Chapter 5
Monte Carlo Modeling of Light Transport in Tissue (Steady State and Time of Flight)

Steven L. Jacques

5.1 Introduction

Monte Carlo simulations are a fundamental and versatile approach toward modeling light transport in tissues. While diffusion theory for light transport is a fast and convenient way to model light transport, it fails when close to sources or boundaries and when absorption is strong compared to scattering; in other words, whenever conditions cause the gradient of fluence rate (or photon concentration) to not be simply linear but to have some curvature. Monte Carlo steps in to treat problems when diffusion theory fails. Figure 5.1 illustrates a Monte Carlo simulation.

In the general Monte Carlo simulation, "photons" are inserted into tissue at a location defined by x, y, z coordinates with a trajectory defined by directional cosines (projection of trajectory onto x, y and z axes). The random distance traveled before the photon interacts with the tissue is based upon the selection of a random number [0,1] and the local attenuation coefficient of the medium. At the end of each photon step, the weight of the photon is reduced by absorption. The remaining non-absorbed weight is redirected according to a scattering (or "phase") function that describes the angular dependence of single scattering by the particular tissue. Once a new trajectory is specified, the photon is again moved a random distance. The details of the photon path, absorption, scattering, reflection, and refraction are described in the following paragraphs.

Refraction at mismatched boundaries and even changes in local optical properties can be included in the simulation. The heat generated, S [W/cm^3], within a small volume depends upon the total photon weight absorbed in the volume, the total number of photons, and power of the laser beam. Total energy absorbed [J/cm^3] can be computed when the energy [J] of the light source is given.

The Monte Carlo method is a widely used approach toward sampling probability density functions for simulating a wide range of problems. The first use of the

S.L. Jacques (✉)
Biomedical Engineering Department, Oregon Health and Science University, Portland, OR, USA
e-mail: jacquess@ohsu.edu

A.J. Welch, M.J.C. van Gemert (eds.), *Optical-Thermal Response of Laser-Irradiated Tissue*, 2nd ed., DOI 10.1007/978-90-481-8831-4_5,

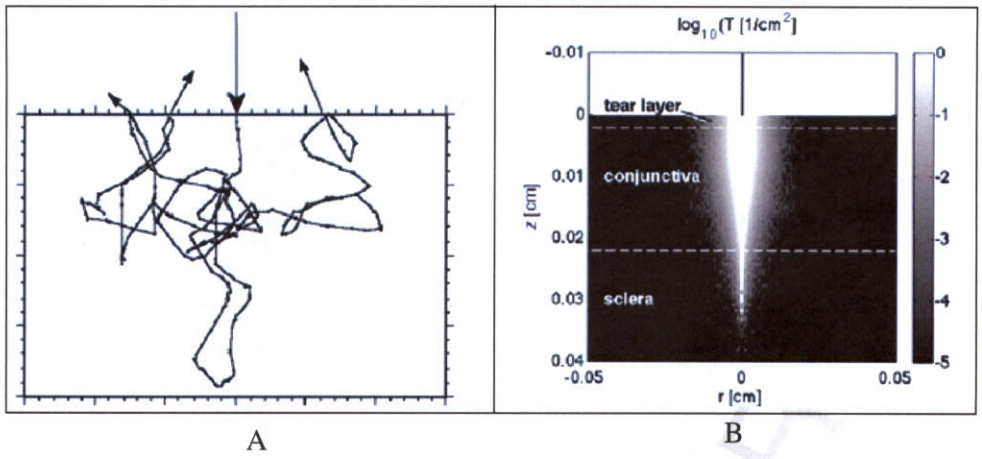

Fig. 5.1 A Monte Carlo simulation of photon transport. (**a**) Movement of a single photon in a light-scattering tissue. When the photon is totally internally reflected at the air/tissue surface, a fraction of the photon weight is allowed to escape as reflectance and the remaining photon weight continues to propagate in the tissue. In this example for the sake of illustration, the third time the photon encounters the surface, the photon is allowed to fully escape. (**b**) Superposition of millions of photons recorded as relative fluence rate, [W/cm^2] per W delivered, which is equivalently stated as transport T [1/cm^2]. The example is the penetration of a CO_2 laser (10.6 μm wavelength) into the conjunctiva of the eye, which illustrates how Monte Carlo can also described the scattering in a situation where absorption dominates over scattering, a situation not well described by diffusion theory

Monte Carlo method for photon transport in biological materials was Adams and Wilson (1983), which considered isotropic scattering [1]. Keijzer et al. (1987) introduced anisotropic scattering into the Monte Carlo simulation of biological tissues, implementing a simulation that propagated photons using cylindrical coordinates, which introduced the Hop/Drop/Spin nomenclature for organizing the program [2]. Prahl et al. (1989) reformulated the program using photon propagation based on Cartesian coordinates, which made the program much simpler to convey in written form [3]. Wang and Jacques (1993) adapted and augmented the work of Keizer and Prahl to write the Monte Carlo Multi-Layered (`MCML`) program that considers tissues with many planar layers with different optical properties [4]. `MCML` is a well-organized program with a simple text input file that the user can modify to specify different problems, which allows various problems to be run without needing to recompile the program each time. `MCML` has been widely promulgated via the web as source code [5–6], and modified by various groups to handle a variety of problems. Jacques (1998) reported on using Monte Carlo to specify the point spread function for light in tissue in planar, cylindrical and spherical coordinates from a plane source, line source or point source, respectively [7], using a minimal Monte Carlo program derived from `MCML` called `mc321.c` that is listed in the Appendix of this chapter for reference and is available on the web [8]. Students have usually modified `mc321.c` to build their own specialized programs. Jacques (2003) prepared a Monte Carlo subroutine, `mcsub.c`, which allows routine calls to Monte Carlo runs

from other programs, discussed in [9], with updated versions listed on the web [10]. Ramella-Roman et al. (2005) modified `mc321.c` to simulate polarized light transport by propagating the Stokes Vector state of a photon [11], and posted the program on the web [12]. This paragraph does not offer a complete review of all contributions to Monte Carlo simulations, but provides a brief history of the programming code that is most widely used in biomedical optics and is freely available on the web.

A small note: throughout this chapter, simple math equations are mixed with programming language, and often an equal sign, "=", is used when an assignment, "←", is the appropriate symbol. The meaning will be obvious, but for those readers familiar with scientific grammar, the poor grammar is intentional.

