SLAC-PUB-13801

Radiation Transport Calculations and Simulations

A. Fassò [a] and A. Ferrari [b] *
[a]SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA 94025, USA
[b]CERN, 1211 Geneva 23, Switzerland

Received on April 2, 2003, revised on November 19, 2003, accepted on December 22, 2003

This article is an introduction to the Monte Carlo method as used in particle transport. After a description at an elementary level of the mathematical basis of the method, the Boltzmann equation and its physical meaning are presented, followed by Monte Carlo integration and random sampling, and by a general description of the main aspects and components of a typical Monte Carlo particle transport code. In particular, the most common biasing techniques are described, as well as the concepts of estimator and detector. After a discussion of the different types of errors, the issue of Quality Assurance is briefly considered.

INTRODUCTION

The Monte Carlo method was invented by John von Neumann, Stanislaw Ulam and Nicholas Metropolis (who gave it its name), and independently by Enrico Fermi. Originally it was not a simulation method, but a device to solve a multidimensional integro-differential equation by building a stochastic process such that some parameters of the resulting distributions would satisfy that equation. The equation itself did not necessarily refer to a physical process, and if it did, that process was not necessarily stochastic [1].

It was soon realized, however, that when the method was applied to an equation describing a physical stochastic process, such as neutron diffusion, the model (in this case a random walk) could be identified with the process itself. In these cases the method (analog Monte Carlo) has become known as a simulation technique, since every step of the model corresponds to an identical step in the simulated process.

Particle transport is a typical physical process described by probabilities (cross sections are interaction probabilities per unit distance). Therefore it lends itself naturally to be simulated by Monte Carlo. Many applications, especially in high energy physics and medicine, are based on simulations where the history of each particle (trajectory, interactions) is reproduced in detail. However in other types of application, typically shielding design, the user is interested only in the expectation values of some quantities (fluence and dose) at some space point or region, which are calculated as solutions of a mathematical equation.

This equation (the Boltzmann equation), describes the statistical distribution of particles in phase space and therefore does indeed represent a physical stochastic process. But in order to estimate the desired expectation values it is not necessary that the Monte Carlo process be identical to it.

In many cases, it is more efficient to replace the actual process by a different one resulting in the same average values but built by sampling from modified distributions Such a biased process, if based on mathematically correct variance reduction techniques, converges to the same expectation values as the unbiased one, but it cannot provide information about the higher moments of statistical distributions (fluctuations and correlations). In addition, the faster convergence in some user-privileged regions of phase space is compensated by a slower convergence elsewhere.

Complexity

In the past, when computers were still slow and expensive, shielding design was based on analytical and numerical techniques (point-kernel formulae based on source terms, build-up factors and attenuation lengths). Computers were used to run simple codes based on those point-kernel formulae or early Monte Carlo codes featuring simple geometries (R-Z) and many physics approximations (e.g. no thermal neutrons, no multiple scattering).

In recent years, thanks to the availability of faster and cheaper computers, and of computer codes of improved quality, an increasing number of particle transport calculations in scientific and applied fields are carried out by means of Monte Carlo programs. The main advantage over the old techniques is mainly in the capability of the most modern codes to handle problems of practically any degree of complexity.

In the Monte Carlo programs before the '90s the common approach was to simplify as much as possible, converting complicated geometries to equivalent symmetrical ones, neglecting the less important physical effects and using various types of approximations to reduce the problem to one in few dimensions, easily integrated by analytical or numerical techniques. The modern Monte Carlo codes, instead, require fewer approximations (an exception are the condensed histories, described in a

*Corresponding author: fasso@slac.stanford.edu

Work supported in part by US Department of Energy contract DE-AC02-76SF00515.

later section) and provide more accurate solutions. The possibility to cope with problems which before could not be solved at all has opened the way to a large number of new applications, one of which are of course analog simulations.

Time

Monte Carlo calculations require much more time than the old analytical techniques: this includes preparation time, time to set up biasing, CPU time and analysis time. Preparation time is mainly devoted to describe and debug the problem geometry.

A long time may be needed to set up biasing in order to reduce CPU time: an efficient biasing scheme may demand a lot of patience and testing. In many cases, the speed and low cost of modern computers allow to work with little or no biasing, especially if a computer cluster is available for parallel computations. But still problems exist where a strong biasing is needed.

It is extremely important to reserve time for a careful analysis: the results must be checked for all kinds of possible mistakes. In particular, the results must be consistent with expectations based on physical judgment: cross-checking with analytical formulae can help.

THE MATHEMATICAL BASIS OF MONTE CARLO

Most of the theoretical and mathematical foundations of Monte Carlo, as well as most basic textbooks on that technique [2, 3, 4, 5, 6] were historically centered on low-energy neutron and photon transport. However, several modern programs apply the same mathematical apparatus to the transport of charged particles and to interactions at higher energies, with some necessary additions which will be mentioned in later sections (condensed histories, decay, transport in electric and magnetic fields).

Mean of a distribution

In one dimension:

Given a variable x, distributed according to a function $f(x)$, the mean or average of another function of the same variable $A(x)$ over an interval [a,b] is given by:

$$\bar{A} = \frac{\int_a^b A(x)f(x)dx}{\int_a^b f(x)dx} \tag{1}$$

Or, introducing the normalized distribution $f'(x)$:

$$f'(x) = \frac{f(x)}{\int_a^b f(x)dx} \tag{2}$$

$$\bar{A} = \int_a^b A(x)f'(x)dx \tag{3}$$

A special case is that of $A(x) = x$:

$$\bar{x} = \int_a^b xf(x)dx \tag{4}$$

In several dimensions:

Given n variables $x, y,\ldots$, distributed according to the (normalized) functions $f'(x), g'(y)\ldots$, the mean or average of a function of those variables $A(x, y\ldots)$ over an n-dimensional domain is given by:

$$\bar{A} = \iint_{x\ y}\!\!\ldots\!\int A(x, y,\ldots)f'(x)g'(y)\ldots dxdy\ldots \tag{5}$$

An n-dimensional integral is often impossible to calculate with traditional methods, but we can sample N values of A with probability $f'g'h'\ldots$ and divide the sum of the sampled values by N:

$$S_N = \frac{\sum_1^N A(x, y, z, \ldots)f'(x)\,g'(y)\,h'(z)\ldots}{N} \tag{6}$$

Since each term of the sum is distributed like A, in this case the integration is also a simulation (Analog Monte Carlo).

Central Limit Theorem

The Central Limit Theorem [7] says that for large values of N, the distribution of averages (normalized sums S_N) of N independent random variables identically distributed (according to *any* distribution with mean $\bar{A}$ and variance $\sigma^2 \neq \infty$) tends to a normal distribution with mean $\bar{A}$ and variance σ_A^2/N:

$$\lim_{N\to\infty} S_N =$$

$$= \lim_{N\to\infty} \frac{\sum_1^N A(x, y, \ldots)f'(x)\,g'(y)\ \ldots}{N} = \bar{A} \tag{7}$$

$$\lim_{N\to\infty} P(S_N) = \frac{1}{\sqrt{\frac{2\pi}{N}}\sigma_A} e^{-\frac{(S_N-\bar{A})^2}{2\sigma_A^2/N}} \tag{8}$$

The Central Limit Theorem is the mathematical foundation of the Monte Carlo method. In words: "Given any observable A, that can be expressed as the result of a convolution of random processes, the average value of A can be obtained by sampling many values of A according to the probability distribution of the random processes".

