefore cannot be presented here. Besides every simulation job sent to the grid is assigned to available nodes, even consecutive jobs with same number of computing nodes can be executed on different nodes with different specifications.

CPU algorithm

Our simulations are a step-by-step path tracking of a single electron that are repeated N times. All electrons have the same initial energy, coordinates and direction. The processes that an electron undergoes are elastic collisions, molecular vibrational excitation, electronic excitation, and ionization. All processes are limited to tracking in liquid water, this limitation arises from theoretical models of cross sections that are available for this type of studies.

For each step, the electron can go through one interaction process that is chosen using a random sampling taking into account the respective total cross sections of each process. The free path which is the distance to the next collision is calculated using the following formula: $\lambda = \frac{-\ln(\varepsilon)}{\Sigma}$ where ε is a uniform random value between 0 and 1, and Σ is the sum of the 4 processes macroscopic cross sections.

Elastic collisions

The elastic collision angular and total cross sections are based on the Brenner and Zaider [22] semi empirical approach for energies below 200 eV, and we used the Rutherford formula for higher incident energies [23, 24]. This method works well for electrons with non-relativistic energy, however it doesn't impose any limit for high energy simulations because in the relativistic range all the interactions become highly dominated by ionization.

The direction vector after an elastic collision is calculated using the following expressions:

$$U = V_x \cos(\theta) + \frac{\sqrt{1 - \cos(\theta)^2}\left(V_x V_z \cos(\varphi) - V_y \sin(\varphi)\right)}{\sqrt{1 - {V_z}^2}}$$

$$V = V_y \cos(\theta) + \frac{\sqrt{1 - \cos(\theta)^2}\left(V_y V_z \cos(\varphi) - V_x \sin(\varphi)\right)}{\sqrt{1 - {V_z}^2}}$$

$$W = V_z \cos(\theta) - \sqrt{1 - \cos(\theta)^2}\sqrt{1 - {V_z}^2}\cos(\varphi)$$

where $V_x, V_y \; and \; V_z$ are the components of the electron's direction vector before the collision. The azimuthal angle φ is randomly chosen between 0 and 2π and the polar angle θ is obtained using a random sampling taking into account the differential cross sections from Brenner and Zaider [22].

Vibrational excitations

Due to the lack of theoretical models characterizing energy loss through vibrational excitations, we used interpolated experimental data of Michaud and Sanche [25, 26]. Vibrational excitations become increasingly important for low energies and mostly for subexcitational energies below ~8 eV because in this range they are the only process involved in the electron's energy loss. The data of Michaud and Sanche [25, 26] cover energies between 1 eV and 100 eV. Direction change is not taken into account after a vibrational interaction and the energy loss is randomly sampled taking into account the different vibrational levels. The energy lost by the electron is saved as a local energy deposition in the medium.

Inelastic collisions

Inelastic collisions including electronic excitation and ionization were described using the First Born Approximation theory as detailed by Emfietzoglou [27]. We calculated the single differential cross sections and the integrated total cross sections for different electrons incident energies, and the computed numerical tables were implemented in our code. The specific cross section values are obtained during the simulation process using a linear interpolation between the precomputed values. This method avoids using the theoretical formulas during the simulation process which can have a large effect on the simulation time. After excitation interactions the electron loses part of its energy without changing direction. After an ionization the incident electron loses part of its energy that is distributed between a local energy loss deposited in the medium, the electron binding energy, and the kinetic energy that is communicated to a secondary electron. The energy transferred to the secondary electron is randomly sampled using the precomputed differential cross sections. The direction of the secondary ejected electron is randomly sampled with respect to models published by Grosswendt et al. [28]. The new direction of the primary electron is calculated taking into account the momentum conservation law with respect to the secondary electron ejection momentum. The binding energy of the ejected secondary electron is considered locally deposited in the medium. Secondary electrons are then tracked just like any primary electron, depositing energies and creating other secondary electrons.

Technically the position, energy, and direction of each secondary electron are firstly stored in a queue to be used later on when the primary electron tracking is finished. A cut-off energy limit is set, if the

electron's energy drops below this limit, the electron is forced to deposit all its remaining kinetic energy in its current position and the tracking is ended. In our, simulations this limit was set to 8.22 eV which corresponds to the lowest water excitation energy.

All the calculations use extensively random values generation. Therefore, the use of a robust Random Number Generator (RNG) is key factor of the simulations results. In our CPU simulations we use the Mersenne-Twister RNG [29] because it provides a large period of random numbers, it's a validated model and it's widely used in Monte Carlo simulations. Moreover, it is now part of the C++ 11 standard library [30].

The algorithm diagram for the CPU version of our code is summarized in figure 1. First the primary electron's energy, position, and initial direction are set, then as long as the kinetic energy of the electron is above our cut-off threshold the tracking process is carried out and energy deposition data is saved in a text file. The whole process is repeated to simulate a fixed number of electron tracks. In fact, since the approach is stochastic, a large number of primary electrons is needed so that the results analysis of energy depositions in a limited volume will converge to an average value.

