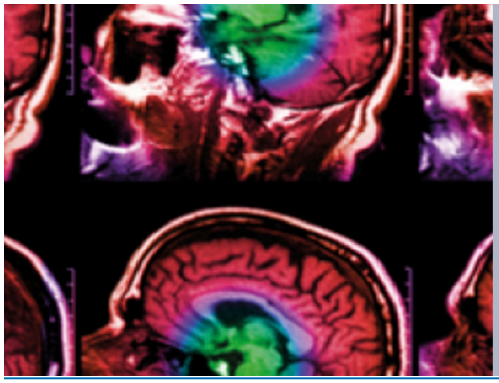
PAPER • OPEN ACCESS

Fast CPU-based Monte Carlo simulation for radiotherapy dose calculation



OPEN ACCESS
IOP Publishing | Institute of Physics and Engineering in Medicine Physics in Medicine & Biology
Phys. Med. Biol. **60** (2015) 6097–6111 doi:10.1088/0031-9155/60/15/6097

Fast CPU-based Monte Carlo simulation for radiotherapy dose calculation

Peter Ziegenhein, Sven Pirner, Cornelis Ph Kamerling and Uwe Oelfke

Joint Department of Physics at The Institute of Cancer Research and The Royal Marsden NHS Foundation Trust, London, SM2 5NG, UK

E-mail: Peter.Ziegenhein@icr.ac.uk and Uwe.Oelfke@icr.ac.uk

Received 26 January 2015, revised 21 June 2015
Accepted for publication 25 June 2015
Published 27 July 2015

Abstract

Monte-Carlo (MC) simulations are considered to be the most accurate method for calculating dose distributions in radiotherapy. Its clinical application, however, still is limited by the long runtimes conventional implementations of MC algorithms require to deliver sufficiently accurate results on high resolution imaging data. In order to overcome this obstacle we developed the software-package PhiMC, which is capable of computing precise dose distributions in a sub-minute time-frame by leveraging the potential of modern many- and multi-core CPU-based computers. PhiMC is based on the well verified dose planning method (DPM). We could demonstrate that PhiMC delivers dose distributions which are in excellent agreement to DPM. The multi-core implementation of PhiMC scales well between different computer architectures and achieves a speed-up of up to 37× compared to the original DPM code executed on a modern system. Furthermore, we could show that our CPU-based implementation on a modern workstation is between 1.25× and 1.95× faster than a well-known GPU implementation of the same simulation method on a NVIDIA Tesla C2050. Since CPUs work on several hundreds of GB RAM the typical GPU memory limitation does not apply for our implementation and high resolution clinical plans can be calculated.

Keywords: Monte Carlo simulation, photon dose calculation, CPU, multi-core

(Some figures may appear in colour only in the online journal)

1. Introduction

Monte Carlo simulation is considered to be the most accurate method to calculate dose distributions in radiotherapy (RT). It is based on explicitly simulating the trajectories and dose depositions of individual particles traversing a target by sampling interactions according to the fundamental laws of physics. This technique was applied in general purpose particle physics simulation software such as EGS4/5 (Nelson *et al* 1985, Bielajew *et al* 1994, Hirayama *et al* 2005), EGNSnrc (Kawrakow 2000), PENELOPE (Baro *et al* 1995, Salvat *et al* 1996, Sempau *et al* 1997, Salvat *et al* 2001) and Geant4 (Agostinelli *et al* 2003).

Because of the stochastic nature of this approach, a large number of particles has to be simulated in order to obtain sufficiently accurate results. Combined with the high complexity of individual interaction sampling this results in immense computational effort, which still limits a comprehensive clinical application of this dose calculation method in radiotherapy treatment planning. Therefore, several authors extensively investigated different elaborate methods to reduce the simulation times of MC algorithms, e.g. by variance reduction techniques (VRT) (Kawrakow and Fippel 2000, Buckley *et al* 2004, Wulff *et al* 2008) and/or by optimizing particle transport mechanics in various implementations specifically designed for clinical dose calculation purposes, like VMC++ (Kawrakow 2001), MCDOSE/MCSIM (Ma *et al* 2002), DPM (Sempau *et al* 2000).

In order to achieve a high performance the Monte Carlo simulation needs to be implemented in a parallel computing environment. Monte Carlo simulations are commonly classified as so called embarrassingly parallel problems. This means that only little or even no effort is required to separate the simulation process into parallel tasks which can be processed concurrently. The achievable speedup is then expected to scale linearly with the available number of processors. This has been shown for instance by Tyagi *et al* (2004) for a 32 processor computer cluster. An independent Monte Carlo simulation was launched on each computer producing an individual dose cube each which were merged after the simulation. The authors report an almost 32× speedup of the simulation as expected. A similar experiment was conducted by Pratx and Xing (2011) in a cloud computing environment. Given that the simulation time is high compared to the setup time, an almost linear performance scaling of the Monte Carlo dose calculation can be observed. For distributed-memory computing architectures the parallelization is straight forward.

Since its availability for general computing, exploiting graphics processing units (GPU) has been very popular in medical physics to speed up computationally intensive tasks. GPUs rely on a massively parallel architecture providing thousands of arithmetic units which are able to work concurrently on shared data. A number of MC dose calculation algorithms have been designed for GPUs such as gDPM 2.0 (Jia *et al* 2011), CPUMCD (Hissoiny *et al* 2011), GMC (Jahnke *et al* 2012), ArchertRT (Su *et al* 2014) and accelerated algorithms by Badal and Badano (2009). Unfortunately, within the scope of one processor (one GPU) when resources are shared the simulation problem is no longer embarrassingly parallel and additional effort has to be put into the parallel model. On the GPU for example Hissoiny *et al* (2011) and Jia *et al* (2011) report that it is difficult to achieve high speed-up factors due to its single-instruction multiple-thread (SIMT) architecture. SIMT dictates that all threads within a certain scheduling unit have to follow the same instruction path in order to be processed in parallel. If one particle is simulated by one thread this limitation conflicts with the stochastic nature of the Monte Carlo simulation. Another drawback of the GPU is that random accesses to the memory are time costly and the memory itself is limited to a few GB.