The Monte Carlo method is indeed an integration technique that allows to solve multi-dimensional integrals by sampling from suitable stochastic distributions.

The accuracy of MC estimator depends on the number of samples:

$$\sigma \propto \frac{1}{\sqrt{N}} \tag{9}$$

Analog Monte Carlo

In an analog Monte Carlo calculation, not only the mean of the contributions converges to the mean of the actual distribution, but also the variance and all moments of higher order:

$$\lim_{N\to\infty} \left[\frac{\sum_{1}^{N} (x-\bar{x})^n}{N} \right]^{\frac{1}{n}} = \sigma_n \tag{10}$$

Then, partial distributions, fluctuations and correlations are all faithfully reproduced: in this case (and in this case only!) we have a real simulation.

PHASE SPACE

Phase Space is a concept of classical Statistical Mechanics. Each phase space dimension corresponds to a particle degree of freedom: three dimensions correspond to position in (real) space: x, y, z, and three other dimensions correspond to momentum: p_x, p_y, p_z (or to energy and direction: E, θ, ϕ). More dimensions may correspond to other possible degrees of freedom: quantum numbers (e.g. spin), particle type, etc. Each particle is represented by a point in phase space. Time can also be considered as a coordinate, or it can be considered as an independent variable: the variation of the other phase space coordinates as a function of time (the trajectory of a phase space point) constitutes a particle "history".

Phase space density

The phase space density Ψ (number of particles in an infinitesimal volume of phase space) is the most general radiometric quantity. It is defined as the derivative of fluence Φ with respect to all phase space coordinates: time, energy and direction vector:

$$\Psi = \frac{\partial \Phi}{\partial t\, \partial E\, \partial \vec{\Omega}} = \dot{\Phi}_{E\vec{\Omega}} \tag{11}$$

Ψ is known as *angular flux* in transport theory and cosmic ray physics. It is a fully differential quantity, but most Monte Carlo solutions are integrals of Ψ over one or more (or all) phase space dimensions: coordinates, time, energy, angle. Fluence Φ, on the opposite, is the most integral radiometric quantity:

$$\Phi = \int_E \int_{\vec{\Omega}} \int_t \dot{\Phi}_{E\vec{\Omega}} dE d\vec{\Omega} dt \tag{12}$$

The Boltzmann equation

The Boltzmann Equation is a balance equation in phase space: at any phase space point, the increment of particle phase space density Ψ in an infinitesimal phase space volume is equal to the sum of all "production terms" minus the sum of all "destruction terms":

$$\frac{1}{v}\frac{\partial \Psi(x)}{\partial t} + \vec{\Omega}\cdot\nabla\Psi(x) + \Sigma_t \Psi(x) - S(x) = \\ = \int_\Omega \int_E \Psi(x)\Sigma_s(x' \to x) dx' \tag{13}$$

where x represents all phase space coordinates: $\vec{r}, \vec{\Omega}, E, t$.
The various elements of the Boltzmann Equationhis have the following physical meaning:

- The $\frac{1}{v}\frac{\partial \Psi(x)}{\partial t}$ term represents the time-dependent density change, for instance due to particle decay
- $\vec{\Omega}\cdot\nabla\Psi(x)$ is the density change due to translational movement without change of energy and direction
- $\Sigma_t\Psi(x)$, where Σ_t is the total macroscopic cross section (inverse of the mean free path), is a term representing absorption
- $S(x)$ represents the particle sources
- the double integral, where Σ_s is the macroscopic scattering cross section, refers to scattering: change in density due to direction (and possibly energy) change, without a change of particle position

All particle transport calculations are explicit or implicit attempts to solve the Boltzmann Equation.

Sources and detectors

To solve the Boltzmann Equation, it is necessary to define one or more sources and one or more detectors.

A source is a region of phase space. In the most general definition, a source consists of one or more particle types, a range of space coordinates and a distribution in angle, energy and time. But in the simplest case a source is simply a monoenergetic monodirectional point source, i.e. a "pencil beam".

A detector too is a region of phase space, in which the user wants to find a solution of the Boltzmann equation. Solutions can be of different type: at a number of (real or phase) space points, averaged over (real or phase) space regions, time-dependent or stationary, etc. More in general, a detector is defined by distributions of Ψ in some of the phase space coordinates and integrated over others. It is interesting to notice the symmetry between sources and detectors: and indeed in some low-energy Monte Carlo codes they can be exchanged (adjoint mode).

The user must define a detector for each solution requested.

The integral form of the Boltzmann equation

The Boltzmann equation shown in Eq. (13) is in integro-differential form. But for Monte Carlo calculations it is more convenient to put it in integral form, carrying out the integration over all possible particle histories. A theorem of statistical mechanics, the Ergodic Theorem, says that the average of a function along the trajectories is equal to the average over all phase space. The trajectories "fill" all the available phase space.

We will change the coordinates along the line s in direction $\vec{\Omega}$:

$$\frac{1}{v}\frac{\partial \Psi}{\partial t} + \frac{d\Psi}{ds} + \Sigma_t \Psi = S + q \tag{14}$$

where q indicates the scattering integral of Eq. (13).

Now let's consider only the fraction of particle phase space density in the detector which is contributed by trajectories going directly from the source to the detector without any scattering ("uncollided term" Ψ_0):

$$\Psi_0 = S \exp\left(-\int_0^s \Sigma_t s ds\right) = S e^{-\beta} \tag{15}$$

where the quantity $\beta = \int_0^s \Sigma_t s ds$, which is called the "optical thickness", is the probability that a particle emitted by the source will reach the detector without absorption nor scattering.

The contribution by trajectories which reach the detector after one scattering ("once-collided" term Ψ_1) can be derived from Ψ_0:

$$\Psi_1 = \int_0^\infty e^{-\beta}\left[\int_E \int_{\vec{\Omega}} \Sigma_s \Psi_0 d\vec{\Omega} dE\right] ds = K\Psi_0 \tag{16}$$

where the integral is concisely represented by the K operator.

In the same way, a twice-collided contribution Ψ_2 is obtained from Ψ_1, and so on. Each term is derived from the previous one, adding one scattering. The resulting series, obtained by successive application of the operator K:

$$\Psi_0 = S e^{-\beta},\ \Psi_1 = K\Psi_0,\ \Psi_2 = K\Psi_1, \ldots, \ldots \Psi_n = K\Psi_{n-1} \tag{17}$$

is called the Neumann series (from the mathematician Carl Neumann).

The total angular flux Ψ, averaged over the detector, is given by the sum of the terms of the Neumann series: $\Psi = \Psi_0 + \Psi_1 + \Psi_2 + \cdots + \Psi_n$ [8].