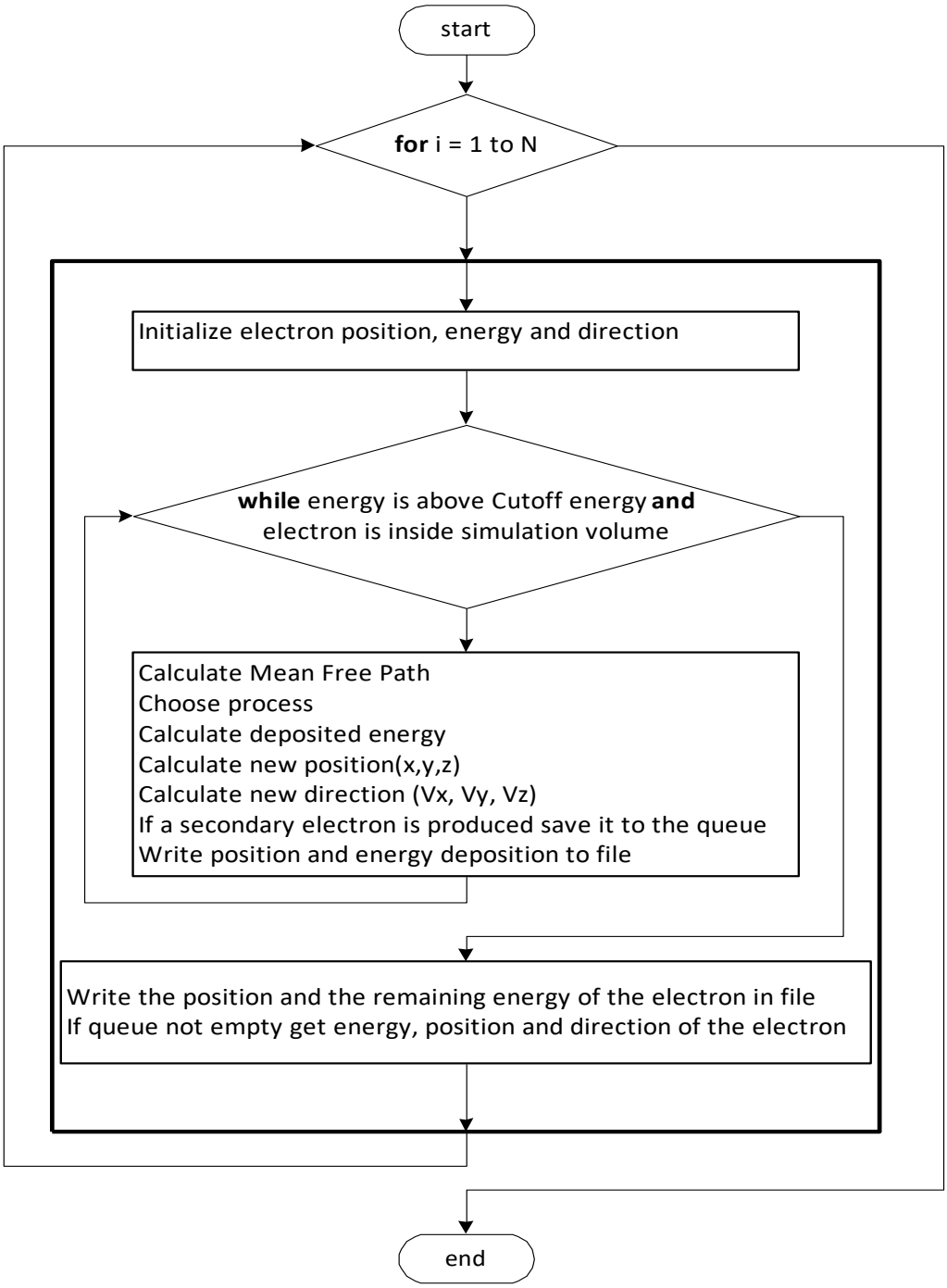


Figure 1: The algorithm pseudocode used for our CPU version.

GPU algorithm

GPU processing enables us to launch multithreaded tasks on a large number of cores. Therefore, the step-by-step path tracking of N electrons can be carried out simultaneously.

We used the Compute Unified Device Architecture (CUDA), as the language for our GPU code. The algorithm is hybrid, it is launched from the Host in a loop until all electrons' calculation carried out on the Device are completed.

The GPU simulation follows the same steps described in the previous section for CPU. First, we calculate processes cross sections, then the free path distance of the electron to the next collision, the deposited energy, the electrons' new direction and, for ionization interactions, the energy transferred to the secondary electron and its direction. The secondary electron's information is saved in a queue on the host memory and its simulation starts as soon as a thread is free and ready to start with a new electron tracking. We always keep the maximum number of threads busy to have the best performance results.

Cross section tables used for different processes are copied to the device global memory before starting the simulation. Although, the shared memory is usually a faster alternative, in our case it would require replicating these tables for each block which would consume a large memory space, moreover, shared memory is smaller in size and its content is reset at the end of the block execution [20]. The different electron variables are stored in the local memory.

At each step the position and the amount of deposited energy are written in a text file, which can only be done from the host memory. Therefore, all the energy deposition data was transferred from the device to the host after each step.

A CUDA stream encapsulates data transfer and kernel launch[20, 31]. Our workload can be equally split into up to 8 streams, based on our hardware capacity, in this way we can overlap data transfer and device calculation which leads to better performance. Our code implementation allows the user to choose the number of threads per block (T) and the number of blocks (B) as well as the number of streams (S). So, we can launch N = T x B x S electron tracks simulation in parallel.

The cuRAND library provides different random number generators [32]. The MTGP32 generator is an adaptation of the Mersenne-Twister developed at Hiroshima University [33], but it does not scale enough for our simulations; 200 sequences with 256 threads per block at most [31]. We used the Philox_4x32_10 generator that has 2^{64} subsequences each with a period of 2^{128} which yield good results. We also tried the Thrust library [34] random number generator which also led to satisfactory results.

Grid Parallel Computing

The grid concept was formulated in 1999 by Ian Foster and Carl Kesselman [35]. The French national grid is France Grilles [19] and is a part of the European Grid Infrastructure federation (EGI) [36]. To be able to access the grid you need an X509 certificate for authentication delivered by a trusted organization, in our case it's RENATER [37], and you need to be part of a Virtual Organization (VO) to have permission to use the resources, in our case we were part of the "grand-est" VO.

The grid infrastructure is best suited for jobs that can be parallelized with no need for communication between the jobs. In our case each electron simulation is completely independent from the others, and we used K nodes for the simulation of N electrons, therefore, the workload of each node is N/K.

On the grid, we should take special care of the random number generation. Using a different seed value on each computing node may result in overlapping value sequences leading to repeated results. Another

approach is to use the same seed on all computing nodes, but instead of using the random values in sequence, we skip ahead k values each time [38]. This way, the simulation algorithm used on the grid is the same used on the CPU of a single machine. The only change is in the random number generation, a skip ahead instruction is added to the algorithm. For each job we start by a p skip ahead, where p is the position of this job with respect to the total number of jobs. Subsequently a k jump ahead is used for every random value generation, where k is the total number of jobs on the grid.

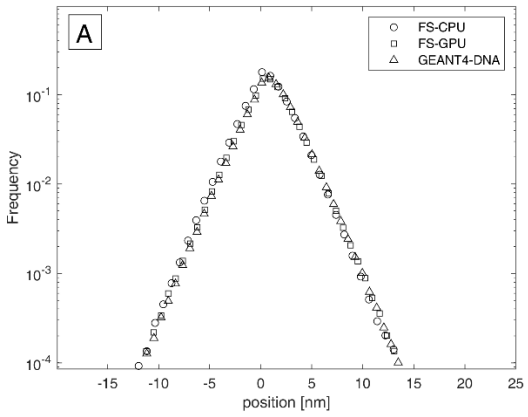
Sending jobs to the grid and getting back the results requires a special set of command packaged in software framework DIRAC (Distributed Infrastructure with Remote Agent Control) [39]. The jobs sent to the grid go through the following major values of status: Waiting (Job is accepted for DIRAC Workload Management System), Scheduled (Job is assigned to a Site), Running (the job is executing on the Computing Element), Done (Job finished successfully), Failed (Job finished with errors), Deleted (Job deleted by the user) and Killed (Job killed by the user). We used python scripts with a set of DIRAC commands, to send our simulation jobs to the grid, and get the jobs logs and results.