5.2 Basic Monte Carlo Sampling

The Monte Carlo simulation of light propagation in tissue requires random selection of photon step size, scattering angle and reflection or transmission at boundaries. This is accomplished by a random number [0,1] assigned to the value of a random variable x, such as photon step size. The relationship is established through the density function $p(x)$ and distribution function $F(x)$ (see Chapter 3). Given $p(x)$, the value of the probability distribution function at a particular value x_1 of the random variable x is

$$F(x_1) = \int_0^{x_1} p(x)dx = \text{function}(x_1) \tag{5.1}$$

This $F(x_1)$ is equated with a random number, RND_1, in the interval [0,1]:

$$RND_1 = F(x_1) \tag{5.2}$$

Then Eq. (5.2) is rearranged to solve for x_1 in terms of RND_1. The resulting expression allows a series of values RND_1 to specify a series of values x_1. The histogram of x_1 values will conform to the probability density function $p(x)$. Figure 5.2 illustrates the procedure.

This process is illustrated in the following section by the random selection of photon step size, s.

5.2.1 Predicting the Step Size of a Photon

The first example is predicting the step size of a photon between interaction events (absorption or scattering) as the photon multiply scatters within a tissue. A photon takes a step (s [cm]) whose length is exponentially distributed, and depends on the value of the total attenuation coefficient $\mu_t[\text{cm}^{-1}]$ equal to $\mu_a + \mu_s$, where μ_a is

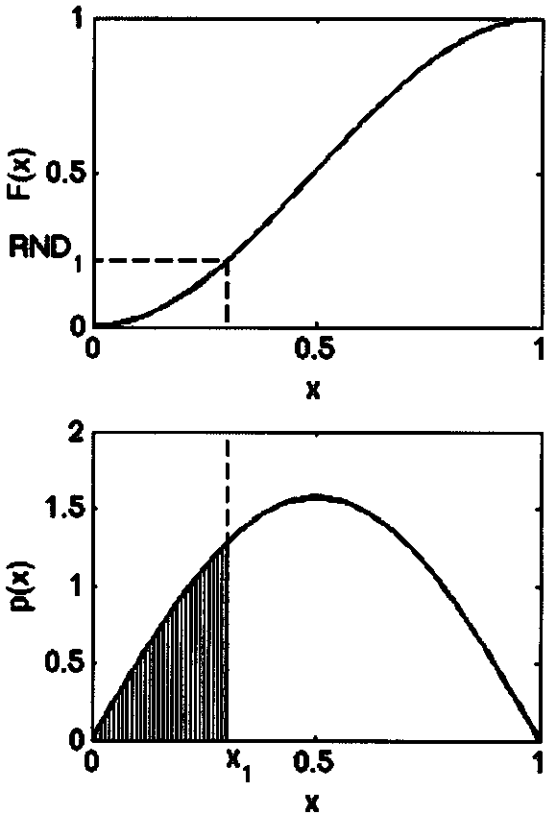

Fig. 5.2 A random number generator [0,1] selects a value RND_1 which is set equal to the distribution function $F(x)$ which then specifies the value x_1. The cross hatched area is equal to RND_1

the absorption coefficient and μ_s is the scattering coefficient. The value $1/\mu_t$ is the photon's mean free pathlength before either absorption or scattering occurs.

The first step is to specify a properly normalized probability density function that describes the probability of a particular photon step size s:

$$p(s) = \exp(-\mu_t s)/\mu_t \tag{5.3}$$

such that

$$\int_0^\infty p(s)ds = \int_0^\infty \frac{e^{-\mu_t s}}{\mu_t} ds = 1 \tag{5.4}$$

The second step is to determine the integral of $p(s)$ evaluated at a particular s_1, which defines the probability distribution function:

$$F(s_1) = \int_0^{s_1} p(s)ds = \int_0^{s_1} \frac{e^{-\mu_t s}}{\mu_t} ds = 1 - e^{-\mu_t s_1} \tag{5.5}$$

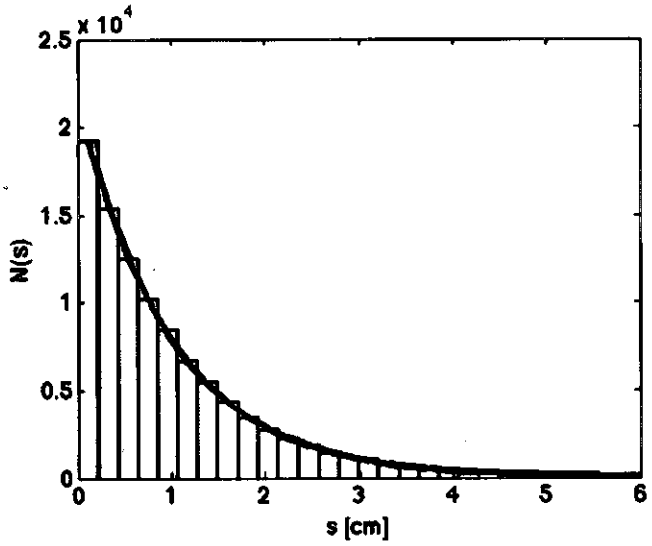

Fig. 5.3 Histogram of step sizes predicted by 1000 random numbers using Monte Carlo sampling (Eq. (5.7)). $N(s)$ is the number of step sizes per bin over a range of s value bins. The value of μ_t is 1 cm^{-1}. The bin size of the histogram is $ds_1 = 0.275$ cm. The curved line is $N(s) = 1000\exp(-\mu_t s_1)ds_1/\mu_t$

The third step is to set $F(x)$ equal to a random number RND_1.

$$RND_1 = F(s_1) = 1 - e^{-\mu_t s_1} \tag{5.6}$$

and solve for s_1:

$$s_1 = \frac{-\ln(1 - RND_1)}{\mu_t} = \frac{-\ln(RND_1)}{\mu_t} \tag{5.7}$$

The terms $(1 - RND_1)$ and (RND_1) are equal in a probabilistic sense because of the uniform distribution of the random number between [0,1]. This is the final answer. A series of random numbers RND_1 will generate a series of step sizes s_1 that follow the probability density function $p(s)$ of Eq. (5.3).

To illustrate, Fig. 5.3 shows a histogram of s_1 values specified by 1000 RND_1 values. The value of μ_t is 1 cm^{-1}. The bin size of the histogram is $ds_1 = 0.275$ cm. Also plotted is the curve $N(s) = 1000\exp(-\mu_t s_1)ds_1$.