It can be noticed that analytical shielding formulae for low energy photons and neutrons are written as: $D = D_0 B e^{-\Sigma x}$, where D (dose) is assumed to be proportional to Φ (fluence), $D_0 e^{-\Sigma x}$ is the uncollided term, and B (build-up factor) is the sum of all collided terms.

INTEGRATION BY MONTE CARLO

Integration efficiency

Traditional numerical integration methods (e.g., the Simpson rule) converge to the true value as $N^{-1/n}$, where N is the number of integration points or intervals and n is the number of dimensions. Integration by Monte Carlo converges as $N^{-1/2}$, independent of the number of dimensions. Therefore:

- $n = 1$: Monte Carlo is not convenient
- $n = 2$: Monte Carlo is about equivalent to traditional methods
- $n > 2$: Monte Carlo converges faster (and the more so the greater the dimensions).

With the integro-differential Boltzmann equation (13) the dimensions are the 7 of phase space, but using the integral form (14) the dimensions over which the integral is calculated are those of the largest number of "collisions" per history (the Neumann term of highest order).

Note that the term "collision" is derived from low-energy neutron/photon transport theory. Here it should be understood in the extended meaning of "interaction where the particle changes its direction and/or energy, or produces new particles".

Random sampling

The use of random sampling techniques is the distinctive feature of Monte Carlo. The central problem of the Monte Carlo technique is:

"Given a Probability Density Function (pdf) of the x variable, $f(x)$, generate a sample of x's distributed according to $f(x)$" (x can be multidimensional).

Solving the integral Boltzmann transport equation by Monte Carlo consists of two essential parts: describing the geometry and materials of the problem, and sampling randomly the outcome of physical events from probability distributions.

Random and pseudorandom numbers

The basis of all Monte Carlo integrations are random values of a variable distributed according to a pdf. In the real physical world, an experiment samples a large number of random outcomes of physical processes: these correspond, in a computer calculation, to pseudo-random numbers sampled from pdf distributions. The basic pdf is the uniform distribution: $f(\xi) = 1$, with $0 \leq \xi < 1$.

Pseudo-random numbers (PRN) are sequences that reproduce the uniform distribution, constructed from mathematical algorithms (PRN generators). A PRN sequence looks random but it is not: it can be successfully tested for statistical randomness although it is generated deterministically. A pseudo-random process

is easier to produce than a really random one, and has the advantage that it can be reproduced exactly.

PRN generators have a period, after which the sequence is identically repeated. However, a repeated number does not imply that the end of the period has been reached. The period of the generator by Marsaglia and Tsang [9] is $> 10^{61}$, and that of the "Mersenne Twister" by Matsumoto and Nishimura [10] is longer than 10^{6000}.

Sampling from a discrete distribution

Consider a *discrete* random variable x, that can assume values $x_1, x_2, \ldots, x_n$ with respective probabilities $p_1, p_2, \ldots, p_n$. Assume $\Sigma_i p_i = 1$, or normalize it.

Let us divide the interval $[0, 1)$ in n subintervals, with limits $y_0 = 0, y_1 = p_1, y_2 = p_1 + p_2, \ldots, y_n = \Sigma_1^n y_n$ (note the use of the cumulative probabilities: this is a typical feature of Monte Carlo sampling that we will meet again when dealing with sampling from continuous distributions).

Generate a uniform pseudo-random number ξ.

Find the i^{th} y-interval such that $y_{i-1} \leq \xi < y_i$ and select $X = x_i$ as the sampled value. Let $P(X)$ be the correspondent probability.

Since ξ is uniformly random:

$$P(X) = P(y_{i-1} \leq \xi < y_i) = y_i - y_{i-1} = p_i \quad (18)$$

Sampling from a generic continuous distribution

Consider a *generic continuous* pdf $f(x)$.
Integrate the distribution function, $f(x)$, analytically or numerically, and normalize to 1 to obtain the normalized cumulative distribution:

$$F(t) = \frac{\int_{x_{min}}^{t} f(x)dx}{\int_{x_{min}}^{x_{max}} f(x)dx} \quad (19)$$

Generate a uniform pseudo-random number ξ.
Get a sample of $f(x)$ by finding the inverse value $X = F^{-1}(t)$, analytically or most often numerically by table look-up and interpolation. Let $P(X)$ be the correspondent probability.

Since ξ is uniformly random:

$$P(a \leq X < b) = P[F(a) \leq \xi < F(b)] = F(b) - F(a) = \int_a^b f(x)dx \quad (20)$$

The rejection technique

Some distributions cannot be easily sampled by integration and inversion.
Let $f'(x)$ be one such distribution (normalized) that we want to sample.
Let $g(x)$ be another distribution function, also normalized, that can be sampled, such that $Cg(x) \geq f'(x)$ for all $x \in [x_{min}, x_{max}]$.
Generate a uniform pseudo-random number ξ_1 to sample X from $g(x)$.
Generate a second pseudo-random number ξ_2.
Accept X as a sample of $f'(x)$ if $\xi_2 > f'(X)/[Cg(x)]$, otherwise re-sample ξ_1 and ξ_2.
The probability of X to be sampled from $g(x)$ is $g(X)$, while the probability that it passes the test is $f'(X)/[Cg(X)]$: therefore the probability to have X sampled and accepted is the product of probabilities $g(X)f'(X)/[Cg(X)] = f'(X)/C$.
Since $f'(x)$ is normalized, the overall efficiency (probability accepted/rejected) is given by

$$\epsilon = \int \frac{f'(x)}{C} dx = \frac{1}{C} \quad (21)$$

The $g(x)$ distribution is generally chosen as a uniform (rectangular) distribution or as a normalized sum of uniform distributions, i.e. a piecewise constant function.

Other sampling techniques

Most discrete distributions, for instance the Poisson distribution, cannot be expressed by a simple enumeration of probabilities. In a similar way, many continuous distributions cannot be integrated and inverted analytically, and cannot be sampled easily by a rejection technique. Although inversion of the cumulative distribution (discrete or continuous) and rejection are the two basic sampling techniques, many other schemes have been found. Others are sometimes preferred because they are faster or easier to implement: for instance, a pdf $f(x) = x^n$ can be easily sampled by taking the largest of $n + 1$ random numbers. A very comprehensive collection of such "recipes" can be found in the "Monte Carlo Sampler" by Everett and Cashwell [11].

PARTICLE TRANSPORT MONTE CARLO

A typical Monte Carlo particle transport code works as follows: each particle is followed on its path through matter. At each step the occurrence and outcome of interactions are decided by random selection from the appropriate probability distributions. All the secondaries issued from the same primary are stored in a "stack" or "bank" and are transported before a new history is started.

Most Monte Carlo transport codes are based on a number of assumptions, not always explicitely stated, which may limit their field of application or require some approximation.

Media and geometry are generally supposed to be *static, homogeneous, isotropic and amorphous*. A static geometry makes it difficult to handle problems with moving targets, although some attempts have been made

successfully [12, 13]. The FLUKA code [19] allows to make calculations of dose rates due to induced activity simulating in a same run the prompt radiation field and that of the radioactive decay, but often radioactive decay may take place in a geometry different from that in which the radionuclides were produced, so that a two-step calculation is necessary with two different geometries [14]. The important problem of transport of cosmic rays in the atmosphere, which would require a description of the variable density of the atmosphere, is normally handled by approximating the continuous density variation by many discrete layers of uniform density [15].