Despite these limitations authors report speed-up factors ranging from 27× (Badal and Badano 2009) to over 6000× (Jahnke *et al* 2012) of their GPU-based MC code compared to

algorithms tested on CPUs. These impressive speed-up factors suggest that GPUs in general are in the order of 100× or even 1000× faster than CPUs. However, these results have to be analyzed with care and in the right context. The reported speed-up factors have mostly been achieved by comparing a non-optimized, single-threaded implementation run on an out-dated CPU-system against a multi-threaded implementation on a modern professional GPU. A fair comparison between GPU and CPU results in a significant lower performance advantage (Jia 2015). Studies carried out for a wide range of computational algorithms show that there is actually only about 2.5× performance advantage in favor of the graphic cards on average (Lee *et al* 2010). GPUs might have a significant advantage for throughput problems for instance in the field of image processing. However for the use in radiotherapy we often find that the performance of GPUs is overrated.

In this paper we introduce PhiMC, a Monte Carlo simulation package which is optimized for modern multi-/many-core CPU-based shared memory systems. PhiMC relies on the physical model implemented in DPM (Sempau *et al* 2000) for non-parallel processors and adapts it for modern parallel architectures. Please note that our work is completely different from Tyagi *et al* (2004). While Tyagi *et al* (2004) uses the original sequential DPM code package and applies it to a distributed cluster environment on several computer nodes, PhiMC re-implements and optimizes the simulation physics for parallel execution within only one shared memory node. Thus, high performance is achieved on a single server or even desktop computer.

2. Materials and methods

2.1. DPM physics

The DPM MC-algorithm is a powerful sequential dose calculation MC-algorithm specifically designed and optimized for simulating the transport of electrons and photons through a heterogeneous medium for radiotherapy class problems. For both particle types it employs transport processes which permit long, heterogeneity boundary crossing simulation steps; photons are simulated by applying the Woodcock δ-scattering technique (Woodcock *et al* 1965) whereas electrons are transported using a mixed class simulation scheme. The latter method combines explicit sampling of hard inelastic interactions, such as Møller scattering and bremsstrahlung, with a condensed history approach. Interactions causing a severe energy loss of the primary electron are simulated using the random hinge technique as implemented by Sempau *et al* (2000). Small deflections are accumulated according to the multiple-scattering theory proposed by Goudsmit and Saunderson (1940). The energy losses due to latter interactions are accounted for by the continuous slowing down approximation. The transport of positrons is computed analogously to this of electrons with the difference that a positron annihilates with an electron of the medium and forms a back-to-back photon pair when it comes to rest. As shown by Chetty *et al* (2002) and (2003) the dose calculated by DPM agrees within (±2%/1–2 mm) of experimental measurements for both, therapeutic photon and electron beams.

DPM was optimized for a single thread execution on CPUs available at the late 90s. The high performance of this implementation was mainly achieved by employing a multiple scattering method for electrons which permits long transport steps across heterogeneity boundaries (Sempau *et al* 2000). Furthermore, a modified version of the RANECU random number generator implementation (James 1990) was used providing a period of $\approx 2 \times 10^{18}$ on 32-bit machines.

2.2. Multi- and many-core implementation

With PhiMC v1.0 we developed a software package specifically designed to deliver accurate high speed dose-calculation by utilizing the full parallel potential of modern multi core and many core CPUs. In order to achieve this our algorithm expands the physics and sequential methods used in the well verified DPM package by high performance implementations, which exploit multiple levels of parallelism modern CPUs offer: thread-level parallelism (TLP) and data-level parallelism (DLP). We take advantage of TLP by subdividing the total number of incident particles into bunches and distributing them among the available CPU cores. In principle each core is capable of simulating its own bunch of particles asynchronously and completely independent of the others; this is commonly referred to as embarrassingly parallel. However, due to the fact that during a simulation all cores need to access and alter data from the large common shared memory, which holds essential information about the voxel-grid, the challenge of orchestrating shared memory accesses arises. For this reason, we developed a sophisticated memory scheme (section 2.2.1), which on the one hand schedules and pipelines reading from and writing to the shared memory and optimizes the access times of frequently used data stored in CPU caches on the other hand. In order to leverage the DLP, which technically is realized by a single instruction multiple data (SIMD) architecture inside the processing unit of a CPU core, we vectorized the instructions in the actual simulation process (section 2.2.2) where ever possible. In contrast to most GPU-based MC-implementations, PhiMC consequently operates on double precision data for the simulation itself as well as for storing the dose distribution.

2.2.1. MC memory-scheme. The right hand side of figure 1 illustrates the way PhiMC allocates and accesses memory. Basically, the data required for simulating particle histories is divided into two classes: on the one hand a large set of voxel-grid data containing information about the density, material and deposited dose of each voxel. On the other hand much smaller data packages encompassing pre-calculated look-up tables for different particle and material types, current particle data as well as temporary simulation data and buffered dose values.The latter (smaller) type of data is kept individually for every CPU core. This includes the data used for generating random numbers. Each core uses its own instance of the random number generator (RNG) that is initialized with an individual seed. The period length of the RNG used for PhiMC (see section 2.3.1) is multiple orders of magnitude larger than the amount of random numbers generated for a simulation. This prevents correlations between the concurrently running Monte Carlo threads on each core.

Because of its enormous size, the voxel-grid data is stored in a shared address space, located on the random access memory (RAM) of the system, which can hold up to several hundreds of giga-bytes of data. In PhiMC we put special emphasis on orchestrating and optimizing the data transfer between shared memory and the individual CPU cores. The fact that writing to the shared memory is only possible sequentially is the sole limiting factor of TLP. This is particular important for the process of adding dose to the voxel-grid stored in shared memory. The multi-core implemenation of PhiMC has to guarantee that only one thread at a time can write to the same grid location. Therefore, we implemented an intelligent memory manager which buffers calculated dose values and pipelines back-writing to shared memory as efficient as possible by preventing race conditions and idling times: When the buffer reaches a certain filling level the corresponding thread liaises with other threads in the system to negotiate a time slot for the write-back operation. During this time slot access to the shared dose cube is denied for all other threads.