5.2.2 *Predicting the Photon Launch Point for a Circular Flat-Field Beam*

The second example is predicting where a photon should be launched at a tissue surface so as to mimic a circular flat-field (i.e., a uniform irradiance) beam of light. Assume the beam of light is delivered perpendicular to the tissue surface. Also assume that we are working in cylindrically symmetric Cartesian coordinates. The Monte Carlo simulation will propagate the photon in 3D using x, y and z coordinates, but we will report out our results only as a function of r and z. Therefore, we need only specify the radial distance r from the center of the beam where a photon

should enter the tissue, and the photon can be launched along the x axis by letting $x = r$. Assume the beam of light has a radius a.

The number of photons per unit area should be constant. For each radial position r there is an annular ring of area $(2\pi r\, dr)$. Therefore, the number of photons to be launched at each choice of r should vary as $(2\pi r\, dr)$. The total area of the light beam is πa^2. The first step is to specify the properly normalized probability density function $p(r)$:

$$p(r) = \frac{2\pi r}{\pi a^2} = \frac{2}{a^2} r \tag{5.8a}$$

such that

$$\int_0^a p(r)dr = \int_0^a \frac{2}{a^2} r dr = 1 \tag{5.8b}$$

The second step is to integrate this $p(r)$, evaluated for a specific r_1, and equate to RND_1.

$$\int_0^{r_1} \frac{2}{a^2} r dr = \frac{r_1^2}{a^2} = RND_1 \tag{5.9}$$

The third step is to solve for r_1 in terms of RND_1:

$$r_1 = a\sqrt{RND_1} \tag{5.10}$$

Now to illustrate, Fig. 5.4 shows the histogram of 1000 r_1 values predicted by 1000 random numbers.

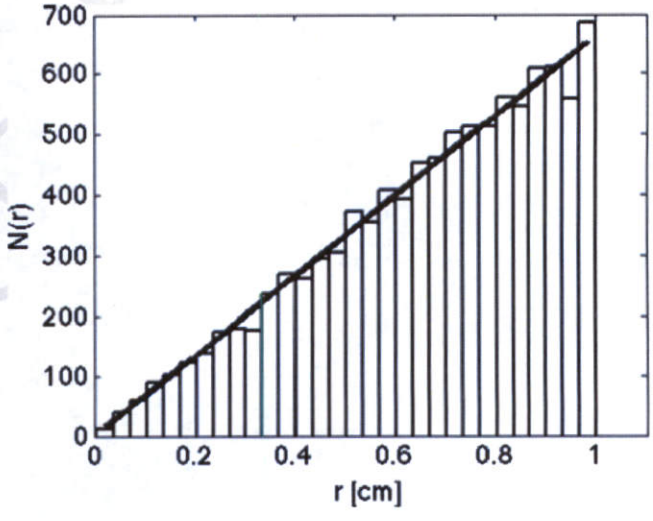

Fig. 5.4 Histogram of radial position of photon launch to achieve a circular flat-field beam of light, perpendicularly illuminating a tissue surface. The histogram is predicted by 1000 random numbers using Monte Carlo sampling (Eq. (5.10)). $N(r)$ is the number of predicted r positions per bin over a range of r value bins. The value of the beam radius a is 1 cm. The bin size of the histogram is $ds_1 =$ 0.0327 cm. The diagonal line is $N(s) = (1000\, 2r\, dr)/a^2$

5.3 The Steady-State Monte Carlo Propagation of Photons in a Tissue

This section presents the basic form of the steady-state Monte Carlo simulation of photon propagation in tissues with optical scattering and absorption properties. The term steady-state means that a stable distribution of light has been achieved, and there are no time dynamics. Section 5.4 will discuss time-resolved Monte Carlo.

It is convenient to consider the Monte Carlo simulation as yielding the fractional density matrix of incident light absorbed, A [1/cm^3], in response to one unit of delivered power or energy. The corresponding heat source matrix S [W/cm^3] or radiant energy density matrix W [J/cm^3] is obtained by scaling A by the beam power P or radiant energy Q, respectively, such that

$$S = PA \tag{5.11a}$$

or

$$W = QA \tag{5.11b}$$

where $\boldsymbol{S}(\boldsymbol{r})$ [W/cm^3] is the local density of power deposition at $r = (x,\ y,\ z)$ in response to a delivered power P [W], and $\boldsymbol{W}(\boldsymbol{r})$ [J/cm^3] is the local density of energy deposition in response to a delivered energy Q [J].

Also, the simulation keeps track of the escaping flux density out of the front surface to which light is delivered, $\boldsymbol{R}_r$ [1/cm^2], and out of the rear surface, $\boldsymbol{T}_r$ [1/cm^2]. The $\boldsymbol{A}$ [1/cm^3] is converted into the fractional transport $\boldsymbol{T}[iz][ir]$[1/cm^2] within the tissue, where iz and ir are the indices of the array T, such that the fluence rate ϕ [W/cm^2] and the fluence ψ [J/cm^2] are

$$\phi(\boldsymbol{r}) = PT(\boldsymbol{r}) \tag{5.12a}$$

$$\psi(\boldsymbol{r}) = QT(\boldsymbol{r}) \tag{5.12b}$$

The following sections will present the algorithm for implementing the Monte Carlo simulation. The overall algorithm is described by the flow diagram in Fig. 5.5.

5.3.1 The Input File

The parameters that govern a Monte Carlo simulation must be passed to the program before it can run. These parameters can be included in the programming at the beginning of a simple Monte Carlo code (as in `mc321.c`, see Appendix), or passed to a Monte Carlo subroutine from another program (as in `mcsub.c` [8, 9]) or read from an input file (as in MCML [4–6]). In Fig. 5.5, these possibilities are referred to as the "input file".