Particle transport is handled as a *Markovian process*, i.e. the fate of a particle depends only on its actual present properties, and not on previous events or histories. This assumption does not seem to present particular problems.

Particles *do not interact with each other*. This is a reasonable assumption in normal situations, but not in extremely intense radiation fields as could be met in a plasma or inside a star. Also, some rare radiation interactions occurring at very high energies cannot be simulated, such as the Chudakov effect [16] (reduced energy deposition by electron-positron pairs due to the cancelling of opposite charges).

Particles *interact with individual electrons, atoms, nuclei and molecules*. This limitation does not allow to simulate effects such as X-ray mirror reflection, which are important in synchrotron radiation optics.

Material properties are not affected by particle reactions. Most Monte Carlo programs require a static material definition: but of course in these conditions it is not possible to handle problems such as burnup in nuclear reactors, where an intense neutron field modifies the isotope composition of a material. To simulate such a situation, it must be pointed out that the usual normalization per primary particle is not possible: burnup is not linear with the number of primary particles, since it depends on the source intensity. Approximations are possible by splitting the calculation in several successive steps [17].

The accuracy and reliability of a Monte Carlo depend on the models or data on which the probability distribution functions are based. The statistical accuracy of the results depends on the number of "histories". Statistical convergence can be accelerated by "biasing" techniques.

Different flavors of Monte Carlo

Microscopic analog Monte Carlo

This type of Monte Carlo uses theoretical models to describe physical processes whenever it is possible, samples from actual physical phase space distributions and predicts average quantities and all statistical moments of any order. Codes belonging to this family, preserving correlations and reproducing fluctuations in the best way allowed by the physics models used, can really be considered simulation codes. They are reasonably safe and generally do not require particular precautions by the user, but are often inefficient and converge slowly. Since this type of Monte Carlon samples from the actual modeled physical distributions, it can fail to predict contributions due to rare but important events.

Macroscopic Monte Carlo

This type of particle transport, called also parametrized Monte Carlo, instead of simulating interactions in detail, uses parametrizations of the reaction product distributions, obtained from fits to data and extrapolations. It is faster than analog microscopic simulations, especially when there are complex reactions, and can be more accurate if the theory contains uncertainties or approximations. It reproduces the single probability distribution functions, but not the correlations among the products of the interactions. Of course, macroscopic Monte Carlo cannot be extended outside the range of the data used for the parametrizations.

Model-based and table-based codes

Originally, before 1980, there were three different types of Monte Carlo radiation transport codes:

- low-energy neutron-photon codes, developed mainly for nuclear reactor problems (criticality and shielding)
- high-energy hadron codes, designed for accelerator shielding or for high-energy physics
- electron-photon codes

Presently, some codes in the first category (for instance MCNP [18]) and some belonging to the second one (e.g., FLUKA [19]) have been extended to cover the whole energy range and to transport a large number of particles, including electrons. However, in the process of extension they have kept some aspects of the original structure. While the cross sections in the low-energy neutron transport codes were based on tabulations derived from evaluated nuclear data files, those used by the high-energy codes were mainly calculated from physical models. In all modern codes it has been necessary to rely on a mixed system where both approaches are followed, each in a different energy range: but while the MCNP developers have made an effort to extend as much as possible the tabulation approach, those of FLUKA have preferred to keep it to a minimum. The advantage of tabulation-based codes is mainly sampling speed, while model-based codes can describe correlations between secondary particles, and offer some predictivity even when experimental data are lacking.

Biased Monte Carlo

In biased Monte Carlo one can sample from artificial distributions, applying a weight to the particles to correct for the bias. In a similar way, the differential is modified when calculating an integral by a change of variable. This form of Monte Carlo predicts average quantities, but not the higher moments: on the contrary, its goal is to minimize the second moment. Indeed, getting the same mean with smaller variance results in a faster convergence. Biasing allows sometimes to obtain acceptable statistics where an analog Monte Carlo would take years of CPU time to converge, but cannot reproduce correlations and fluctuations. It must also be pointed out that biased Monte Carlo makes only privileged observables converge faster (some regions of phase space are sampled more than others).

GEOMETRY

The algorithms to build a geometry and to track particles inside it differ from code to code. In some codes the geometry is built from basic solids, in others from surfaces, and in some others from both. The user describes the problem geometry by means of input "cards" (text lines) or, in some codes, by user-written routines. In some cases, it is possible to define repeated structures, obtained by translations or rotations of basic prototypes. Complex geometries such as that of a human body can be described in some Monte Carlo codes by means of "voxels" (small elementary parallelepipeds): this feature is typically used to import Computed Tomography scans.

MONTE CARLO EVENTS

Particle histories are sequences of various events or processes, physical (e.g. a particle interaction with an atom or with a nucleus) or referring to transport through the geometry, for instance crossing a boundary between two materials. Events belong to two categories: discrete (or point-like) and continuous. Actual physical events are essentially discrete, but for convenience certain sequences of many similar, microscopic events are described as a continuous macroscopic one, sampled from a suitable distribution.

Discrete processes

Discrete physics processess include atomic interactions by photons (Compton, photoelectric effect, pair production, coherent scattering), and by charged particles (bremsstrahlung, δ ray emission, large angle Coulomb scattering). Each of these processes are sampled only when the energy of a particle is higher than given thresholds, built-in or set by the user. Nuclear interactions include absorption and nuclear scattering (elastic and non-elastic). Decays are also a type of discrete physics processes.

Point-like transport processes are boundary crossing and escape from the problem geometry.

Continuous processes

Charged particles lose energy and change direction as a result of thousands of discrete collisions with atomic electrons. To simulate in detail each collision would require prohibitive computer times, except at very low particle energies. Therefore, generally many discrete scatterings are replaced by a straight continuous step, and the corresponding energy losses and changes of direction are "condensed" into a sum of losses (stopping power) and an overall scattering angle. This approach, common to most Monte Carlo particle transport programs, is known as the condensed-history technique [20]. Some programs, however, provide a user option to simulate in detail the single scatterings in particular situations (very low energies, boundary crossing, conditions required by a multiple scattering theory not satisfied).

Ionization energy losses belong to this type of continuous physical processes: all losses lower than a preset threshold are continuously distributed along a particle step. Any loss larger than threshold is simulated as a discrete energy imparted to an electron which is then transported separately (δ ray). Energy loss fluctuations can be simulated for the losses below threshold.

Multiple Coulomb Scattering is another aspect of the same continuous process: a deflection angle, sampled from a theoretical distribution, is applied to each particle step. Some corrections are needed, to account for the ratio between the length of the straight step and the actual path length (Path Length Correction, PLC), for the lateral displacement, etc. [21]

Other continuous transport events are neutral particle displacements between interactions or boundaries and particle displacements in vacuum, also implemented as "steps". When magnetic or electric field are present, charged particle steps must be subdivided into smaller steps to follow the curvature of the trajectory while keeping the dE/dx and multiple scattering approximations.