We launched all our simulations on the grid two times, the first time using 10 nodes and the second time 100 nodes. This choice helped us to highlight the impact of the waiting time in grid simulation and the total simulation time.

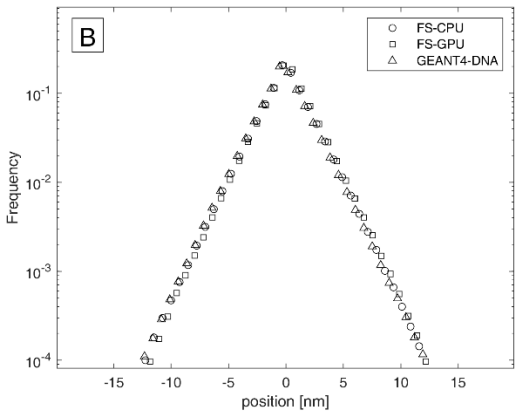
Results and Discussion

Track structure simulations of electrons with energies 100 eV, 1 keV, and 10keV were carried out in infinite volume of water and electrons were followed until their kinetic energy dropped below the 8.22 eV cut-off. All electrons start from the origin in the positive direction of the x axis. Energy deposition positions in the volume were saved, and their coordinates are shown in figures 2, 3 and 4. Coordinates are plotted into a histogram for each spatial component (figures A, B and C). Also, the energy deposition distribution is represented showing the occurrences of energy loss values in the medium (figure D) for different interaction types. Figures D reveal the 9 energy depositions coming from vibrational and rotational excitations of the water molecules (below 1 eV) and 5 electronic excitation values (between 8 and 14 eV) as well as 5 ionization values (above 10 eV). The results obtained using both the CPU and GPU running versions, were compared with the same simulation conditions using Geant4-DNA track structure processes [4-7]. A good agreement is obtained between the spatial coordinates' components along the 3 axis directions. Also, the energy deposition frequencies were compared showing good agreement with the results of Geant4-DNA.

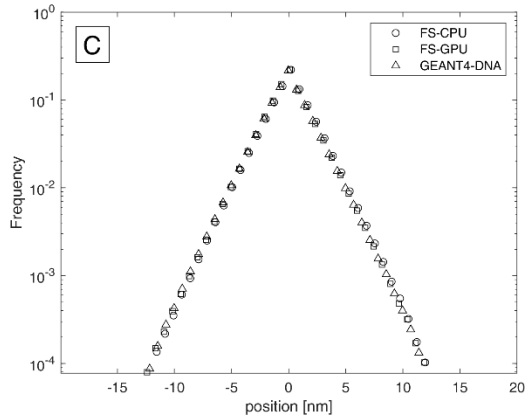
Geant4 simulations were carried out using Geant4-DNA physics processes, for 4 different interactions; ionization, electronic excitation, elastic scattering and vibration and rotation excitations. The processes rely on the First Born Approximation for inelastic collisions [27] such as ionization and electronic excitation. The elastic scattering process uses the Rutherford elastic model [23, 24], and the vibration and rotation excitations cross sections are based on interpolations of experimental data of Michaud and Sanche [25, 26]. The electron source is placed at the center of the reference and the initial direction of the electrons is directed along the x axis.



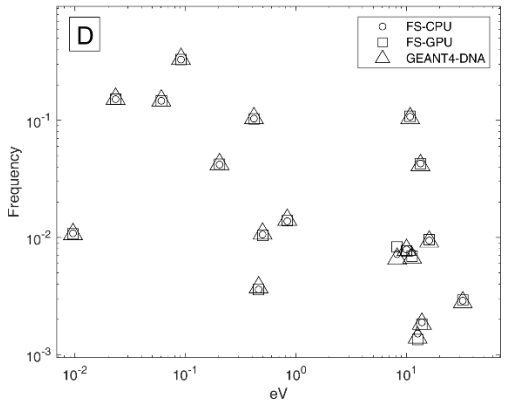
Histogram of X-axis positions



Histogram of Y-axis positions

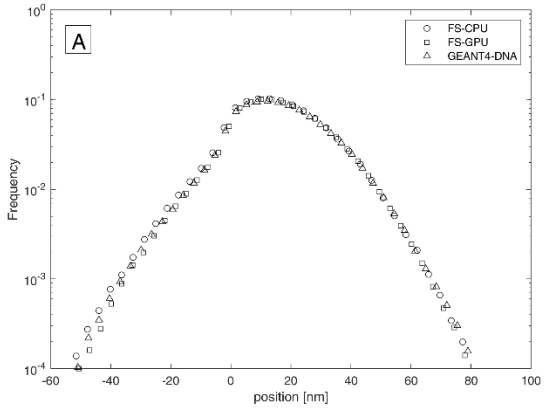


Histogram of Z-axis positions

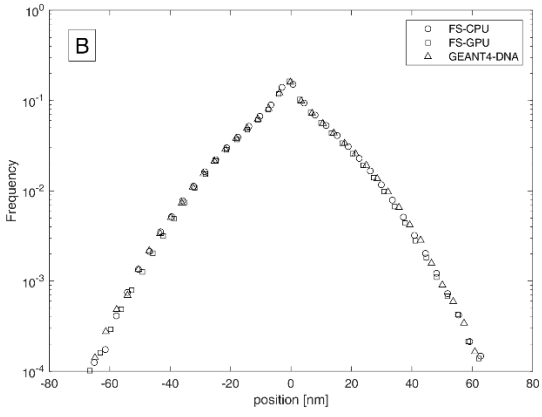


Histogram of energy depositions

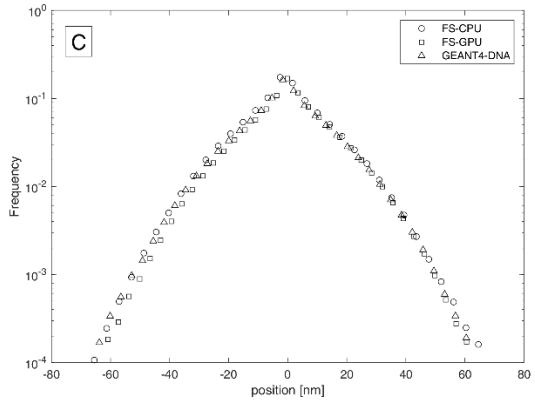
Figure 2 : Energy deposition coordinates (A, B, and C) and frequencies (D) for 100 eV electrons in liquid water. Results were obtained using CPU and GPU versions of our software and compared with Geant4-DNA simulations.