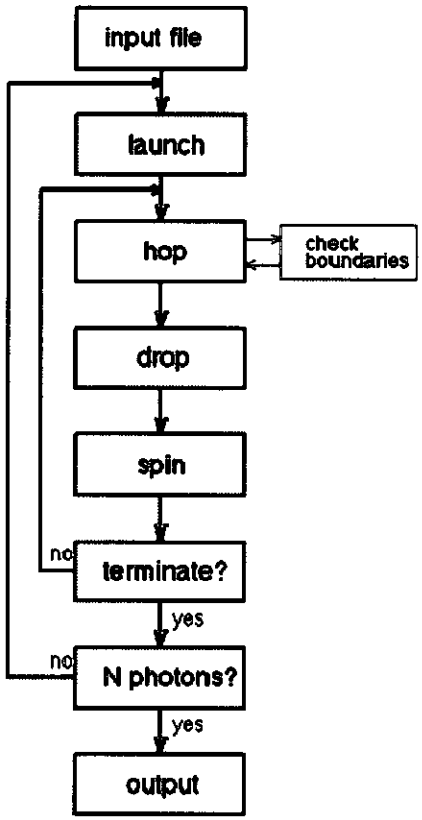

Fig. 5.5 The flow-diagram for a steady-state Monte Carlo simulation

In this simple summary of a generic Monte Carlo simulation, a simple homogeneous tissue is specified. The tissue parameters to be specified are:

$$\begin{array}{lll} \mu_a & [\mathrm{cm}^{-1}] & \text{absorption coefficient of tissue} \\ \mu_s & [\mathrm{cm}^{-1}] & \text{scattering coefficient of tissue} \\ g & [-] & \text{anisotropy of scattering of tissue} \\ n & [-] & \text{refractive index of tissue} \end{array} \tag{5.13}$$

where [–] denotes [dimensionless]. The above parameters specify an infinite optically homogeneous medium, and the simulation will track the 3D movement of a photon over an infinite spatial range. To specify boundaries, for example the front and rear surfaces of a slab of tissue, the following parameters should be specified:

$$\begin{array}{ll} D \text{ [cm]} & \text{thickness of the tissue slab} \\ n_f \text{ [}-\text{]} & \text{refractive index of external medium outside front of tissue} \\ \mathrm{n_r} \text{ [}-\text{]} & \text{refractive index of external medium outside rear of tissue} \end{array} \tag{5.14}$$

The position $z = 0$ is aligned with the front surface of the tissue slab, and the rear surface is aligned with $z = \mathrm{D}$. As the program runs, the history of the photon

is recorded by deposition of "photon weight" into spatially distributed bins, either $A(x, y, z)$ for full 3-D accumulations or $A(r, z)$ for circular symmetric laser beams. In the example of this chapter, the bins are denoted $A[iz][ir]$ [photon weight/bin]. The grid of bins for recording this A is independent of the photon propagation, which is implemented in x, y, z coordinates. Once a photon is absorbed at x, y, z, its position in this example is recorded as z, where $r = \sqrt{x^2 + y^2}$, and some "photon weight" is deposited in $A[iz][ir]$. The parameters to be specified for recording output are:

$$\begin{array}{ll} N_z\ [-] & \text{number of } z \text{ bins (depth)} \\ N_r\ [-] & \text{number of } r \text{ bins (radial position)} \\ dz\ \ [\text{cm}] & \text{size of } z \text{ bins} \\ dr\ \ [\text{cm}] & \text{size of } r \text{ bins} \end{array} \tag{5.15}$$

where the bins cover the extent of the tissue,

$$dz = \frac{D}{N_z} \tag{5.16}$$

Finally, the number of photons that will be launched must be specified,

$$N_{\text{photons}} \quad [-] \quad \text{number of photons to be launched} \tag{5.17}$$

Typically about 10^4–10^8 photons are launched, depending on how much computation time is appropriate or available. The run time will vary with the optical properties. As will be discussed in later sections, each photon will take an average step size of $1/(\mu_s + \mu_a)$ and at each step the photon weight will drop via multiplication by the fraction $\mu_s/(\mu_s + \mu_a)$ called the albedo. The weight is initially 1 upon launching the photon and drops until it reaches a threshold value, THRESHOLD, which is typically 10^{-4}. The time required to reach THRESHOLD is proportional to the number of steps, N_{steps}, taken by the photon

$$\text{THRESHOLD} = \left(\frac{\mu_s}{\mu_s + \mu_a}\right)^{N_{\text{steps}}} \tag{5.18}$$

and the number of steps required for each photon is

$$N_{\text{steps}} = \frac{\ln(\text{THRESHOLD})}{\ln\left(\frac{\mu_s}{\mu_s + \mu_a}\right)} \tag{5.19}$$

As the absorption μ_a decreases, N_{steps} increases. As the scattering μ_s increases, N_{steps} increases. The values of g and n have no effect. Therefore, the time required to run N_{photons} is

$$t = N_{\text{photons}}\, t_{\text{per.step}}\, N_{\text{steps}} \tag{5.20}$$

where $t_{per.step}$ is the time required per photon step in the simulation. One may run several example simulations with known values of THRESHOLD, μ_s, μ_a, and $N_{photons}$, and record the time required for each simulation. Then fit the data to solve for $t_{per.step}$:

$$t_{per.step} = \frac{t}{N_{photons} N_{steps}} \tag{5.21}$$

For example, if it takes 23.0 s to run 10^5 photons with optical properties $\mu_a = 1\ cm^{-1}$, $\mu_s = 100\ cm^{-1}$, using THRESHOLD $= 10^{-4}$, then $t_{per.step}$ is 249 ns. Now, if one desires a simulation that takes exactly $t = 10$ min (600 s) for properties $\mu_a = 0.4\ cm^{-1}$ and $\mu_s = 65\ cm^{-1}$, then the choice of $N_{photons}$ is

$$\begin{aligned} N_{photons} &= \frac{t}{t_{per.step} N_{steps}} = \frac{t}{t_{per.step}} \frac{\ln\left(\frac{\mu_s}{\mu_s + \mu_a}\right)}{\ln(\text{THRESHOLD})} \\ &= \frac{600s}{249 \times 10^{-9}} \frac{\ln\left(\frac{65}{65 + 0.4}\right)}{\ln\left(10^{-4}\right)} = 1.60 \times 10^6 \end{aligned} \tag{5.22}$$

So one would launch 1.60 million photons for a 10-min simulation using the above optical properties. In this manner, the choice of $N_{photons}$ for a given simulation can be made on the basis of available time, regardless of the optical properties in the simulation.

Now the Monte Carlo simulation can proceed to run.