PhiMC is designed in a way, that small packages of data, which are frequently read, created or altered, are stored separately in the cache of each processor core. Even if this leads to

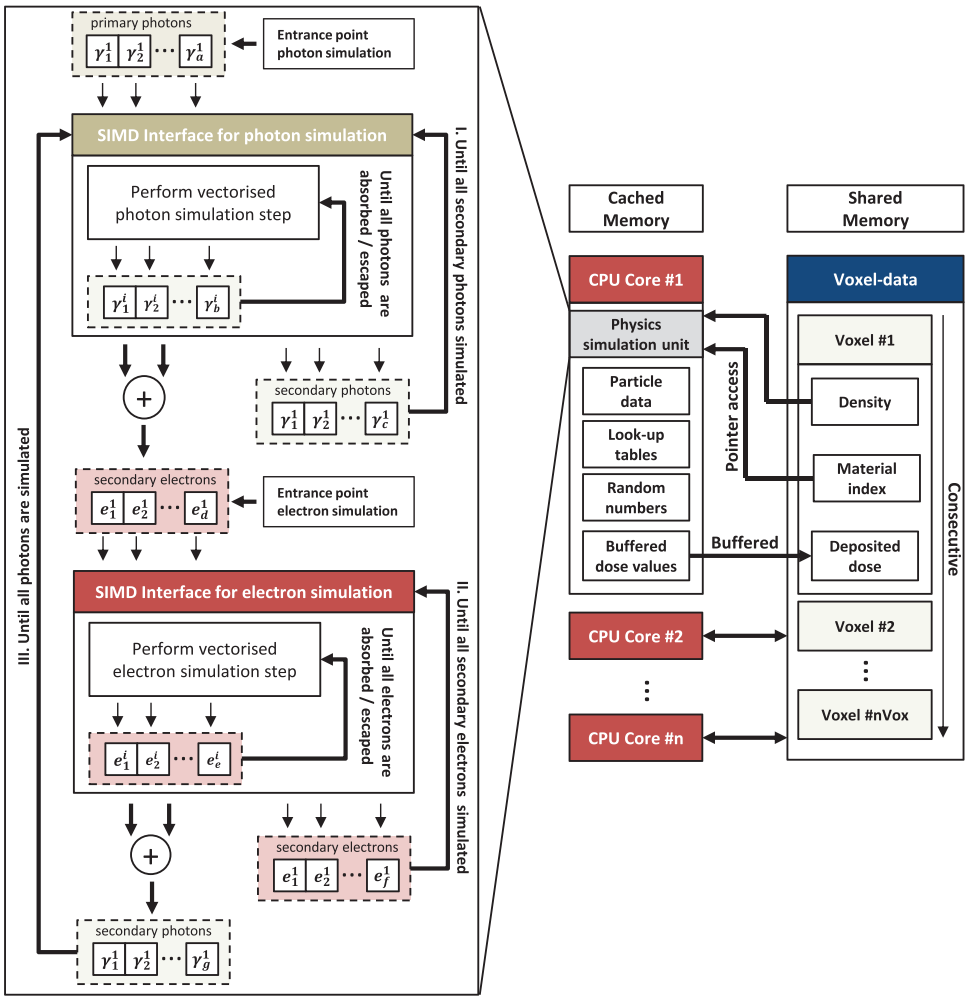


Figure 1. Simulation process (left) and memory model (right) as implemented in PhiMC. The simulation unit simulates the transport of photons and electrons separately whereas the data scheme ensures that each core holds its own set of frequently used data as closely as possible to the processing unit and orchestrates their access to their shared memory.

slight data redundancies, it ensures minimal access times as well as that all cores can simulate particle histories completely asynchronously.

2.2.2. The simulation process. After PhiMC has set up the memory structure as explained above and a simulation unit on every available core, it generates incident particles according to the applied source model (section 2.3.3). The particles are combined in batches fitting in the cache of a CPU core and equally distributed among the available simulation units. On the left side of figure 1 the simulation process of PhiMC is illustrated. In PhiMC we employ a ping–pong approach for the simulation of particles. For the simulation of photons, first, the vector of incident photons is transferred to the photon simulation interface, which propagates them through the medium by executing vectorized simulation steps until all photons either are

absorbed or have left the phantom. Secondary photons, which are created during this simulation loop are stored temporarily and copied back to the initial input vector after the simulation of the primary particles is completed (I). This procedure is repeated until no new photons are created during a simulation loop. Second, all electrons generated during this process are added to the main electron vector and computed analogously directly thereafter by the electron simulation interface (II). In the third step all secondary photons created during the simulation of the electron vector are passed to the photon simulation interface again (III). This process continues until all particles have been absorbed or have left the target.

The ping–pong approach for the simulation of particles as shown in figure 1 is similar to the approach introduced for the GPU-based implementations in Jia *et al* (2011). The goal is the same for both platforms: to process only one type of particle at the same time. However the effect is different. On the GPU this leads to a reduction in thread divergence which improves parallelism. On the CPU the ping–pong approach leads to an efficient use of cache memory which improves the bandwidth of the system.

2.3. The PhiMC framework

In order to ensure that the simulation algorithm scales well on different CPU-types and future processor architectures, PhiMC is implemented with no dependency on the employed operating system and hardware. This is achieved by including pervasive state-of-the-art libraries. Due to its modular multi-layer structure, all proprietary parts of PhiMCs' simulation unit can be substituted by other libraries and application programming interfaces (API) without much effort. Some of the key modules are described briefly in the following.

2.3.1. Random number generation. The backbone of every MC simulation is the optimal choice of the underlying Random Number Generator. In the case of PhiMC the RNG not only has to pass statistical test for randomness (Kendall and Smith 1938) but also be exceptionally fast. Recently released hardware based true random number generators (TRNG)[1] are too slow to fulfill our requirements. Instead, we included the high-performance *Intel MKL*[2] implementation of the widely used mersenne twister pseudo random number generator (PRNG) which was proposed by Matsumoto and Nishimura (1998). This RNG has a period length equal to $2^{19937} - 1$ which is significantly better compared to the RNG used in the original DPM ($\approx 2 \times 10^{18}$) and gDPM (2^{192}) on GPUs. In order to fully exploit the potential of the hardware accelerated parallel capabilities of this implementation and to minimize the functions overhead, we generate arrays of random numbers $x \in (0, 1)$ instead of only one random number at a time. When the simulation module requests a random number, a value is taken from that pre-generated array rather then being calculated on the fly. When the last random number is used up, the array is re-populated using the hardware accelerated parallel implementation.

2.3.2. TPL and DLP realization. PhiMC uses open multi-processing (OpenMP), an API that provides shared memory multi-processing functionality, to distribute the individual history simulations among the available CPU-cores and control their shared memory access. OpenMP is supported by almost all compiler and operating systems. In order to leverage the SIMD potential of modern CPUs, PhiMC performs vectorized operations where ever possible. On

[1] https://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide.

[2] https://software.intel.com/en-us/intel-mkl.

that account our algorithm uses the C/C++ extension kit Intel Cilk Plus. Although it also supports multi-core parallelism, in PhiMC Intel Cilk is only used as a DLP catalyst.