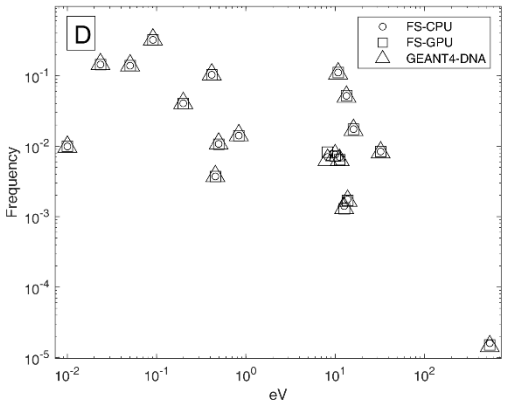
Histogram of X-axis positions



Histogram of Y-axis positions

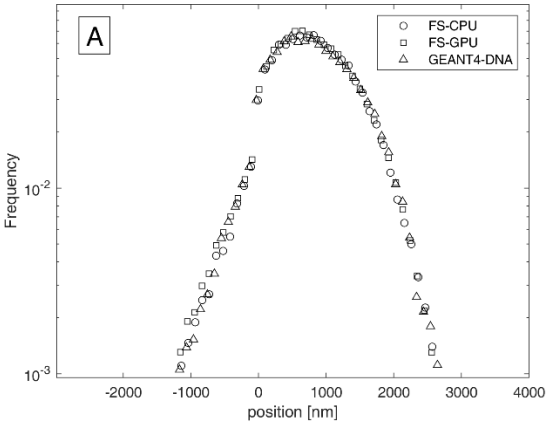


Histogram of Z-axis positions

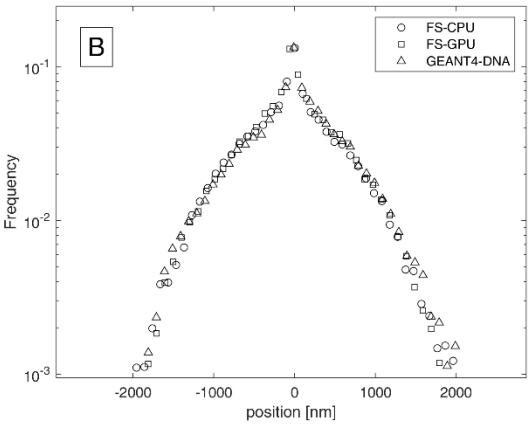


Histogram of energy depositions

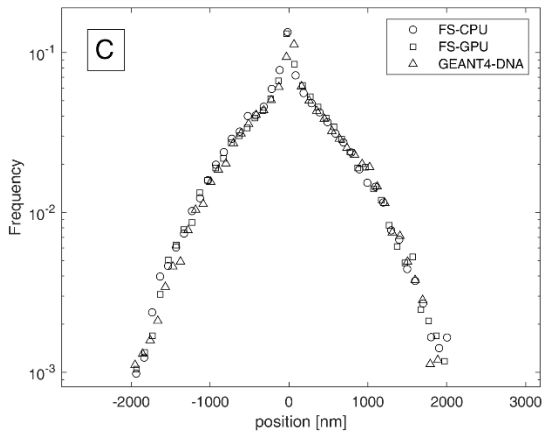
Figure 3 : Energy deposition coordinates (A, B, and C) and frequencies (D) for 1 keV electrons in liquid water. Results were obtained using CPU and GPU versions of our software and compared with Geant4-DNA simulations.



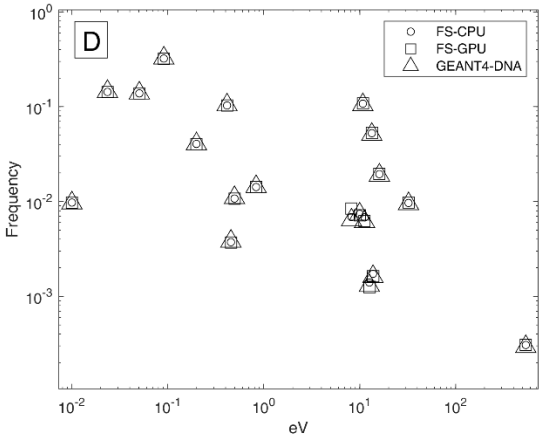
Histogram of X-axis positions



Histogram of Y-axis positions



Histogram of Z-axis positions



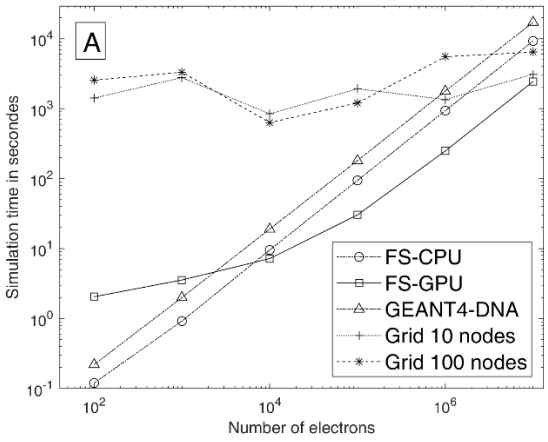
Histogram of energy depositions

Figure 4 : Energy deposition coordinates (A, B, and C) and frequencies (D) for 10 keV electrons in liquid water. Results were obtained using CPU and GPU versions of our software and compared with Geant4-DNA simulations.

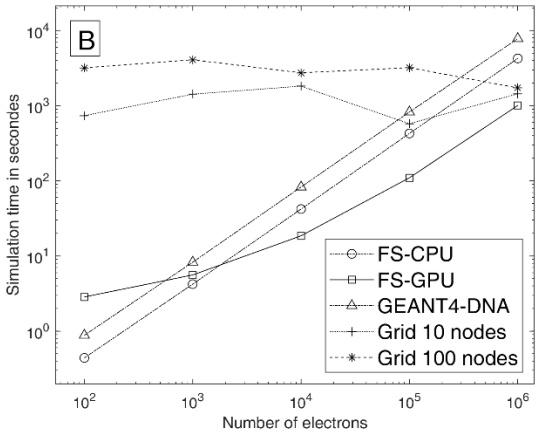
Simulations were carried out on an Intel I9 CPU with a clock speed of 3.10 GHz, 32 GB of RAM and a Nvidia Quadro P4000 GPU. CPU and Geant4 simulations were single threaded. For GPU simulations, we used a configuration of 512 block, 32 thread by block and 8 streams for information transfer between the host and the device. For simulations on the computing grid "France Grilles", we used 10 and 100 calculation nodes. Simulation times were measured for the same number of incident electrons and for different electron energies, 100 eV, 500 eV, 1 keV, 10 keV, 100 keV, 500 keV and 1 MeV. Results are reported in Figure 6 for 4 different energies. The figure shows calculation time versus the number of initial electrons for different incident energies (100 eV, 500 eV, 1 keV and 10 keV). The results include CPU and GPU versions of our code, the Geant4-DNA processes, and Grid calculations using 10 and 100 nodes.