5.3.2 *Launching Photons*

Photons are launched with an initial weight (w) set to 1.0. As the photon propagates, this weight will be decremented as "photon weight" is "dropped" into the bins $A[iz][ir]$, where iz and ir are index pointers to the bins for the depth z and radial position r, respectively. A large number of photons will be launched ($N_{photons}$) and later at the end of the program all the weight deposited in $A[iz][ir]$, as well as any weight that escaped the tissue as transmission or reflectance, will be normalized by $N_{photons}$, such that the final results are reported as the fraction of all delivered light that is either absorbed, reflected or transmitted.

The position of the photon is specified in Cartesian coordinates, (x,y,z), and all the propagation is done in full 3D. The recording of $A[iz][ir]$ is in cylindrical coordinates, but one could implement $A[iz][ix][iy]$ if desired. If one is using 100×100 bins for $A[iz][ir]$, one needs only to fill 10^4 bins with photon weight. If one is using $100 \times 100 \times 100$ bins for $A[iz][ix][iy]$, one needs to fill 10^6 bins with photon weight. Since the signal-to-noise in any bin is roughly proportional to $\sqrt{A}/A$, it takes many more photons to attain good signal-to-noise filling $A[iz][ix][iy]$ than filling.

$A[iz][ir]$. This is why cylindrical coordinates are often chosen for use. But there is no reason not to record data as $A[iz][ix][iy]$. The choice of recording does not affect the photon propagation.

The trajectory of the photon will be specified by the trajectory cosines (ux, uy, uz):

$$\begin{aligned} ux &= \cos\theta \cos\varphi \\ uy &= \cos\theta \sin\varphi \\ uz &= \cos\theta \end{aligned} \tag{5.23}$$

where θ is the angle that the trajectory makes with respect to the z axis, and φ is the angle that the trajectory makes with the x axis.

So let's launch some photons.

5.3.2.1 Collimated Launch

In this example, photons are launched perpendicular to the tissue and enter the tissue exactly at the origin $(x, y, z) = (0, 0, 0)$. The photon position and trajectory are

$$\begin{aligned} x &= 0 \\ y &= 0 \\ z &= 0 \end{aligned} \tag{5.24a}$$

and

$$\begin{aligned} ux &= 0 \\ uy &= 0 \\ uz &= 1 \end{aligned} \tag{5.24b}$$

The photon is directed straight downward, so $uz = 1$. The values of ux and uy are zero because no component of the trajectory is directed in the x or y directions.

The case of collimated launch as a flat-field beam, i.e., uniform irradiance, where the beam has a radius a, was discussed in Section 5.2.2. Photons can be placed along the x axis according to Eq. (5.10) with $y = 0$, $z = 0$.

5.3.2.2 Isotropic Point Source

To launch a photon isotropically, i.e., with no preferential direction, at a position located within the tissue, for example at $x = 0$, $y = 0$, and $z = 0.1$ cm, the launch specification is:

$$\begin{aligned} x &= 0 \\ y &= 0 \\ z &= 0.1 \end{aligned} \tag{5.25a}$$

and

$$\cos\theta = 2RND - 1$$
$$\sin\theta = \sqrt{(1-\cos^2\theta)}$$
$$\varphi = 2\pi\, RND \tag{5.25b}$$

if ($\varphi < \pi$)

$$\sin\varphi = \sqrt{(1-\cos^2\varphi)}$$

else

$$\sin\varphi = -\sqrt{(1-\cos^2\varphi)}$$

such that

$$\begin{aligned} ux &= \sin\theta\cos\varphi \\ uy &= \sin\theta\sin\varphi \\ uz &= \cos\theta \end{aligned} \tag{5.25c}$$

The z position is 0.1 cm. The trajectory (ux,uy,uz) is randomly selected such that

$$\sqrt{ux^2 + uy^2 + uz^2} = 1 \tag{5.26}$$

It is necessary for the total length of the trajectory vector to be unity.

The photon is launched at $x = 0$ and $y = 0$ because we wish to retain cylindrical symmetry. If one launched at any other x,y position the cylindrical symmetry would be broken. Then the results would have to be recorded as $A[iz][ix][iy]$. The results recorded as $A[iz][ir]$ would respond as if photons were launched in a ring of radius $\sqrt{x^2+y^2}$.

5.3.2.3 Collimated Gaussian Beam

Consider a collimated Gaussian beam that irradiates a tissue, which has a $1/e$ radius of b. The probability density function for the radial position of launch is

$$p(r) = \frac{e^{-(r/b)^2} 2\pi r}{\pi b^2} \tag{5.27}$$

which can be sampled using the Monte Carlo sampling method by

$$r = b\sqrt{-\ln(RND)} \tag{5.28}$$

To launch a Gaussian beam when using cylindrically symmetric results, one can launch the photon at $x = r$. The launch parameters are:

$$\begin{aligned} x &= b\sqrt{-\ln(RND)} \\ y &= 0 \\ z &= 0 \end{aligned} \tag{5.29a}$$

and

$$\begin{aligned} ux &= 0 \\ uy &= 0 \\ uz &= 1 \end{aligned} \tag{5.29b}$$

5.3.2.4 Focused Gaussian Beam

Consider the same Gaussian beam but focused to a focal point within the tissue at depth z_{focus}. Let the focus have a radial Gaussian distribution with a $1/e$ radius of w. Then the launch parameters are calculated:

$$\begin{aligned} x &= w\sqrt{-\ln(RND)} \\ y &= 0 \\ z &= 0 \end{aligned} \tag{5.30a}$$

and

$$\begin{aligned} x_{\text{focus}} &= w\sqrt{-\ln(RND)}\,\text{sign}(2RND - 1) \\ \text{temp} &= \sqrt{((x - x_{\text{focus}})^2 + z_{\text{focus}}{}^2)} \\ \sin\theta &= -(x - x_{\text{focus}})/\text{temp} \\ \cos\theta &= z_{\text{focus}}/\text{temp} \end{aligned} \tag{5.30b}$$

such that

$$\begin{aligned} ux &= \sin\theta \\ uy &= 0 \\ uz &= \cos\theta \end{aligned} \tag{5.30c}$$

There is a tendency in Monte Carlo simulations using cylindrical coordinates to not get sufficient photons in the bins along the z axis, because the size of each bin is $(2\pi r\ dr\,dz)$, so the size of bins near $r = 0$ is very small, and the likelihood of photon deposition in such a small bin is quite low. During this focused Gaussian launch, one should launch toward both $+x_{\text{focus}}$ and $-x_{\text{focus}}$ positions, which forces the photons to cross the z axis. Doing this causes the bins along the central axis to not be so neglected, and is accomplished by the extra term sign (2 *RND*–1) in the expression for x_{focus}. The function sign () has a value of +1 or −1.