Thresholds and cut-offs

Transport and production thresholds are needed because of the limits of validity of the physics models, and also to reduce computer time.

Transport thresholds

When the energy of a particle becomes lower than a specified transport threshold, transport of that particle is terminated and its remaining energy is deposited at that point, or better, in the case of a charged particle, is distributed along its residual range. The user's choice

of a threshold will depend on the "granularity" of the geometry or of the scoring mesh and on the interest in a given region. To reproduce correctly electronic equilibrium, neighboring regions should have the same electron energy threshold, and *not* the same range threshold. Because photons travel longer paths than electrons of the same energy, photon thresholds should be lower than electron thresholds.

Production thresholds

Energy thresholds need to be set also for explicit production of secondaries by photons and electrons. A δ-ray threshold sets the limit between discrete and continuous ionization energy losses. In a similar way, an energy threshold can be set for explicit production of bremsstrahlung photons: below that threshold, the electron radiative energy loss will be included in dE/dx stopping power.

BIASING

Biased sampling, namely sampling from non-natural probability distributions, has the purpose to accelerate statistical convergence. This goal can be pursued in two different, and often complementary, ways: reducing the variance of a detector score for a same computer time, or reducing the computer time needed to attain the same variance. To evaluate the effectiveness of a biasing technique, it is customary to define a figure of merit for an estimator called "computer cost" [22]:

$$F = \sigma^2 t \tag{22}$$

where σ^2 is the variance of a detector score and t the CPU time per primary particle.

Some biasing techniques are aiming at reducing σ^2, others at reducing t, but all of them are generally referred to as variance reduction techniques. Often reducing σ^2 increases t and vice versa. The variance σ^2 converges like $1/N$ (where N = number of particle histories), while the computer time t is obviously proportional to N: therefore minimizing $\sigma^2 t$ means reducing σ^2 at a faster rate than t increases or vice versa. The choice depends on the problems, and sometimes the combination of several techniques is the most effective.

Bad judgment, or excessive "forcing" on one of the two variables, can have catastrophic consequences on the other one, making computer cost "explode".

The two basic rules of biasing

In the Boltzmann equation (eq. (13),(14),(15)), there are two ingredients: the particle phase space density $\Psi(x)$ and various operators acting on $\Psi(x)$ (where x stands for all phase space coordinates). Some of the operators are based on probability distribution functions. Both the phase space density and the pdf's can be biased, assigning to each particle a statistical weight.

The particle phase space density $\Psi(x)$ can be replaced by a fictitious density $\Psi'(x)$. The particle weight $w(x)$ must be replaced by a new weight $w'(x)$ such that $w\Psi(x) = w'(x)\Psi'(x)$. For instance, one particle can be replaced by two identical particles with half its weight and with the same position, energy and direction.

First rule of biasing:

$$\underline{\text{weight} \times \text{particle density must be conserved}}$$

The pdf-based operators appearing in the Boltzmann equation are:

- Source S: space-energy-angular probability distribution
- $e^{-\beta}$: probability distribution of distance to interaction
- Scattering cross section Σ_s: double-differential probability distribution in energy and angle

A pdf $P(x)$ can be replaced by a fictitious pdf $P'(x)$. The particle weight $w(x)$ must be replaced by a new weight $w'(x)$ such that $w(x)P(x) = w'(x)P'(x)$.

Second rule of biasing:

$$\underline{\text{weight} \times \text{probability must be conserved}}$$

Importance Biasing

Importance biasing acts on the particle density so that the density sampled in a given phase space region is proportional to its contribution to the score by a detector of interest. The most common form of importance biasing combines two techniques: Surface Splitting (also called Geometry Splitting), which reduces σ^2 but increases t, and Russian Roulette, which does the opposite. This kind of importance biasing is the simplest and easiest to use of all variance reduction techniques. If used alone, it is also very safe, since it introduces only very small or zero weight fluctuations in a same space region.

The user assigns a relative importance to each geometry region or cell (the actual absolute value doesn't matter), based on one or both of the following criteria:

1. the expected fluence attenuation in that region with respect to other regions
2. the expected probability of contribution to score by particles entering that region

Importance biasing is commonly used to keep a constant particle population, compensating for attenuation due to absorption or distance (first criterium), or to reduce sampling in space regions which are not likely to contribute to result (second criterium).

Surface Splitting

If a particle crosses a region boundary, coming from a region of importance I_1 and entering a region of higher importance $I_2 > I_1$, the particle is replaced on average by $n = I_2/I_1$ identical particles with the same phase space coordinates (type, position, energy, direction). The weight of each "daughter" is multiplied by $1/n = I_1/I_2$: as always in correct biasing, weight is conserved. If I_1/I_2 is too large, excessive splitting may occur with codes which do not provide an appropriate protection.

Russian Roulette

Russian Roulette (RR) acts in the opposite direction of splitting: if a particle crosses a boundary of importance I_1 to one of lower importance $I_2 < I_1$, the particle is submitted to a random survival test: with a chance I_2/I_1 the particle survives with its weight increased by a factor I_1/I_2, and with a chance $(1 - I_2/I_1)$ the particle is killed (its weight is set to zero). Again, the weight is conserved (on average):

$$\frac{I_2}{I_1} \times \frac{I_1}{I_2} + \left(1 - \frac{I_2}{I_1}\right) \times 0 = 1 \tag{23}$$

Weight Window

The Weight Window technique too is a combination of splitting and Russian Roulette, but it is based on the absolute value of the weight of each individual particle, rather than on relative region importance. The user sets an upper and a lower weight limit, generally as a function of region, energy and particle. Particles having a weight larger than the upper limit are splitted so that the weight of their daughters will get a value between the limits (i.e., it will be brought "inside the window"). Particles having a weight smaller than the lower limit are submitted to Russian Roulette: they are killed or have their weight increased to bring them "inside the window", depending on a random choice. The new weight is calculated so that average weight is conserved. The Weight Window is a more powerful biasing tool than simple RR/splitting based on region importance, but requires also more experience and patience to set up correctly, since the expected absolute weights are not known a priori. It has often been said that "it is more an art than a science", but a "weight window generator" has been implemented in the MCNP program [18], which helps the user in this task [23], although it requires some experience to generate useful results and entails some additional CPU cost.

A Weight Window, possibly energy dependent, is essential when other biasing techniques generate excessive weight fluctuations in a given phase space region, since large weight fluctuations tend to generate large variances.

The effect of a Weight Window can be understood as follows. Killing a particle with a very low weight (with respect to the average for a given phase space region), decreases the computer time per history, but has very little effect on the score (and therefore on the variance of the score). On the other hand, splitting a particle with a very large weight increases the computer time per history, in proportion to the number of additional particles to be tracked, but at the same time reduces the variance by avoiding large fluctuations in the contributions to scoring. The global effect is to reduce the figure of merit $\sigma^2 t$.