The results show a linear increase of computing time for CPU and Geant4-DNA simulations. GPU computing time shows slower performance for low electron number simulations and a faster performance for high numbers of electrons, e.g., above 10^4 electrons for 100 eV energies, 10^3 electrons for 10 keV electrons. This is due to the fact that GPU clock frequency is generally slower than CPU frequency and low electrons numbers do not occupy the total resources of the GPU until a certain number of electrons is reached where multithreading becomes more rewarding. Once the total capacity of the GPU is being used, its best performance is reached, and the computing time increases linearly with the number of initial electrons. Our GPU acceleration results are in agreement with other published studies [40]. The GPU performance is highly affected by data transfer between the GPU, the host system and the local hard drive for final results, which slows down the simulation process. Also, conditions branching contribute to slowing the GPU computing process. Therefore, depending on the volume of the data to be extracted and saved on the local hard drive, we might get different acceleration ratios with different simulations.

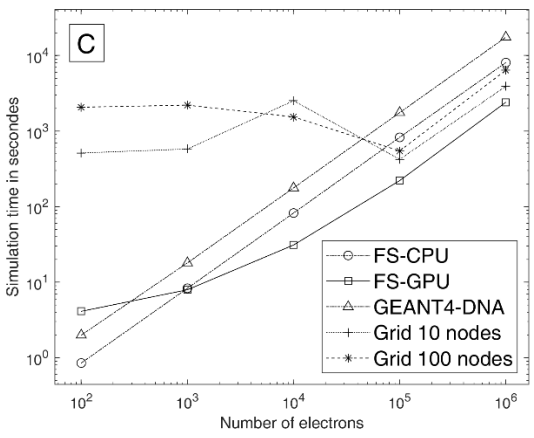
The performance of the CPU version is comparable to Geant4-DNA, while the GPU brings an advantage above 10^4 events and reaches a maximum ratio of 10 for higher numbers.



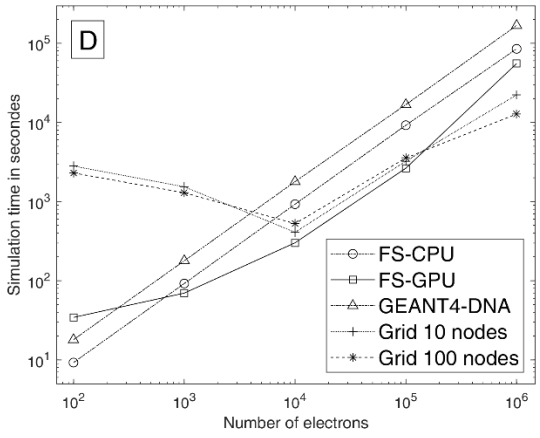
Electrons initial energy 100 eV



Electrons initial energy 500 eV

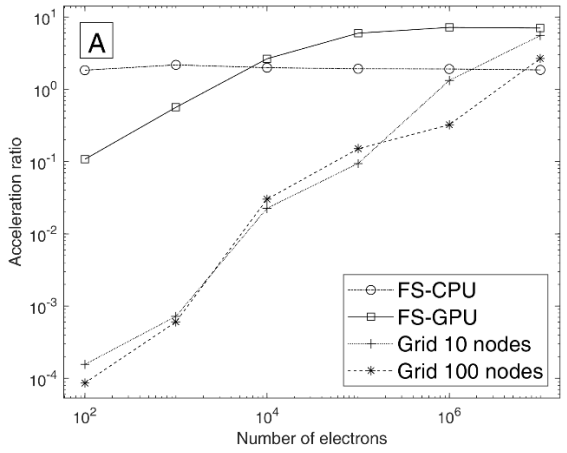


Electrons initial energy 1keV

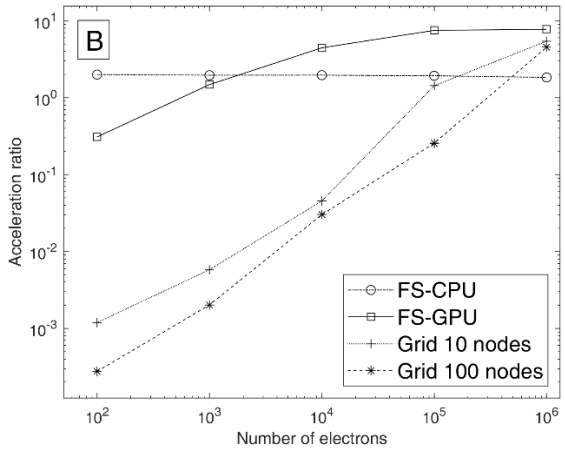


Electrons initial energy 10 keV

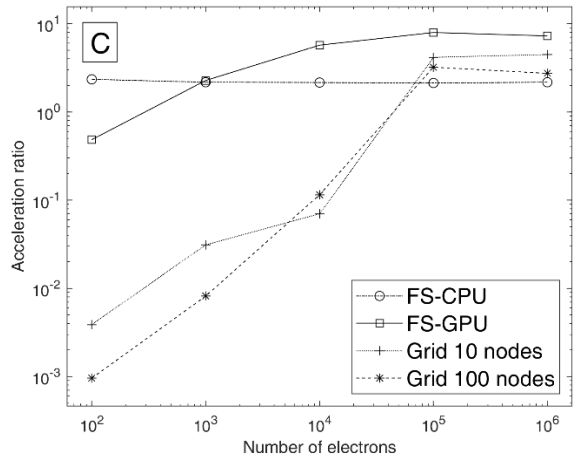
Figure 5: Simulation time comparison for electrons of different energies, using different simulation tools: CPU and GPU versions of this work, Geant4-DNA processes and France Grille computing grid using 10 and 100 nodes.