5.3.3 Hop

Now the photon is launched along a trajectory, and takes a step along this trajectory. The standard procedure for taking the step is described as the "Standard Hop" in Section 5.3.3.1. But if the tissue has a front and/or rear boundary, then a second step to "Check boundaries" is taken, as described in Section 5.3.3.2, and if the photon is attempting to escape the tissue, a procedure is used to decide whether the photon escapes or is reflected back into the tissue.

5.3.3.1 Standard Hop

The step size of the photon's step (or hop) must be determined. The step size is calculated:

$$s = -\ln(RND)/\mu_t \tag{5.31}$$

as was described in Section 3.2.1. Now the position of the photon is updated:

$$\begin{aligned} x &= x + s\,ux \\ y &= y + s\,uy \\ z &= z + s\,uz \end{aligned} \tag{5.32}$$

5.3.3.2 Check Boundaries

As part of the Hop step, there is a side box labeled "check boundaries" in Fig. 5.5, which is used when there are front and surface boundaries in the problem. This boundary check is denoted by a side box to emphasize that it is part of the Hop step.

As the photon moves toward the front surface of the tissue and attempts to cross the boundary to escape the tissue, there is a possibility that the photon will be reflected by the surface boundary where the air/tissue interface (or *external medium*/tissue interface) presents a mismatch in refractive indices ($n_f \neq n$ or $n_r \neq n$, for front and rear boundaries, respectively). The method chosen here for handling the boundary is to let a fraction of the photon weight escape the tissue as observable reflectance, and let the remaining fraction of photon weight reflect back into the tissue and continue to propagate.

After taking the step, the position of the photon is checked to see if the photon is still within the tissue or has escaped the tissue:

$$\begin{aligned} &\text{If } z < 0 \\ &\qquad \text{photon is trying to escape} \\ &\text{else} \\ &\qquad \text{photon still within tissue} \end{aligned} \tag{5.33}$$

If the photon is trying to escape, then a partial step is taken along the escaping trajectory that will just reach the boundary surface. The size of the partial step s_1 is

$$s_1 = \text{abs}(z/uz) \tag{5.34}$$

where abs() denotes the absolute value function. First, the photon retracts the full step it took in Hop, which led to escape,

$$\begin{aligned} x &= x - s\,ux \\ y &= y - s\,uy \\ z &= z - s\,uz \end{aligned} \tag{5.35a}$$

and then the photon takes the partial step s_1,

$$\begin{aligned} x &= x + s_1\,ux \\ y &= y + s_1\,uy \\ z &= z + s_1\,uz \end{aligned} \tag{5.35b}$$

Now the photon is located at the boundary surface. Next, a decision is made about how much of the photon weight escapes or how much is reflected.

The probability of reflectance at the boundary is a function of the angle of incidence, encoded as the value uz, and the refractive indices n_1 and n, where n_1 denotes the external medium and equals either n_f or n_r for the front and rear boundaries, respectively. Consider a photon striking the front surface. The internal reflectance, R_i, is calculated using the Fresnel reflection equation:

$$\begin{aligned} R_i = {} & \frac{\sin\theta^2\,(\sin\theta_1\cos\theta_2 - \cos\theta_1\sin\theta_2)^2}{2} \\ & \times \frac{\left((\cos\theta_1\cos\theta_2 + \sin\theta_1\sin\theta_2)^2 + (\cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2)^2\right)}{\left((\sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2)^2\,(\cos\theta_1\cos\theta_2 + \sin\theta_1\sin\theta_2)^2\right)} \end{aligned} \tag{5.36}$$

where

n_1	refractive index of incident medium	$= n_{\text{tissue}}$
n_2	refractive index of transmitted medium	$= n_f$
$\cos\theta_1$	incident trajectory	= uz
$\sin\theta_1$	incident trajectory	$= (1 - uz^2)^{1/2}$
$\sin\theta_2$	transmitted trajectory	$= \sin\theta_1(n_1/n_2)$
$\cos\theta_2$	transmitted trajectory	$= (1 - \sin\theta_1^2)^{1/2}$

Once the reflectance R_i is computed, a fraction $(1 - R_i)$ of the current photon weight is allowed to escape and the remaining fraction of weight is reflected back into the tissue to continue propagating. The escape is recorded by placing its remaining weight in the reflectance array bin, $R_r[ir]$,

$$R_r = R_r[ir] + (1 - R_i)w \tag{5.37}$$

where the radial position of escape is r:

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ ir &= \text{round}(r/dr) + 1 \end{aligned} \tag{5.38}$$

The function round(r/dr) denotes taking the integer value of the value r/dr, rounding down, such that a photon somewhere within the location of one bin will be assigned to that particular bin. (Note: In this summary, ir extends from 1 to N_z, and ir is not allowed to equal zero. Sometimes, programs let ir extend from 0 to N_z–1. Not mentioned here but important in programming is that a check should be made that ir does not exceed Nr, the allocated size of $Rr[ir]$.)

The remaining fraction of the photon weight is reflected by the boundary,

$$w = R_i w \tag{5.39}$$

and the trajectory with respect to the z axis is reversed,

$$uz = -uz \tag{5.40}$$

The ux and uy components of the trajectory are not changed. Then the photon position is updated by taking the remaining portion of the original step, which equals $s - s_1$:

$$\begin{aligned} y &= (s - s_1)ux \\ y &= (s - s_1)uy \\ z &= (s - s_1)uz \end{aligned} \tag{5.41}$$

Some of the photon weight has escaped the boundary and some has been reflected by the boundary back into the tissue to continue propagating.

The procedure for testing if the photon is attempting to escape the tissue through its rear surface boundary and contributing to $T[iz][ir]$, and for modifying the position and trajectory accordingly, is very similar to the above procedure for the front surface, and is not outlined here. If one wished to place other boundaries in the problem, like a lateral cylindrical boundary as if the tissue were held within a tube (e.g., a glass test tube, or a pipe), this "check boundaries" box is the proper place within the program to implement such special boundaries.