A too wide window is of course ineffective. But too narrow windows should also be avoided, otherwise too much CPU time would be spent in repeated applications of splitting and Russian Roulette cancelling each other. A typical ratio between the upper and the lower edge of the window is about 10. It is also possible to do Russian Roulette without splitting (setting the upper window edge very high) or splitting without Russian Roulette (setting the lower edge $= 0$).

Time reduction: Leading Particle Biasing

Leading Particle Biasing (LPB), available only for electrons, positrons and photons, is used to avoid the geometrical increase with energy of the number of particles in an electromagnetic shower [24]. In every electromagnetic interaction two particles are present in the final state (at least in the approximations made by most Monte Carlo codes). With LPB, only one of the two particles is randomly kept, and its weight is adjusted to conserve weight×probability. The most energetic particle is kept with higher probability, proportional to its energy, as the one which is more efficient in propagating the shower, and its weight is adjusted to account for the bias.

Leading Particle Biasing is very effective at reducing the CPU time per history t, but increases the score variance σ^2 by introducing large weight fluctuations, since a few low energy particles end carrying a large weight, while many energetic particles will have a small weight.

In theory, therefore, LPB should be backed up by a Weight Window. But experience has shown that the variance increase with respect to a few time-consuming showers is partially balanced by a better sampling of phase space in the many biased showers, each with its vertex at a different position, which are handled within the same CPU time. LPB is essential for shielding calculations at high energy electron accelerators: to simulate in analog mode a multi-GeV electromagnetic shower would take an extremely long computer time: at the maximum of a shower produced by a single 20 GeV electron in iron, the number of particles to be transported (photons, electrons and positrons) is about 1000! But LPB is important also for shielding calculations at proton accelerators, to avoid spending too much time dealing with electromagnetic showers due to π^0

decay. It is also possible to activate LPB only when the parent particle has an energy lower than a user-defined threshold.

A warning is necessary: with LPB, as with many biasing schemes, energy is conserved only on average.

Non-analog absorption

Non-analog absorption, also called survival biasing, is applicable to low-energy neutron transport. There are three possibilities to handle neutron scattering and absorption (we will indicate with σ_s the scattering cross section and with σ_t the total cross section):

1. Analog: at each collision scattering and absorption are sampled with the actual physical probability: σ_s/σ_t and $(1-\sigma_s/\sigma_t)$ respectively. If scattering is selected, the weight is unchanged; if absorption, the weight becomes zero:

2. Biased without absorption: systematic survival, weight reduced by a factor σ_s/σ_t:

3. Biased with user-defined absorption probability u. Scattering has probability $1-u$. The weight is reduced by a factor $\frac{\sigma_s}{\sigma_t(1-u)}$:

Option 2. is a special case of Option 3. with $u=0$. Option 3. is available in the FLUKA code [19]. Note the exchanges between probabilities and weights, such that the product Probability$\times$weight will remain unchanged.

A small survival probability is often assigned to thermal neutrons to limit the number of scatterings in non-absorbing media; it can also be very useful in materials with unusual scattering properties such as iron. On the other hand, survival probabilities too small with respect to the physical one σ_s/σ_t may introduce large weight fluctuations due to the very different number of collisions suffered by each neutron: in these cases, a Weight Window should be applied.

Biased survival without absorption makes a neutron to remain alive forever, except for escaping from the geometry. After many collisions the neutron will have a very small weight. In codes where this option is chosen, Russian Roulette or Weight Window are needed at each collision to avoid tracking neutrons with a weight too small to make a significant contribution to score.

Biasing mean free paths

Decay length

The mean life or the average decay length of unstable particles can be artificially shortened, to increase the generation rate of decay products. This technique is typically used to enhance statistics of muon or neutrino production. The kinematics of decay (decay angle) can also be biased to enhance sampling in a preferred direction.

Interaction length

In a similar way, the hadron or photon mean free path for nuclear inelastic interactions can be artificially decreased by a predefined particle or material dependent factor. This option is useful for instance to increase the probability of beam interaction in a very thin target or in a material of very low density.

Interaction length biasing is also necessary to sample photonuclear reactions with acceptable statistics, since the photonuclear cross sections are much smaller than those for electromagnetic processes.

Exponential transformation

The mean free path biasing for inelastic interactions can also be biased in the opposite way, by increasing it, possibly in a preferred direction (path-length stretching). This can be useful in case of attenuation in a very thick shield. The exponential transformation accelerates convergence at large shielding depths, but slows it down at small depths. In general, it should be accompanied with a Weight Window, since it can generate large weight fluctuations due to the different ways a particle can reach a given depth. Experience shows that a similar biasing efficiency can be obtained by geometry splitting, without introducing fluctuations.

User-written biasing

Provided each particle within the program can be assigned a weight, it is possible for users to write their own biasing. The case most often encountered is that of a biased source energy spectrum, in which some energies are preferentially sampled, but it is also common to bias the source spatial and/or angular distribution. Of course, in such cases it is the user's responsibility to ensure that weights are adjusted in a statistically correct way.

Sampling from a biased distribution

The sampling technique based on the cumulative pdf can be extended to modify the sampling probability in different parts of the interval (importance sampling).

We replace $f(x)$ by $g(x)=f(x)h(x)$, where $h(x)$ is any appropriate weight function. We normalize $f(x)$ and $g(x)$:

$$f'(x)=\frac{f(x)}{\int_{x_{min}}^{x_{max}} f(x)dx}=\frac{f(x)}{A} \tag{24}$$

$$g'(x)=\frac{g(x)}{\int_{x_{min}}^{x_{max}} g(x)dx}=\frac{g(x)}{B} \tag{25}$$

and consider the biased cumulative normalized distribution $G(x)$:

$$G(x) = \frac{\int_{x_{min}}^{x} g(x)dx}{B} \tag{26}$$

Now let us sample from the biased cumulative normalized distribution G instead of the original unbiased F: let us take a uniform random number ξ, and get the sampled value X by inverting $G(x)$:

$$X = G^{-1}(\xi) \tag{27}$$

The particle weight must be multiplied by the ratio of the unbiased to the biased normalized pdf at $x = X$:

$$w' = w\frac{f(X)}{g(X)} = w\frac{B}{A}\frac{1}{h(X)} \tag{28}$$

A special case is that when the biasing function chosen is the inverse of the unbiased pdf:

$$h(x) = \frac{1}{f(x)} \qquad g(x) = f(x)h(x) = 1 \tag{29}$$

$$G(x) = \frac{\int_{x_{min}}^{x} g(x)dx}{\int_{x_{min}}^{x_{max}} g(x)dx} = \frac{x - x_{min}}{B} = \\ = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{30}$$

In this case, $X = G^{-1}(\xi) = x_{min} + \xi(x_{max} - x_{min})$ and the weight of the sampled particle is multiplied by

$$\frac{B}{A}\frac{1}{h(X)} = \frac{x_{max} - x_{min}}{A} f(X) \tag{31}$$

Because X is sampled with the same probability over all possible values of x, independently of the value $f(X)$ of the function, this technique is used to ensure that sampling is done uniformly over the whole interval, even though $f(x)$ might have very small values in some x range. For instance, it may be important to avoid undersampling in the high-energy tail of a spectrum, steeply falling with energy but more penetrating at high energies, such as that of cosmic rays or synchrotron radiation.