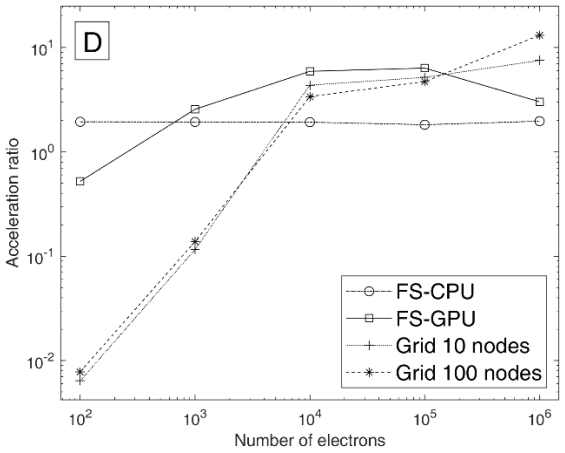
Electrons initial energy 100 eV



Electrons initial energy 500 eV



Electrons initial energy 1 keV



Electrons' initial energy 10 keV

Figure 6: Acceleration ratios with respect to Geant4-DNA

Number of electrons	FS-CPU	FS-GPU	Grid 10 nodes	Grid 100 nodes
10^2	1.9021	0.1082	0.0002	0.0001
10^3	2.1784	0.5618	0.0007	0.0006
10^4	1.9871	2.6426	0.0225	0.0301
10^5	1.9178	5.9677	0.0941	0.1501
10^6	1.9047	7.1713	1.3232	0.3224
10^7	1.8520	7.0275	5.5347	2.6683

Table 3: acceleration ratios for 100 eV electrons with respect to Geant4-DNA. Different numbers of incident electrons are used on CPU, GPU and computing grid configurations using 10 and 100 nodes.

Number of electrons	FS-CPU	FS-GPU	Grid 10 nodes	Grid 100 nodes
10^2	2.0256	0.3093	0.0012	0.0003
10^3	1.9580	1.4795	0.0058	0.0020
10^4	1.9632	4.4576	0.0454	0.0302
10^5	1.9281	7.5066	1.4316	0.2566
10^6	1.8363	7.7547	5.4212	4.5349

Table 4: acceleration ratios for 500 eV electrons with respect to Geant4-DNA. Different numbers of incident electrons are used on CPU, GPU and computing grid configurations using 10 and 100 nodes.

Number of electrons	FS-CPU	FS-GPU	Grid 10 nodes	Grid 100 nodes
10^2	2.3437	0.4842	0.0039	0.0010
10^3	2.1676	2.2699	0.0311	0.0082
10^4	2.1516	5.7060	0.0700	0.1149
10^5	2.1249	7.9430	4.1537	3.2121
10^6	2.1680	7.2607	4.4688	2.7344

Table 5: acceleration ratios for 1 KeV electrons with respect to Geant4-DNA. Different numbers of incident electrons are used on CPU, GPU and computing grid configurations using 10 and 100 nodes.

Number of electrons	FS-CPU	FS-GPU	Grid 10 nodes	Grid 100 nodes
10^2	1.9447	0.5227	0.0064	0.0078
10^3	1.9384	2.5626	0.1163	0.1383
10^4	1.9288	5.9233	4.3653	3.3943
10^5	1.8221	6.3610	5.1831	4.7183
10^6	1.9710	3.3474	7.5245	13.1163

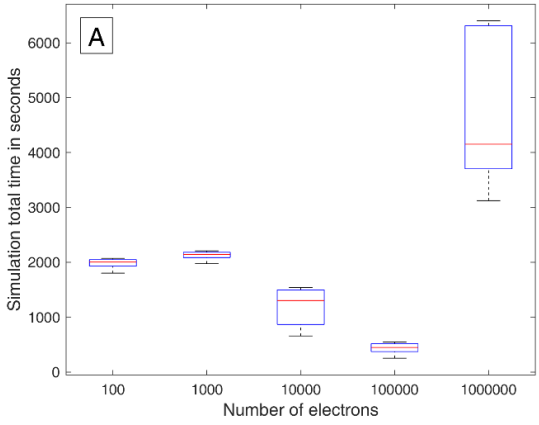
Table 6: acceleration ratios for 10 KeV electrons with respect to Geant4-DNA. Different numbers of incident electrons are used on CPU, GPU and computing grid configurations using 10 and 100 nodes.

The simulation time on the grid is the time that separates the reception of the first job in the queue manager and the time the last job is finished. The grid results are greatly influenced by the waiting time in the queue manager, caused by the number of jobs that are scheduled by all users of the grid. Therefore, the waiting time becomes negligible for long computing time simulations. Also, the calculation time is not the same on all the grid's nodes because the hardware is not the same, therefore, the simulation might be partially slowed down by one slow hardware. Figure 7 shows the variations of the simulation total time (A), waiting time (B) and CPU time (C), for the initial electrons' energy of 1 keV simulation, on 100 nodes. Comparing the total simulation time and the waiting time suggests that this latter has the higher influence

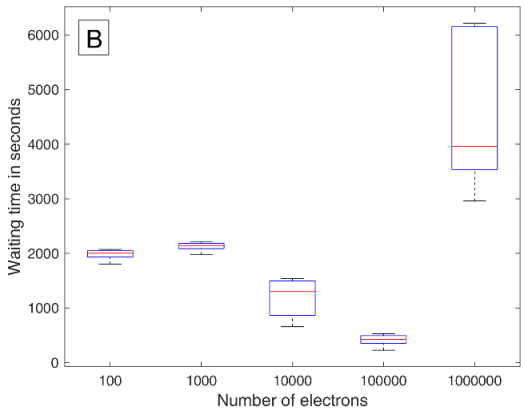
on measured performances. Note that this is only true for simulations where CPU time is of the same order of waiting time, and this influence becomes increasingly negligible for larger simulations requiring longer CPU time. Figure 7 (C) shows a linear increase of CPU time with increasing number of electrons, however there is no clear trend for the waiting time as shown on Figure 7 (B).

We assume that the grid becomes increasingly advantageous when the simulation computing time exceeds the queuing time. For our simulations on 10 nodes configuration, the threshold is attained with 10^6 electrons of 10 keV energy where the simulation time was about ~16000 seconds and the queuing time was over ~8000 seconds. For lower number of electrons or lower incident energies the overall time is dominated by the queuing component. While on 100 nodes configuration this threshold is attained for 1000 electrons of 500 keV energy. The simulation becomes increasingly demanding for higher energies and the queuing time increases with required nodes as well, therefore higher nodes configurations are only advantageous for CPU demanding simulations.