5.3.4 Drop

Arriving at its new position, the photon must interact with the tissue. Upon interaction, a fraction μ_a/μ_t of the photon's weight is absorbed and the remaining μ_s/μ_t fraction of the photon's weight is scattered and continues to propagate. The absorbed fraction is placed in the bin that encloses the current photon position. This process is summarized by the following calculation steps:

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ ir &= \text{round}(r/dr) + 1 \\ iz &= \text{round}(z/dz) + 1 \\ A[iz][ir] &= A[iz][ir] + w(\mu_a/\mu_t) \\ w &= w(\mu_s/\mu_t) \end{aligned} \tag{5.42}$$

where ir and iz are index values and dr and dz are bin width and depth, respectively. This finishes the absorption event.

Note that ir and iz should not be allowed to exceed N_r and N_z, respectively, lest one exceed the allocated size of $A[iz][ir]$. Always check this. Often, the last bin is used as an overflow bin and any photon weight that is deposited outside the array is simply accumulated in the last iz,ir bin. At the end of the simulation, the values of weigth in $Rr[ir]$, $Tr[ir]$ and $A[iz][ir]$ are the fraction of total delivered photon weight. The sum of $Rr[ir]$, $Tr[ir]$ and $A[iz][ir]$ over all bins divided by N_{photons} will equal unity. In Section 5.3.7, the proper normalization of these fractional weights by the size of the bins will convert Rr and Tr into the fraction of delivered power or energy escaping per unit surface area and convert A into the fraction of delivered power or energy deposited per unit volume. But the values of Rr, Tr and A determined from the overflow bins are meaningless after normalization.

Next, is the scattering event.

5.3.5 Spin

The photon is scattered into a new trajectory according to some scattering function. The two angles of scatter are θ and φ , the deflection and azimuthal scattering angles, respectively. This section describes how to calculate the new trajectory after sampling the probabilities for θ and φ.

The most commonly used function for the deflection angle θ is the Henyey-Greenstein (HG) function (1941), which was proposed for describing the scattering of light from distant galaxies by galactic dust. The original paper does not offer any explanation for the function, but simply asserts its use. But the HG function is actually very interesting. The function is here expressed as a function of the angle of deflection, θ, and the anisotropy of scattering, g. Using the definitions in Chapter 3 (Section 3.4.3.1) where the probability density functions for $p(\varphi)$ and $p(\theta)$ are considered independently:

$$p(\theta) = \frac{1}{2} \frac{1 - g^2}{\left(1 + g^2 - 2g\cos\theta\right)^{3/2}} \tag{5.43}$$

which has the properties that

$$\int_0^{\pi} p(\theta)\, 2\pi \sin\theta \, d\theta = 1 \tag{5.44a}$$

and

$$\int_0^{\pi} p(\theta)\cos\theta\, 2\pi\, \sin\theta d\theta = g \tag{5.44b}$$

The last equation is the definition of g. Hence, the HG function is an identity with respect to the definition of g. If you choose a value g to define $p(\theta)$ using Eq. (5.43), the definition of g in Eq. (5.44b) will yield exactly g.

The Monte Carlo sampling of the HG function is specified by the following sequence of calculations:

$$\cos(\theta) = \frac{1 + g^2 - \left(\dfrac{1 - g^2}{(1 - g + 2gRND)^{3/2}}\right)^2}{2g} \tag{5.45}$$

If g is 0, then use $\cos\theta = 2\,RND - 1$. If g is 1.0, then simply let $\cos\theta = 1.0$. Otherwise, use the Monte Carlo sampling in Eq. (5.45).

The azimuthal angle is calculated:

$$\varphi = 2\pi RND \tag{5.46}$$

To update the trajectory based on the values of $\cos\theta$ and φ specified using random numbers, use the following calculations:

$$\begin{aligned}
\sin\theta &= \sqrt{(1 - \cos\theta^2)} \\
\text{temp} &= \sqrt{(1 - uz^2)} \\
uxx &= \sin\theta\,(ux\,uz\,\cos\varphi - uy\sin\varphi)/\text{temp} + ux\cos\theta \\
uyy &= \sin\theta\,(uy\,uz\,\cos\varphi - ux\sin\varphi)/\text{temp} + uy\cos\theta \\
uzz &= -\sin\theta\,\cos\varphi\,\text{temp} + uz\cos\theta
\end{aligned} \tag{5.47a}$$

If the trajectory is extremely close to alignment with the z axis, *i.e.* nearly $(uz,\ uy,\ uz) = (0, 0, \pm 1)$, do not use Eq. (5.24) above, but instead use:

$$\begin{aligned}
&uxx = \sin\theta\cos\varphi \\
&uyy = \sin\theta\sin\varphi \\
&\text{if } uz \geq 0 \qquad uzz = \cos\theta \\
&\text{else} \qquad uzz = -\cos\theta
\end{aligned} \tag{5.47b}$$

Finally, one updates the trajectory:

$$ux = uxx$$
$$uy = uyy$$
$$uz = uzz$$

The photon is now oriented along a new trajectory, and ready to take a new step s (see Fig. 5.5).

There are alternative scattering functions. Mie theory is an important scattering function to consider. Perhaps an experiment has yielded a particular scattering function, and one wishes to run a simulation using this function. This chapter will not discuss these alternatives, but as long as the criteria of Eqs. (5.44a, b) are followed, most any scattering function for the deflection angle θ can be used. Sometimes the scattering function does not lend itself to a solution of $\cos\theta$ in terms of a random number, as in Eq. (5.3). Also, sometimes one wishes to consider an azimuthal scattering angle φ that depends on the deflection angle θ, as in Mie scattering of polarized light. In such cases, the "rejection method" is a useful approach [10].

5.3.6 *Terminate?*

The photon will continue propagating as its weight becomes progressively smaller. How can one stop the photon yet properly conserve energy? The "Roulette Method" is used to terminate the photon. A threshold value (THRESHOLD) is chosen, typically 10^{-4}. When the photon's weight drops below this threshold value, the roulette procedure is employed. A random number (*RND*) is generated; and if this random number is less than a small fraction called CHANCE, typically 0.10, then the photon weight is increased by dividing w by CHANCE. For CHANCE = 0.10, this would be a 10-fold increase in w. Otherwise, the photon is terminated. Consequently, 9 out of 10 times the photon is terminated, but 1 out of 10 times the photons weight is increased 10-fold and the photon continues to propagate. The result is that photons are usually terminated, but energy is conserved by the occasional surviving photon being given extra weight. Since millions of photons are run, the statistically averaged result is correct. In summary, the roulette method is implemented by the following:

```
if(w < THRESHOLD)
  if(RND <= CHANCE)
        w = w/CHANCE                    (5.49)
  else
    terminate the photon
```

Once the photon is terminated, a new photon can be launched. One checks to see if the total number of photons has already reached the maximum number

(N_{photons}) requested by the input file. If not, then a new photon is launched. If yes, the simulation is complete, and it is time to prepare the results for output.