RESULTS FROM A MONTE CARLO CALCULATION

Estimators

It is often said that a Monte Carlo calculation is a "mathematical experiment" [5, 25]. Each aspect of a real experiment has indeed its Monte Carlo equivalent:

Experimental technique ⇒ Estimator
Instrument ⇒ Detector
Measurement ⇒ Score, or Tally
Result of an experiment ⇒ Monte Carlo result

Just as a real measurement, a score is obtained by sampling from a statistical distribution. As an experimental result consists in an average of measurement values, a statistical error and a systematic error, a Monte Carlo result is an average of scores, a statistical error, and a systematic error generally unknown. There are often several different techniques to measure the same physical quantity: in the same way the same quantity can be calculated with different kinds of estimators.

Estimator types

There are various types of estimators, depending on the quantity to be estimated and on its topology (phase space region over which the quantity is integrated).

The Boundary Crossing estimator is used to estimate the fluence or the current of particles at a physical boundary between two space regions. Possible results are mono or multi-differential fluence spectra, as a function of energy, angle, particle type, etc.

The Track Length estimator calculates the fluence of particles in a region of real space. The results are fluence spectra as a function of particle energy and type, based on their path lengths within the region volume.

A Pulse Height estimator is used to simulate the response of a spectrometer (for instance a Ge detector). The quantity estimated is the energy deposited in a region of real space, and the result is the spectrum of deposited energy within the region volume.

Scalar integral estimators are used to predict scalar quantities, or their densities, such as deposited energy, inelastic interactions ("stars"), induced activity, etc. in a region of real space .

A Mesh (or Binning) estimator is a special case of Scalar estimator, providing a 2D or 3D space distribution of a scalar quantity (scalar fluence, energy deposition, star density) over a regular subdivision of a portion of real space in sub-volumes, generally independent from the tracking geometry.

Estimator extensions

Most Monte Carlo codes have built-in estimators, to be activated and tailored by the user. The results are usually averaged over one run and normalized to one primary particle. Additional flexibility can be achieved by convolution, off-line or on-line, with energy-dependent conversion factors. For instance, differential fluence can be convoluted with conversion coefficients to obtain effective dose or ambient dose equivalent. Other possible extensions can be obtained by event-by-event estimators for correlated data analysis, or by full or partial dumping of events: steps, interactions, etc, for off-line analysis.

Detectors

While an estimator is a technique to "measure" a certain quantity, a detector is an instantiation (a concrete application) of an estimator in a particular region of phase space. For instance, a track-length estimator of fluence

can be concretely applied as a particular detector consisting in a sphere of given radius centered at specified coordinates. Often a Monte Carlo user wants to get a result at one or more detectors, and has no interest in what happens elsewhere. In this case, biasing can be used to accelerate convergence in the neighborhood of a detector, at the expense of other parts of phase space (but if biasing is done correctly, for $N \to \infty$ the integration will converge to the true value everywhere, although at a different speed). Thanks to the modern fast computers, however, it is often possible to obtain a good result with a multiple detector (a mesh detector) covering a large region of real space. This kind of detector can be very useful to identify shielding weaknesses in unexpected places, as revealed for instance by color plots.

Statistical errors

The variance of the mean of an estimated quantity x (e.g., fluence), calculated in N batches, is given by:

$$\sigma^2_{<x>} = \frac{1}{N-1}\left[\frac{\sum_1^N n_i x_i^2}{n} - \left(\frac{\sum_1^N n_i x_i}{n}\right)^2\right] \tag{32}$$

where n_i = number of histories in the i^{th} batch, $n = \sum n_i$ = total number of histories in the N batches, x_i = average of x in the i^{th} batch.

The variance can be calculated for single histories (in the limit $N = n$, $n_i = 1$), or for batches of several histories each (not necessarily the same identical number).

The distribution of scoring contributions by single histories can be very asymmetric, since many histories contribute little or zero. Scoring distributions from batches tend to Gaussian for $N \to \infty$, provided $\sigma^2 \neq \infty$ (Central Limit Theorem).

The standard deviation of an estimator calculated from batches or from single histories is an estimate of the standard deviation of the actual distribution ("error of the mean"). How good such an estimate is, depends on the type of estimator and on the particular problem, but it converges to the true value for $N \to \infty$.

Effect of sampling inefficiency on statistical errors

The following table, adapted from the MCNP Manual [18], suggests that the actual meaning of a calculated statistical error may be different in Monte Carlo from that expected based on the statistics of experiments.

Relative error	Quality of tally
50 to 100%	Garbage
20 to 50%	Factor of a few
10 to 20%	Questionable
$< 10\%$	Generally reliable

Why does a 30% calculated error mean in fact an uncertainty of a "factor of a few"? Because the actual error corresponds to the sum (in quadrature) of two uncertainties: one due to the fraction of histories which give a zero contribution, and one which reflects the spread of the non-zero contributions. The MCNP guideline is empirically based on experience, not on a mathematical proof, but it has been generally confirmed also in other codes.

Other errors

Just as in experimental measurements, in addition to statistical errors, Monte Carlo results can be affected by systematic errors and in some unfortunate cases even by mistakes.

Systematic errors due to code weaknesses

A frequent case of systematic error is due to code weaknesses. Different codes are based on different physics models and some models are better than others, or are better in a certain energy range. Common code weaknesses are the presence of artifacts due to imperfect algorithms, e.g., energy deposited at a point in the middle of a step, inaccurate path length correction for multiple scattering, missing correction for cross section and dE/dx change over a step, etc.

While model quality is best tested by experimental benchmarks at the microscopic level (e.g. thin target experiments), good tests of algorithm quality are generally provided by benchmarks at the macroscopic level (thick targets, complex geometries).

Some uncertainty is also unavoidable concerning the experimental data on which a Monte Carlo code is based. An error of 1% in the absorption cross section can lead to an error of nearly a factor 3 in the shielding attenuation of a thick wall (10 attenuation lengths). Results can never be better than allowed by available experimental data.

Systematic errors due to lack of information

Not all necessary information needed to describe the problem is always available to the user: for instance material composition is not always well known. In particular concrete and soil composition, important for shielding asnd environmental calculations, are difficult to obtain with good accuracy. The water content can affect critically the effectiveness of concrete [26], but it is generally unknown and changing with time. Beam losses are one of the most important parameters in accelerator shielding, but most of the time they can only be guessed. Another source of uncertainty is the possible

presence of additional not well defined material (cables, supports. . .)

Systematic errors, due to simplification

Many geometries cannot be reproduced exactly or would require too much effort, and therefore must be necessarily simplified. Air contains humidity and pollutants and has a density variable with atmospheric pressure. In general, these details are neglected in most calculations.

Mistakes and bugs

Code bugs

Unfortunately Monte Carlo codes can contain bugs. Physics bugs, due to poor design, have been found in some codes in the past, e.g. non-uniform azimuthal scattering distributions, energy non-conservation, incorrect extrapolation of cross section data, etc. Programming bugs also may exist, as in any other software, but tend to become less frequent as a code becomes more mature.