2.3.3. Source and patient modeling. A simple source model was implemented in PhiMC to test the simulation algorithm. A finite radiation source was modeled according to the method provided by Altenstein *et al* (2012) which is based on Fippel *et al* (2003). The interpretation of the CT-data was taken from the BEAMnrc implementation (Rogers *et al* 2001): The hounsfield units (HU)-range is divided into four major types of tissue (air, lung tissue, soft tissue and bone) while the density in each voxel is linearly interpolated within the tissue classes. In order to calculate intensity modulated radiotherapy (IMRT) plans, particles are generated according to beam fluence maps. We implemented the same Metropolis sampling algorithm using Markov chains (Hastings 1970) which was also used in Jia *et al* (2011) for gDPM 2.0. This sampling method is not only well suited for GPUs but also efficient on modern CPUs. All of these models have been implemented to be easily exchangeable against more accurate implementations in the future. The discussion of more sophisticated source and patient models is beyond the scope of this paper.

2.4. Test cases and performance metric

The dose value D of each voxel computed in a full MC simulation with a sufficiently large number of simulated particles is a statistical quantity, which, according to the central limit theorem (CLT), is distributed normally with the standard deviation σ. To ensure that the requirements of the CLT are met, in the following, we only analyze voxels whose dose value exceeds 50% of the maximum dose value of the whole phantom. The uncertainty of the dose distribution is expressed using the mean relative error $\langle \sigma / D_{\max} \rangle$ defined as:

$$\langle \sigma / D_{\max} \rangle = \sqrt{\frac{\frac{1}{N_0} \sum_i \left[\langle d_i^2 \rangle - \langle d_i \rangle^2 \right]}{(D_{\max}/N_0)^2 N_{50}}} \tag{1}$$

with N_0 being the total number of histories simulated, N_{50} the number of dose entries which exceed 50% of the maximum dose $D_{\max}$ and $\langle d_i \rangle$ is the dose value in voxel i. Please note that the relative uncertainties are defined using the maximum dose of the plan instead of the actual dose of the voxel in question. This definition underestimates the mean uncertainties of the dose distribution but it allows for a direct performance comparison to other publications (e.g. Jia *et al* (2011)).

In order to prove that the results calculated by PhiMC are equivalent to those computed by the reference algorithm DPM, we apply the two one-sided test (TOST) procedure as proposed by Schuirmann (1987). A standard two tailed t-test is incapable of ascertaining statistical similarity, since not neglecting the null hypothesis of indifference does in general not imply statistically secured similarity. The TOST solves this issue by replacing the null hypothesis H_0 of indifference with an interval hypothesis of non-equivalence within the boundaries θ_u, θ_l:

$$H_0: \quad D_t - D_r \leqslant \theta_l \quad \text{or} \quad D_t - D_r \geqslant \theta_u \tag{2}$$

$$H_1: \quad \theta_l < D_t - D_r < \theta_u \,, \tag{3}$$

where D_t and D_r denote the dose of the considered voxel as calculated by PhiMC and DPM, respectively. Therefore, neglecting the interval hypotheses is a direct proof that the results of both algorithms lie within the equivalence interval $[\theta_l, \theta_u]$.

In this test, we ran $n = m = 40$ simulations with both algorithms, our PhiMC and the original DPM implementation in Fortran 'as–is'. In each run we simulated a sufficient amount of incident particles, so that the mean relative error $\langle \sigma / D_{\max} \rangle$ of all voxels with a dose higher than 20% of the max dose $D_{\max}$ lies well below 1% in each simulation for all tested phantoms. We then calculated the mean dose values $\langle D_t \rangle$ and $\langle D_r \rangle$ of each voxel as well as $s_{t,r} = \sum (D^i_{t,r} - \langle D_{t,r} \rangle)^2$ in order to calculate the corresponding $t_{l,u}$-values:

$$t_{l,u} = \sqrt{\frac{nm}{n+m}} \sqrt{\frac{n+m-2}{s_t + s_r}} \left(\langle D_t \rangle - \langle D_r \rangle - \theta_{l,u} \right) \tag{4}$$

As stated by Schuirmann (1987) the doses of a single voxel as calculated by both algorithms are equivalent with a confidence of 90% if both one-handed parts of the interval hypotheses can be neglected with a significance level of $\alpha = 0.05$, i.e. if $t_l > t_{1-\alpha,n+m-2}$ and $t_u < -t_{1-\alpha,n+m-2}$. In order to accumulate the results of this quantitative analyses for all considered voxels, the passing rates P, i.e. the number of voxels for which the hypotheses were neglected divided by the total number of considered voxels, is calculated for all discussed phantoms and source types.

Runtime and accuracy of PhiMC was tested against a selection of phantoms and two clinical cases. The phantoms have been chosen to match commonly investigated geometries in other high ranked Monte Carlo studies. All phantoms consist of $61 \times 61 \times 150$ voxel with a voxel size of $0.5 \times 0.5 \times 0.2$ cm^3. We used a water phantom, a bone phantom and a lung phantom. The bone- and lung phantom consist of a water box with a layer of the respective material ranging from $z = 5$ cm to $z = 10$ cm. The material is isotropic in x- and y-direction for all phantoms. The resolution and size of the clinical cases is given in table 2. Both patients have been planned based on a pencil-beam pre-calculated dose data set.

In order to investigate the performance and scalability of our PhiMC implementation the performance has been tested on several commonly available CPU-based systems: Two single node workstation systems XeonV1[3] and XeonV3[4] and one low-cost desktop system which will be denoted as i7[5].

3. Results

3.1. PMC accuracy

Table 1 shows the results of the TOST comparison between our newly developed PhiMC and the original reference algorithm DPM. Tests have been carried out for electrons and photons. The number of simulated primary particles is chosen high to eliminate statistical variation as far as possible. The TOST passing rate for using electrons is found to be 100% for all three phantom studies while the mean relative statistical uncertainty is below 0.5%. Using photons the ratio of equivalent voxel pairs passing the test exceeds 99.5% for all phantom geometries. According to the TOST procedure PhiMC is in excellent agreement with DPM. Comparative iso-dose plots and integrated depth dose curves shown in figure 3 support this finding.

3.2. Simulation speed and runtime scaling

PhiMC was explicitly designed to deliver high performance on modern CPU-based systems. Thus extensive runtime and scaling studies have been conducted to demonstrate the performance of our algorithm.

[3] 2 × Intel Xeon E5-2650 8 Core—2.00 GHz.
[4] 2 × Intel Xeon E5-2699 v3 18 Core—2.3 GHz (C1 Pre-Production Processors).
[5] Intel i7-4770 4 Core—3.4 GHz.

Table 1. Comparison of PhiMC and DPM.