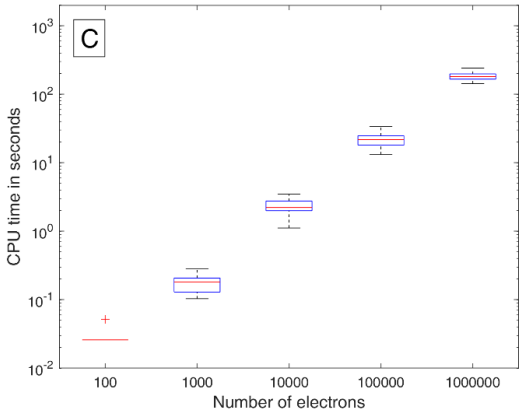
Queuing time is a random parameter that depends on the grid state and users' number. Figure 8 shows the waiting time distribution for jobs sent on 10 and 100 nodes. We can notice that 75% of all jobs have a waiting time less than 3000 seconds using 10 nodes and less than 4000 seconds for 100 nodes jobs. Thus, simulations requiring CPU time above these queuing time thresholds can be accelerated using the grid facility.



Total simulation time variation on 100 nodes



Waiting time variation on 100 nodes



CPU time variation on 100 nodes

Figure 7: Grid 1 KeV electron initial energy simulation, total, waiting, and CPU time

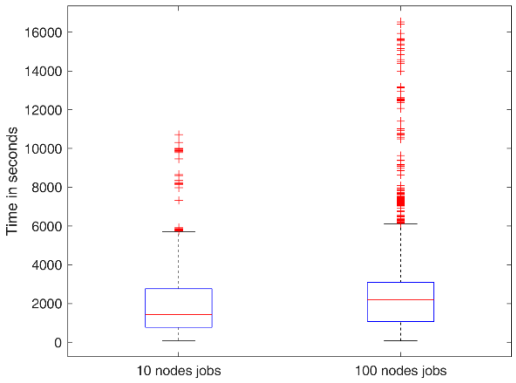


Figure 8 : Queuing time distribution for 10 and 100 nodes configurations using the computing grid.

Conclusion

In this work, we have presented a comparison between 3 technical approaches for Monte-Carlo ionizing track structure simulations. A GPU version of track structure processes was developed with a user-friendly graphical interface. Comparison with CPU simulations showed a clear advantage of GPU multicore usage. Also, comparisons with Geant4 simulations on CPU using a single thread showed that GPU had an advantage for high events simulations with an acceleration reaching 8 times faster. The Geant4 simulations were carried out using one thread to compare with our CPU single threaded performance. Multithreading both Geant4 and our CPU code would simply divide the computing time by the number of threads and the computing time is linearly scalable. However, taking into account that Geant4 has a multithreading enabled version and considering that the simulation time decreases linearly with increasing number of available threads, multithreaded Geant4 simulations can accelerate the computing time by a factor of ~28 for example on a I9-7940X. Using a large computing grid is not practical for simulations requiring less than few hours CPU time, mainly because of the technical complexity of submitting jobs and queuing times that might be lengthy depending on the grid's workload. However, for large number of electrons and higher initial energies, simulations require more than 2~3 hours of CPU time, grid usage becomes more advantageous than GPU. Therefore, depending on the availability of the hardware, GPUs present a cheap and practical solution for simulations of up to 2~3 hours. Along with multithreaded CPU calculations, GPU can be an adapted alternative to large computing grids which might not be easily accessible at times, or access protocol complexity might slow down the overall process. Finally, the choice of cross sections was solely based on comparison requirements with the Geant4-DNA package. Hence, the cross sections integrated in our GPU simulation tool are the same as in Geant4-DNA. However, both the inelastic and the elastic cross sections models become inaccurate for low energies, thus increasing the results uncertainty. In the future, the Rutherford model can be replaced with the partial wave expansion to improve the elastic scattering collisions, in fact recent studies showed the importance of elastic scattering models and their effect on numerical simulations [41, 42].

Acknowledgment

The authors would like to thank the research council of the Saint Joseph University for their support, France Grilles for providing computing resources on the French National Grid Infrastructure, the Institut Pluridisciplinaire Hubert Curien in Strasbourg, France, for the technical discussions, and the Geant4-DNA collaboration. M. A. Bernal thanks the FAPESP foundation in Brazil for financing his research activities through the projects 2011/51594-2, 2015/21873-8, 2018/15316-7, and 2020/08647-7. In addition, he acknowledges the financial support received from the CNPq through the 306298/2018-0 fellowship.

References

[1] Agostinelli S, Allison J, Amako K, Apostolakis J, Araujo H, Arce P, et al. Geant4—a simulation toolkit. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 2003;506:250-303. https://doi.org/10.1016/S0168-9002(03)01368-8
[2] Allison J, Amako K, Apostolakis J, Araujo H, Arce Dubois P, Asai M, et al. Geant4 developments and applications. IEEE Transactions on Nuclear Science. 2006;53:270-8. https://doi.org/10.1109/tns.2006.869826
[3] Allison J, Amako K, Apostolakis J, Arce P, Asai M, Aso T, et al. Recent developments in Geant4. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 2016;835:186-225. https://doi.org/10.1016/j.nima.2016.06.125
[4] Incerti S, Baldacchino G, Bernal M, Capra R, Champion C, Francis Z, et al. THE GEANT4-DNA PROJECT. 2010;01:157-78. https://doi.org/10.1142/s1793962310000122
[5] Incerti S, Ivanchenko A, Karamitros M, Mantero A, Moretto P, Tran HN, et al. Comparison of GEANT4 very low energy cross section models with experimental data in water. Med Phys. 2010;37:4692-708. https://doi.org/10.1118/1.3476457
[6] Bernal MA, Bordage MC, Brown JMC, Davídková M, Delage E, El Bitar Z, et al. Track structure modeling in liquid water: A review of the Geant4-DNA very low energy extension of the Geant4 Monte Carlo simulation toolkit. Physica Medica. 2015;31:861-74. https://doi.org/10.1016/j.ejmp.2015.10.087
[7] Incerti S, Kyriakou I, Bernal MA, Bordage MC, Francis Z, Guatelli S, et al. Geant4-DNA example applications for track structure simulations in liquid water: A report from the Geant4-DNA Project. Med Phys. 2018. https://doi.org/10.1002/mp.13048
[8] Fang Q, Boas DA. Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units. Opt Express. 2009;17:20178-90. https://doi.org/10.1364/OE.17.020178
[9] Chouin N, Bernardeau K, Davodeau F, Chérel M, Faivre-Chauvet A, Bourgeois M, et al. Evidence of Extranuclear Cell Sensitivity to Alpha-Particle Radiation Using a Microdosi Metric Model. I. Presentation and Validation of a Microdosimetric Model. Radiation Research. 2009;171:657-63. https://doi.org/10.1667/RR1371.1
[10] Jia Y, Lin ZW. The Radiation Environment on the Moon from Galactic Cosmic Rays in a Lunar Habitat. Radiation Research. 2010;173:238-44. https://doi.org/10.1667/RR1846.1
[11] Emfietzoglou D, Kyriakou I, Garcia-Molina R, Abril I, Nikjoo H. Inelastic Cross Sections for Low-Energy Electrons in Liquid Water: Exchange and Correlation Effects. Radiation Research. 2013;180:499-513. https://doi.org/10.1667/RR13362.1
[12] Friedland W, Dingfelder M, Kundrát P, Jacob P. Track structures, DNA targets and radiation effects in the biophysical Monte Carlo simulation code PARTRAC. Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis. 2011;711:28-40. https://doi.org/10.1016/j.mrfmmm.2011.01.003