User mistakes

Of course, as in any other kind of calculation, the user can make mistakes. But the preparation of the input for a Monte Carlo calculation can be very complex, and the probability of user errors is not negligible. This is even more so in case the input includes the writing of user code. Some frequent mistakes [27] are the following:

- mis-typing the input: some codes are more or less good at checking, but the final responsibility is the user's
- error in user code: the built-in features should be used as much as possible
- wrong units
- wrong normalization of the results
- unfair biasing: energy/space cuts cannot be avoided, but must be done with much care
- forgetting to check if gamma production is available in the neutron cross sections used

QUALITY ASSURANCE

Many enforcing authorities require that Quality Assurance (QA) be applied to shielding design procedures. But imposing it on Monte Carlo calculations can only be done in a very limited way. Monte Carlo is not a "black box", and different users are likely to choose different approaches, equally valid, to the same problem. Among other difficulties, a strict QA would probably require forbidding to write user code, and would be incompatible with the use of most variance reduction techniques ("they are more an art than a science", as reminded above). However, QA can be required on keeping proper documentation about code version, input, outputs, applied biasing, possible user code, assumptions, normalization, etc. Other recommended QA features are that critical calculations be submitted to peer reviews and to audits made by independent experts.

FUNDING

This work was supported by the Department of Energy [grant number DE-AC-03-76SF00515].

REFERENCES

1. Householder,A.S. (1949) Proc. of a symposium on the Monte Carlo Method, Los Angeles June 27–July 1, 1949. Cited in the preface of the Proc. of a symposium on Monte Carlo Methods, University of Florida, March 16–17, 1954, ed. by H.A. Meyer, Wiley, New York 1963
2. Kalos,M.H. and Whitlock,P.A (2008) Monte Carlo Methods, 2nd edition, Wiley-VCH, Berlin
3. Lux,I. and Koblinger, L. (1990) Monte Carlo Particle Transport Methods: Neutron and Photon Calculations, CRC Press, Boca Raton, FL
4. Carter,L.L. and Cashwell,E.D. (1975) Particle-Transport Simulation with the Monte Carlo Method, ERDA Crit. Serv. Ser., National Technical Information Service, Springfield, MA
5. Hammersley,G.M. and Handscomb,D.C. (1964) Monte Carlo Methods, Wiley New York
6. Spanier,J. and Gelbard,E.M. (1969) Monte Carlo Principles and Neutron Transport Problems, Addison-Wesley, Reading, MA
7. Pólya,G. (1920) Über den zentralen Grenzwertsatz der Wahrscheinlichkeitsrechnung und das Momentenproblem (in German) *Mathematische Zeitschrift* **8**, 171–181
8. Dupree,S.A. and Fraley,K. (2002) A Monte Carlo primer, Kluwer/Plenum, New York
9. Marsaglia,G. and Tsang,W.W. (2004) The 64-bit universal RNG, *Statistics & Probability Letters* **66**, 183–187
10. Matsumoto,M. and Nishimura,T. (1998) Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Trans. Modeling Computer Simulation* **8**, 3–30
11. Everett,C.J. and Cashwell, E.D. (1983) A Third Monte Carlo Sampler, Los Alamos Report LA–9721–MS
12. Biaggi,M., Ballarini,F., Burkard,W., Egger,E., Ferrari,A. and Ottolenghi,A. (1999) Physical and biophysical characteristics of a fully modulated 72 MeV therapeutic proton beam: model predictions

and experimental data. *Nucl. Instrum. Meth.* **B159**, 89–100

13. Paganetti,H., Jiang,H. and Trofimov,A. (2005) 4D Monte Carlo simulation of proton beam scanning: modelling of variations in time and space to study the interplay between scanning pattern and time-dependent patient geometry, *Phys. Med. Biol.* **50**, 983–990
14. Brugger,M., Khater,H., Mayer,S., Prinz,A., Roesler,S., Ulrici,L. and Vincke,H. (2005) Benchmark studies of induced radioactivity produced in LHC materials, Part II: remanent dose rates, *Radiat. Prot. Dosim.* **116**, 12–15
15. Battistoni,G., Ferrari,A., Muraro,S. and Sala,P.R. (2007) Atmospheric muon simulation using the FLUKA MC model, *Nucl. Phys. B (Proc. Suppl.)* **168**, 286–288
16. Chudakov,A.E. (1955) *Izv. Akad. Nauk SSSR, Ser. Fiz.* **19**, 650
17. Rubbia,C. et al. (1995) Conceptual Design of a Fast Neutron Operated High Power Energy Amplifier, CERN report CERN/AT/95-44 (EET)
18. X-5 Monte Carlo Team (2003) MCNP: a general Monte Carlo N-particle transport code, Los Alamos Report No. LA-UR-03-1987
19. Battistoni,G., Muraro,S., Sala,P.R., Cerutti,F., Ferrari,A., Roesler,S. Fassò,A. and Ranft,J. (2007) The FLUKA code: Description and benchmarking, Proc. of the Hadronic Shower Simulation Workshop 2006, Fermilab 6–8 September 2006, M. Albrow, R. Raja eds., AIP Conference Proceeding 896, 31–49;
    Ferrari,A., Sala,P.R., Fassò,A. and Ranft,J. (2005) FLUKA: a multi-particle transport code, CERN–2005–10, INFN/TC_05/11, SLAC–R–773
20. Berger,M.J. (1963) Monte Carlo calculation of the penetration and diffusion of fast charged particles, in Methods in Computational Physics, ed. by B. Alder, S. Fernbach and M. Rotenberg (Academic, New York, 1963), Vol. 1, pp. 135–215
21. Ferrari,A., Sala,P.R., Guaraldi,R. and Padoani,F. (1992) An improved multiple scattering model for charged particle transport, *Nucl. Instrum. Meth. in Phys. Res.* **B71**, 412–426
22. Juzaitis,R.J. (1980) Minimizing the cost of splitting in Monte Carlo radiation transport simulation, Thesis, Los Alamos Report LA–8546–T
23. Booth,T.E. and Hendricks,J.S. (1984) Importance estimation in forward Monte Carlo calculations, *Nucl. Technol. Fusion* **5**, 90.
24. Van Ginneken,A. (1978) AEGIS – a Program to Calculate the Average Behavior of Electromagnetic Showers, Fermilab-FN-309
25. Galison,P. (1996) Computer simulations and the trading zone, in "The Disunity of Science: Boundaries, Contexts, and Power", ed. by P. Galison and P.J. Stump, Stanford University Press, Stanford
26. Brandl,A., Hranitzky,C. and Rollet,S. (2005) Shielding variation effects for 250 MeV protons on tissue targets, *Radiat. Prot. Dosim.* **115**, 195–199
27. Stevenson,G.R., Fassò,A. and Hoefert, M. (1994) Designing accelerators without the perfect simulation code, Proc. 1st Workshop on Simulating Accelerator Radiation Environments (SARE 1), Santa Fe (New Mexico) 11-15 January 1993, Los Alamos Report LA–12835–C, pp. 41–47