Source	No. of histories	Phantom	$\langle\sigma/D_{max}\rangle$ DPM [%]	$\langle\sigma/D_{max}\rangle$ PhiMC [%]	P [%]	T PhiMC [s]
20 MeV e	$9 \cdot 10^6$	H_2O	0.49	0.48	100.0	7.2
MeV e	$9 \cdot 10^6$	H_2O-Bone-H_2O	0.32	0.31	100.0	8.4
MeV e	$9 \cdot 10^6$	H_2O-Lung-H_2O	0.50	0.49	100.0	8.4
MV γ	$9 \cdot 10^8$	H_2O	0.47	0.47	99.7	47.3
MV γ	$9 \cdot 10^8$	H_2O-Bone-H_2O	0.41	0.41	99.7	58.8
MV γ	$9 \cdot 10^8$	H_2O-Lung-H_2O	0.46	0.45	99.6	44.7

Note: The dose of three phantom cases has been compared for electrons and photons using the two one-sided test (TOST). Relative mean uncertainty and runtimes on the XeonV3 system are also shown.

Figure 2 summarizes the runtimes of calculating dose in water on different CPU-based systems depending on the number of photon histories. The graph reveals two important findings: First, the simulation time scales linearly with the number of particles on all systems. Although it is not shown explicitly, the same holds true for the simulation of electron histories. Second, the runtime scales very well with the number of available cores of a CPU system and with advancing processor technology. For example simulating 100 million photon histories on the XeonV1 system takes about 174.7 s using a single CPU core. Using all available cores results in a runtime of 12.9 s which is about 13.4× faster. The PhiMC implementation also performs well on common desktop CPUs: The low-cost desktop i7 system is approximately 5.8× faster than a single core simulation on a professional XeonV1 server. The fastest execution times have been measured on the XeonV3 system. It provides runtimes which are up to 33.3× faster compared to the XeonV1 system. Simulating 250 million photon histories in the water phantom can be completed within 13.1 s. The respective simulation runtimes for 250 million histories using the bone and lung phantoms can be extrapolated from the runtimes shown in table 1 due to the linear scaling behavior of the simulation. Figure 2 also shows the simulation runtime of the original DPM code executed on the XeonV1 system. Naturally DPM is only tested on a single core since it is not capable of exploiting a multi-core CPU architecture.

Table 2 shows the performance of PhiMC on an IMRT prostate case and on an IMRT head and neck case. The runtimes include the generation of the 250 million particle histories according to the method described in section 2.3.3. The voxel resolution is of clinical quality while the number of histories has been chosen to keep the average relative uncertainty well below 1%. Figure 4 shows the resulting dose distribution of both clinical cases. Dose is reported as dose to water.

4. Discussion and outlook

In this work we demonstrate that high-quality Monte Carlo simulations for advanced, clinical RT can be performed on CPU-based systems in less than a minute. Our newly developed framework PhiMC implements the physical models originally used in DPM which have been proven to be accurate (Chetty *et al* 2002, 2003). Using the statistical Two One-Sided Test procedure we validated that dose distributions calculated by PhiMC and DPM are in excellent agreement.

The performance of PhiMC was investigated on several phantom cases and two typical clinical patient cases. The number of histories was chosen to be comparable to other high-ranked

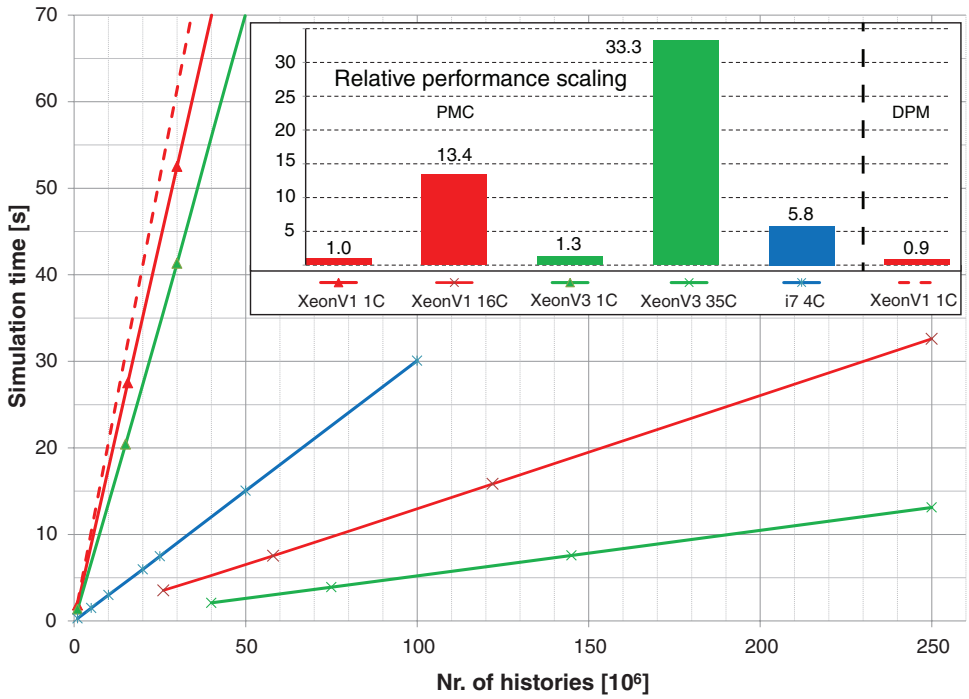


Figure 2. Absolute simulation times with respect to the number of incident photons in water on different machines. The number of cores used is stated behind the machine specifier (e.g. 16C means that 16 cores have been used for the calculation). The relative speed-up of the simulation is summarized in the bar chart at the top right corner of the graph.

Table 2. PhiMC runtimes and uncertainties acquired on realistic clinical cases using the XeonV3 system.