[13] Friedland W, Schmitt E, Kundrat P, Dingfelder M, Baiocco G, Barbieri S, et al. Comprehensive track-structure based evaluation of DNA damage by light ions from radiotherapy-relevant energies down to stopping. Sci Rep. 2017;7:45161. https://doi.org/10.1038/srep45161
[14] RITRACKS: A software for simulation of stochastic radiation track structure, micro- and nano-dosimetry, radiation chemistry and DNA damage by heavy ions https://software.nasa.gov/software/MSC-25937-1; [accessed 17 September 2022]
[15] Kirk DB, Hwu W-mW. Programming Massively Parallel Processors, 2nd Edition: A Hands-on Approach: Morgan Kaufmann Publishers Inc.; 2012.
[16] Hofinger S, Acocella A, Pop SC, Narumi T, Yasuoka K, Beu T, et al. GPU-accelerated computation of electron transfer. J Comput Chem. 2012;33:2351-6. https://doi.org/10.1002/jcc.23082
[17] Bert J, Perez-Ponce H, El Bitar Z, Jan S, Boursier Y, Vintache D, et al. Geant4-based Monte Carlo simulations on GPU for medical applications. Phys Med Biol. 2013;58:5593-611. https://doi.org/10.1088/0031-9155/58/16/5593
[18] Tian Z, Jiang SB, Jia X. Accelerated Monte Carlo simulation on the chemical stage in water radiolysis using GPU. Phys Med Biol. 2017;62:3081-96. https://doi.org/10.1088/1361-6560/aa6246
[19] FRANCE GRILLES http://www.france-grilles.fr/home/; [accessed 14 September 2022]
[20] John Cheng MG, Ty McKercher. Professional CUDA C Programming: Wrox Press Ltd.; 2014.
[21] The Qt Framework https://www.qt.io/; [accessed 14 September 2022]
[22] Brenner DJ, Zaider M. A computationally convenient parameterisation of experimental angular distributions of low energy electrons elastically scattered off water vapour. Physics in Medicine and Biology. 1984;29:443-7. https://doi.org/10.1088/0031-9155/29/4/015
[23] Bransden BH. Atomic Collision Theory: W. A. Benjamin; 1970.
[24] Emfietzoglou D, Papamichael G, Kostarelos K, Moscovitch M. A Monte Carlo track structure code for electrons (approximately 10 eV-10 keV) and protons (approximately 0.3-10 MeV) in water: partitioning of energy and collision events. Phys Med Biol. 2000;45:3171-94. https://doi.org/10.1088/0031-9155/45/11/305
[25] Michaud M, Sanche L. Total cross sections for slow-electron (1-20 eV) scattering in solid H2O. Phys Rev A Gen Phys. 1987;36:4672-83. https://doi.org/10.1103/physreva.36.4672
[26] Michaud M, Wen A, Sanche L. Cross Sections for Low-Energy (1–100 eV) Electron Elastic and Inelastic Scattering in Amorphous Ice. 2003;159 Radiation Research:3-22, 0. https://doi.org/10.1667/0033-7587(2003)159[0003:csflee]2.0.co;2
[27] Emfietzoglou D, Karava K, Papamichael G, Moscovitch M. Monte Carlo simulation of the energy loss of low-energy electrons in liquid water. Phys Med Biol. 2003;48:2355-71. https://doi.org/10.1088/0031-9155/48/15/308
[28] Grosswendt B, Waibel E. Transport of low energy electrons in nitrogen and air. Nuclear Instruments and Methods. 1978;155:145-56. https://doi.org/10.1016/0029-554X(78)90198-2
[29] Matsumoto M, Nishimura T. Mersenne twister. ACM Transactions on Modeling and Computer Simulation. 1998;8:3-30. https://doi.org/10.1145/272991.272995
[30] Random Number Generation in C++11 https://isocpp.org/files/papers/n3551.pdf; [accessed 14 September 2022]
[31] CUDA Toolkit Documentation https://docs.nvidia.com/cuda/index.html; [accessed 14 September 2022]
[32] cuRAND (The API reference guide for cuRAND, the CUDA random number generation library) https://docs.nvidia.com/cuda/archive/10.2/curand/index.html; [accessed 14 September 2022]
[33] Saito M, Matsumoto M. Variants of Mersenne twister suitable for graphic processors. ACM Transactions on Mathematical Software. 2013;39:1-20. https://doi.org/10.1145/2427023.2427029
[34] The API reference guide for Thrust, the CUDA C++ template library. https://docs.nvidia.com/cuda/thrust/index.html; [accessed 14 September 2022]

[35] Foster I, Foster IT, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure: Morgan Kaufmann Publishers; 1999.
[36] EGI: Advanced computing for research https://www.egi.eu/; [accessed 17 September 2022]
[37] Portail des services RENATER https://services.renater.fr/; [accessed 17 September 2022]
[38] Haramoto H. MM, L'Ecuyer P. . A Fast Jump Ahead Algorithm for Linear Recurrences in a Polynomial Space. Springer, Berlin, Heidelberg; 2008. p. 9. https://doi.org/10.1007/978-3-540-85912-3_26
[39] Dirac Project http://diracgrid.org/; [accessed 17 September 2022]
[40] Tsai MY, Tian Z, Qin N, Yan C, Lai Y, Hung SH, et al. A new open-source GPU-based microscopic Monte Carlo simulation tool for the calculations of DNA damages caused by ionizing radiation --- Part I: Core algorithm and validation. Med Phys. 2020;47:1958-70. https://doi.org/10.1002/mp.14037
[41] Taioli S, Trevisanutto PE, de Vera P, Simonucci S, Abril I, Garcia-Molina R, et al. Relative role of physical mechanisms on complex biodamage induced by carbon irradiation. The Journal of Physical Chemistry Letters. 2020;12:487-93. https://doi.org/10.1021/acs.jpclett.0c03250
[42] de Vera P, Taioli S, Trevisanutto PE, Dapor M, Abril I, Simonucci S, et al. Energy Deposition around Swift Carbon-Ion Tracks in Liquid Water. Int J Mol Sci. 2022;23:6121. https://doi.org/10.3390/ijms23116121