Case	No. of histories	Size [voxel^3]	Resolution [mm^3]	$\langle \sigma / D_{max} \rangle$ [%]	T_{PMC} [s]
IMRT prostate	$2.5 \cdot 10^8$	$256 \times 256 \times 234$	$1.95 \times 1.95 \times 2.0$	0.68	26.7
	$1.0 \cdot 10^8$	$256 \times 256 \times 234$	$1.95 \times 1.95 \times 2.0$	1.06	10.8
IMRT HN	$2.5 \cdot 10^8$	$256 \times 256 \times 116$	$1.56 \times 1.56 \times 3.0$	0.55	18.8
	$1.0 \cdot 10^8$	$256 \times 256 \times 116$	$1.56 \times 1.56 \times 3.0$	0.89	8.2

publications. The dose of the optimized prostate case could be simulated in 26.7 s while the simulation of the H&N case took only 18.8 s due to the smaller anatomical volume. Compared to the GPU-based implementation of gDPM 2.0 presented by Jia *et al* (2011) we could achieve a speed-up of up to 1.95× for a comparable IMRT H&N case. This speed-up was measured on a dose grid which comprises four times more voxels compared to the patient case tested on the GPU. On CPUs a larger amount of memory is available. PhiMC exploits this fact and is able to calculate dose on a full clinical resolution voxel grid (see table 2) while the resolution of the GPU test cases in Jia *et al* (2011) is reduced by a factor of 2 in x- and y-direction, respectively. Assuming a linear runtime scaling with the number of particle histories simulated, the

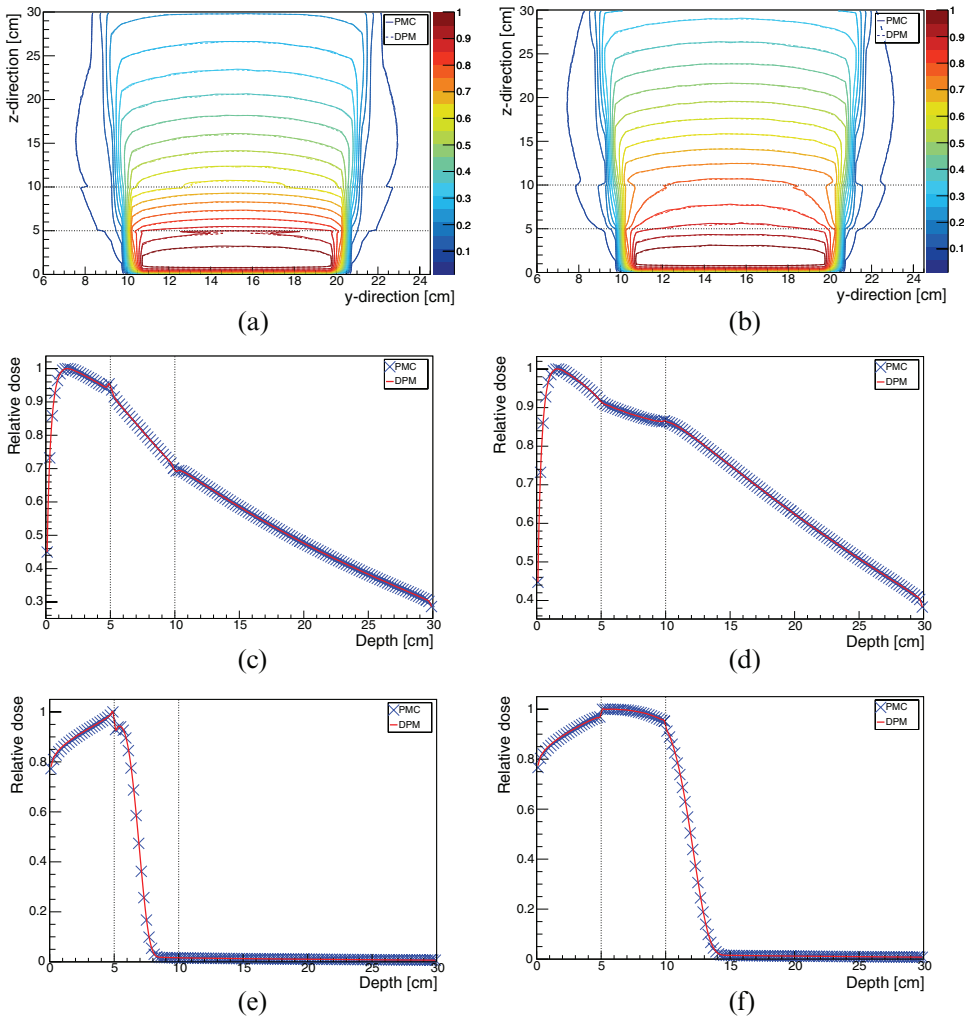


Figure 3. Comparison of the simulation results of PhiMC and DPM using the bone phantom (a), (c), (e) and the lung phantom (b), (d), (f), 900 million photons have been calculated for figures (a)–(d) while 9 million electrons have been calculated for the comparison in (e), (f). An open field of 10×10 cm was used in all cases. (a) 6 MV photons bone phantom. (b) 6 MV photons lung phantom. (c) Depth dose 6 MV photons bone phantom. (d) Depth dose 6 MV photons lung phantom. (e) Depth dose 20 MeV electrons bone phantom. (f) Depth dose 20 MeV electrons lung phantom.

speed-up of PhiMC is 1.26× and 1.31× for the lung phantom and bone phantom, respectively. The average relative uncertainties produced by gDPM and PhiMC are comparable.

The performance evaluation across different platforms (figure 2) shows that PhiMC is able to achieve competitive runtimes on both server CPUs and low-cost desktop CPUs. The performance scales well with the number of cores. This was expected since on CPU core-level Monte Carlo simulations can be formulated as embarrassingly parallel problems. Please note that on modern server CPUs one cannot expect an ideal linear scaling with the number of cores, since the clock frequency of the processor is reduced due to a power management scheme when multiple cores are in use. Thus for example a performance scaling of 13.4× on

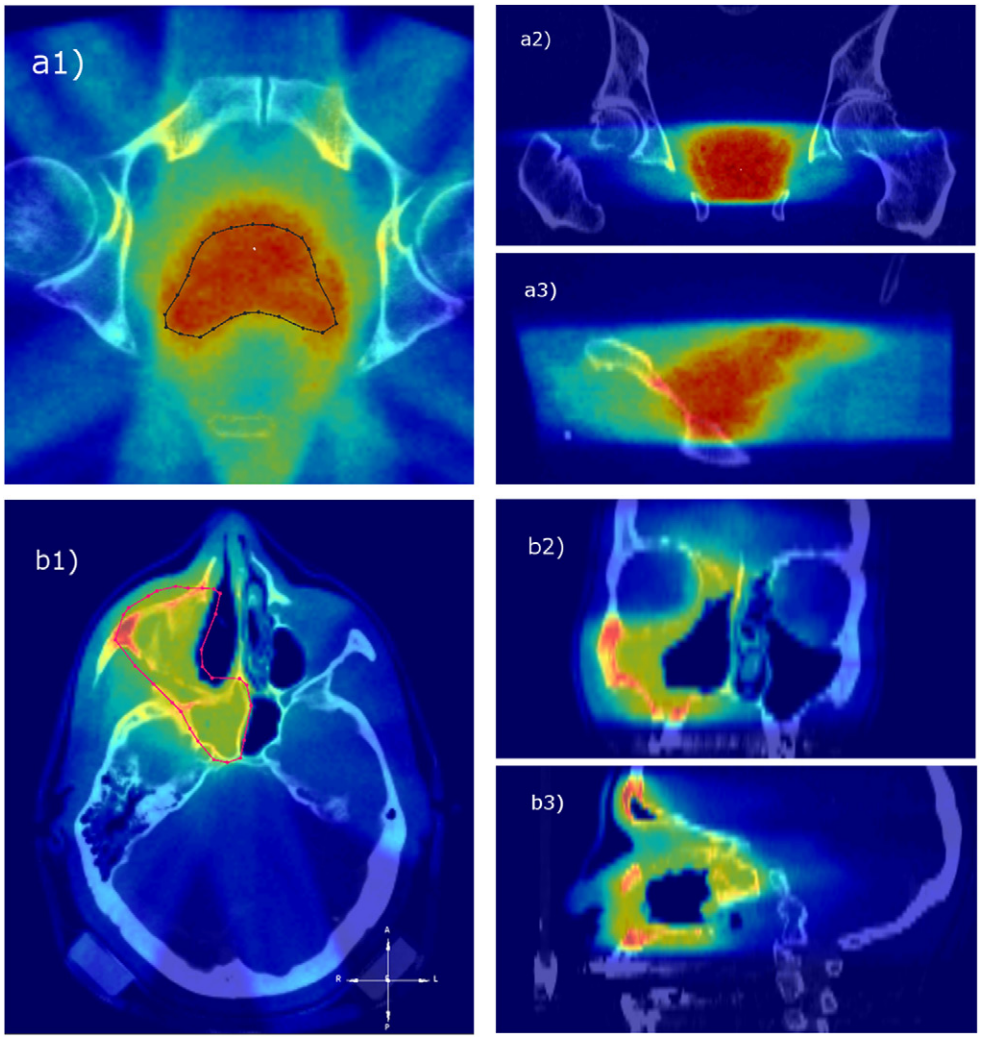


Figure 4. MC simulation results for a nine beam IMRT prostate plan (a1)–(a3) and a nine beam head&neck plan (b1)–(b3) using the PhiMC package. The PTV contours are outlined on the transversal views. The colormap consists of 64 colors which are equally distributed between 25% and 100% of the maximal dose.

16 cores compared to single core execution (see figure 2) is an excellent result. PhiMC runs efficiently on a variety of CPU architectures. The processor used in the test platform XeonV1 was introduced at the beginning of 2012 while the XeonV3 version will be released in the near future. Compatibility over this almost three year span in technology is handled through the use of OpenMP and Intel Cilk which abstract the actual hardware. With the parallel model introduced in this paper we are confident that PhiMC can also exploit future CPU technologies efficiently.

Figure 2 also shows the performance of the original DPM implementation as it is openly provided by the author (Sempau *et al* 2000)[6]. PhiMC shows a slightly better single-core

[6] http://inte.upc.edu/downloads.

performance which is probably due to the use of Intels MKL random number generator and explicitly exploiting wide SIMD register via Intel Cilk. A larger performance gap was not expected since DPM was compiled with the lates Intel Fortran compiler which automatically generates optimized serial code for the XeonV1 server system. However the compiler cannot automatically generate parallel code which exploits multiple CPU cores. That is why the development of PhiMC was necessary in the first place.

The performance scaling of PhiMC on multiple CPU cores is limited by the dose data transportation process. While the particle simulation itself runs independently on each arithmetic unit in parallel, the deposited energy values have to be scored into a shared dose cube in main memory sequentially. The available memory bandwidth is efficiently used in PhiMC through the buffered write-back memory scheme we introduced in section 2.2.1. However with a rising number of CPU cores involved in the simulation the pressure on the memory bandwidth increases. Although we do not observe a problem on current technologies (XeonV3) a potential bottleneck could form for future technologies since commonly arithmetic performance increases faster than memory bandwidth. To prevent this issue we are currently investigating the use of a buffered hash table technique to reduce the number of energy deposition write back operations.

While the embarrassingly parallel character of the MC simulation leads to an excellent scaling on the core level, the exploitation of the instruction level parallelism on SIMD registers is quite poor. Here the CPU has the same problems as the GPU which uses the similar SIMT technique. Due to the fact that MC simulations are of statistical nature and particles never follow the same execution path, the SIMD respectively SIMT concept cannot be used efficiently with the DPM physics model. This issue has also been reported for other GPU-based Monte Carlo implementation (Hissoiny *et al* 2011, Jia *et al* 2011). Since the width of SIMT on GPUs is larger than for SIMD on CPUs, the divergence problem is expected to be more severe on GPUs. Unfortunately, the execution path divergence in Monte Carlo will become more significant. Due to technical issues and power limitations it is expected that performance on future (GPU and CPU) processors will increase by employing an even higher degree of data level parallelism. Therefore we think it is worthwhile to invest future research effort concerning Monte Carlo into the field of variance reduction techniques in view of modern processor hardware.

5. Conclusion

In this work we demonstrate that an optimized CPU-based MC-algorithm achieves a higher performance compared to a well-known GPU-based implementations. Simulating the dose distribution on clinical cases in full resolution can be done in less than a minute with an accuracy of under 1%. Thus it is not mandatory to invest in a specialized Tesla GPU device to enable fast, high quality MC dose calculations. Conventional single node workstations can be utilized as well as they are already present in most clinics and research facilities. Furthermore, it is not necessary to write program code in a hardware specific language (like CUDA). PhiMC is written in C++ which can be maintained by most scientist and software developer. Another advantage is that modern CPUs can employ up to several hundreds of GB of RAM while GPUs are limited to only a few GB. This allows the CPU-based implementation to simulate dose for very large clinical therapy plans and for smaller voxel sizes.

Acknowledgments

This research at The Institute of Cancer Research was supported by Cancer Research UK under Programme C33589/A19727 and NHS funding to the NIHR Biomedical Research Centre at The Royal Marsden and The Institute of Cancer Research.

We would like to thank the Intel Corporation (UK) Ltd. for providing access to the E5-2699v3 pre-production system from their Swindon HPC lab.

References

Agostinelli S *et al* 2003 Geant4—a simulation toolkit *Nucl. Instrum. Methods Phys. Res.* **506** 250–303
Altenstein G, Nill S, Heller J, Heid O and Oelfke U 2012 A novel 2d binary collimator for IMRT dose delivery: dosimetric characterization using Monte Carlo simulations *Phys. Med. Biol.* **57** N345
Badal A and Badano A 2009 Accelerating Monte Carlo simulations of photon transport in a voxelized geometry using a massively parallel graphics processing unit *Med. Phys.* **36** 4878–80
Baro J, Sempau J, Fernández-Varea J and Salvat F 1995 Penelope: an algorithm for Monte Carlo simulation of the penetration and energy loss of electrons and positrons in matter *Nucl. Instrum. Methods Phys. Res.* **100** 31–46
Bielajew A F, Hirayama H, Nelson W R and Rogers D W O 1994 History, overview and recent improvements of EGS4 *NRCC Report* PIRS-0436 National Research Council, Ottawa, Canada
Buckley L A, Kawrakow I and Rogers D 2004 CSnrc: correlated sampling Monte Carlo calculations using EGSnrc *Med. Phys.* **31** 3425–35
Chetty I J, Charland P M, Tyagi N, McShan D L, Fraass B A and Bielajew A F 2003 Photon beam relative dose validation of the DPM Monte Carlo code in lung-equivalent media *Med. Phys.* **30** 563–73
Chetty I J, Moran J M, McShan D L, Fraass B A, Wilderman S J and Bielajew A F 2002 Benchmarking of the dose planning method (DPM) Monte Carlo code using electron beams from a racetrack microtron *Med. Phys.* **29** 1035–41
Fippel M, Haryanto F, Dohm O, Nüsslin F and Kriesen S 2003 A virtual photon energy fluence model for Monte Carlo dose calculation *Med. Phys.* **30** 301–11
Goudsmit S and Saunderson J 1940 Multiple scattering of electrons *Phys. Rev.* **57** 24
Hastings W K 1970 Monte Carlo sampling methods using Markov chains and their applications *Biometrika* **57** 97–109
Hirayama H, Namito Y, Bielajew A F, Wilderman S J and Nelson W R 2005 The EGS5 code system *Stanford Linear Accelerator Center Report* SLAC-R-730
Hissoiny S, Ozell B, Bouchard H and Després P 2011 GPUMCD: a new GPU-oriented Monte Carlo dose calculation platform *Med. Phys.* **38** 754–64
Jahnke L, Fleckenstein J, Wenz F and Hesser J 2012 GMC: a GPU implementation of a Monte Carlo dose calculation based on Geant4 *Phys. Med. Biol.* **57** 1217
James F 1990 A review of pseudorandom number generators *Comput. Phys. Commun.* **60** 329–44
Jia X, Gu X, Graves Y J, Folkerts M and Jiang S B 2011 GPU-based fast Monte Carlo simulation for radiotherapy dose calculation *Phys. Med. Biol.* **56** 7017
Jia X *et al* 2015 GPU technology is the hope for near real-time Monte Carlo dose calculations *Med. Phys.* **42** 1474–6
Kawrakow I and Fippel M 2000 Investigation of variance reduction techniques for Monte Carlo photon dose calculation using XVMC *Phys. Med. Biol.* **45** 2163
Kawrakow I 2000 Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version *Med. Phys.* **27** 485–98
Kawrakow I 2001 *Advanced Monte Carlo for Radiation Physics, Particle Transport Simulation and Applications* (Berlin: Springer) pp 229–36
Kendall M G and Smith B B 1938 Randomness and random sampling numbers *J. R. Stat. Soc.* **101** 147–66
Lee V W *et al* 2010 Debunking the 100× GPU versus CPU myth: an evaluation of throughput computing on CPU and GPU *ACM SIGARCH Computer Architecture News* **38** 451–60
Matsumoto M and Nishimura T 1998 Dynamic creation of pseudorandom number generators *Monte Carlo and Quasi-Monte Carlo Methods* **2000** 56–69
Ma C, Li J, Pawlicki T, Jiang S, Deng J, Lee M, Koumrian T, Luxton M and Brain S 2002 A Monte Carlo dose calculation tool for radiotherapy treatment planning *Phys. Med. Biol.* **47** 1671
Nelson W R, Hirayama H and Rogers D W O 1985 The EGS4 code system *Standard Linear Accelerator Report* 265
Pratx G and Xing L 2011 Monte Carlo simulation of photon migration in a cloud computing environment with mapreduce *J. Biomed. Opt.* **16** 125003–39
Rogers D W O, Ma C M, Ding G X, Walters B, Sheikh-Bagheri D and Zhang G G 2001 BEAMnrc users manual *NRC Report* PIRS 509 (a) revF

Salvat F, Fernandez-Varea J, Baro J and Sempau J 1996 Penelope, an algorithm and computer code for Monte Carlo simulation of electron–photon showers *Technical Report* Centro de Investigaciones Energeticas Medioambientales y Tecnologicas (CIEMAT), Madrid (Spain)

Salvat F, Fernández-Varea J M, Acosta E and Sempau J 2001 PENELOPE, a code system for Monte Carlo simulation of electron and photon transport *Proc. Workshop/Training Course, OECD/NEA (5–7 November 2001) NEA/NSC/DOC (2001)* 19

Schuirmann D J 1987 A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability *J. Pharmacokinet. Biopharm.* **15** 657–80

Sempau J, Acosta E, Baro J, Fernández-Varea J and Salvat F 1997 An algorithm for Monte Carlo simulation of coupled electron–photon transport *Nucl. Instrum. Methods Phys. Res.* **132** 377–90

Sempau J, Wilderman S J and Bielajew A F 2000 DPM, a fast, accurate monte carlo code optimized for photon and electron radiotherapy treatment planning dose calculations *Phys. Med. Biol.* **45** 2263

Su L, Yang Y, Bednarz B, Sterpin E, Du X, Liu T, Ji W and Xu X G 2014 $\text{ARCHER}_{\text{RT}}$—a GPU-based and photon–electron coupled Monte Carlo dose computing engine for radiation therapy: software development and application to helical tomotherapy *Med. Phys.* **41** 071709

Tyagi N, Bose A and Chetty I J 2004 Implementation of the DPM Monte Carlo code on a parallel architecture for treatment planning applications *Med. Phys.* **31** 2721–5

Woodcock E, Murphy T, Hemmings P and Longworth S 1965 Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry *Applications of Computing Methods to Reactor Problems: Argonne National Laboratories Report* ANL-7050 vol 557

Wulff J, Zink K and Kawrakow I 2008 Efficiency improvements for ion chamber calculations in high energy photon beams *Med. Phys.* **35** 1328–36