5.3.7 Normalizing Results for Output

Now, all the photons (N_{photons}) have been run, and it is time to save the output of the simulation. The data has been stored in the array A, as $A[iz][ir]$ or perhaps $A[iz][ix][iy]$, in units of [photon weight/bin]. Either way, the key parameter is the volume V [cm^3] associated with each bin. For $A[iz][ir]$, the volumes vary with the value of ir,

$$V = 2\pi(ir - 0.5)dr^2 dz \tag{5.50a}$$

and for $A[iz][ix][iy]$, the volumes are all equal,

$$V = dx\, dy\, dz \tag{5.50b}$$

Then the values A [photon weight/bin] are normalized by the appropriate V and by the value N_{photons} to yield the absorbed fraction, A [1/cm^3], for each pixel:

$$A[ir, iz] = \frac{A\,[ir, iz]}{V\,[ir, iz]\,N_{\text{photons}}} \tag{5.51}$$

The fractional transport, T [1/cm^2], is then calculated as

$$T = \frac{A}{\mu_a} \tag{5.52}$$

Recall from Eqs. (5.12a, b) that fluence rate ϕ [W/cm^2] equals the incident power P [W] times T, $\phi = PT$, and the fluence ψ [J/cm^2] equals the incident energy Q [J] times T, $\psi = QT$.

Keep in mind that the values of A, V and μ_a are specific to each bin, when calculating $T[iz][ir]$ or $T[iz][ix][iy]$. In this summary of a simple implementation of the Monte Carlo method, the μ_a was assumed to be uniform, as well as the scattering properties, so every bin had the same value of μ_a. But in MCML, for example, bins at different depths can have different values of μ_a.

The light fluxes that have escaped at the front and rear surface boundaries are similarly normalized, but in this case the surface area AREA rather than the volume V is used. The value of AREA$[ir]$ is $2\pi(ir - 0.5)dr^2$. The array of escaping photons, $R_r[ir]$ [photon weight/bin], is converted to the fractional escaping flux density, $R_r[ir][1/\text{cm}^2]$, by the expression:

$$R_r = \frac{R_r}{AREA\; N_{\text{photons}}} \tag{5.53}$$

This completes the discussion of the steady-state Monte Carlo simulation. The output is the fractional transport, $T\,[iz][ir][1/\text{cm}^2]$, and the fractional escaping flux

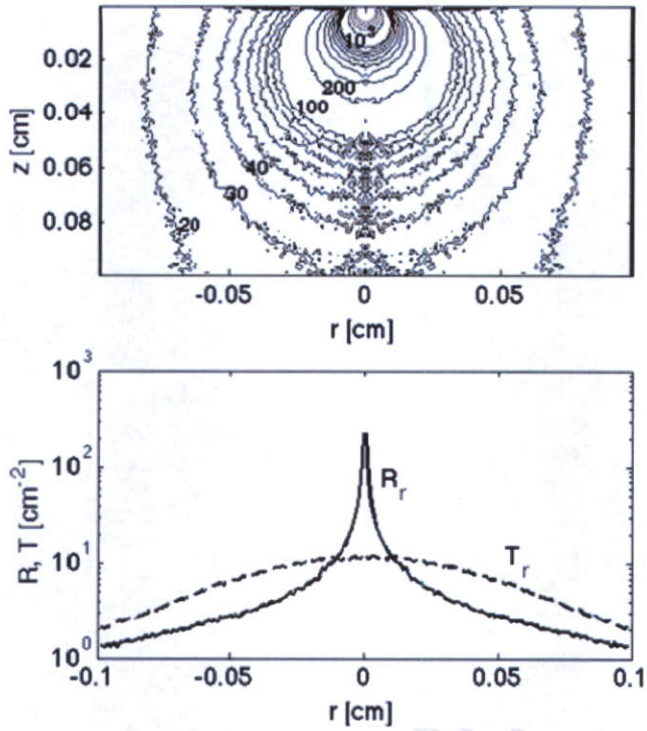

Fig. 5.6 Example steady-state Monte Carlo simulation. The optical properties were $\mu_a = 1\,\text{cm}^{-1}$, $\mu_s = 100\,\text{cm}^{-1}$, $g = 0.90$, $n = 1.4$, for a 1-mm-thick tissue slab with air/tissue boundaries at front and rear surfaces. There were 10^6 photons propagated during a run time of 3 min 49 s on a laptop computer (2 GHz processor). (**a**) Iso-T contours for 20, 30, 40, 100, 200 and 10^3 1/cm^2], T denotes the fractional transport. The noise is evident along the central axis near $r = 0$ because the bins are smaller and collect fewer photons. Running more photons improves the signal-to-noise. (**b**) The reflectance (R_r) and transmission (T_r) as flux densities of escape [1/cm^2] versus radial position r at front and rear surfaces, respectively

densities, $R_r[ir][1/\text{cm}^2]$ and $T_r[ir][1/\text{cm}^2]$. This discussion ends with one example calculation shown in Fig. 5.6, showing T, R_r and T_r for a 1-mm-thick slab of tissue, with light delivered as a pencil beam of collimated light at the origin: $(x, y, z) = (0, 0, 0), (ux, uy, uz) = (0, 0, 1)$.

5.4 Time Resolved Monte Carlo Propagation

Time-resolved Monte Carlo simulation is almost identical to the steady-state simulation discussed above, except for some minor changes. There are actually several ways to implement time-resolved Monte Carlo, and this section shows one approach that illustrates the basic idea.

In this example, the photon is allowed to propagate with no absorption, and the total path of the photon is accumulated after each step s throughout the propagation:

$$L = L + s \tag{5.54}$$

Since there is no absorption, $\mu_a = 0$ and $\mu_t = \mu_s$. Since the speed of light is c, the time duration of the propagation is:

$$t = \frac{L}{c} \tag{5.55}$$

where c is the speed of light in the tissue, $c = c_o/n$ where c_o is the speed of light in vacuo. The time for a photon to escape out the front surface is determined from its total pathlength L at the moment of escape divided by c.

Assume that one is interested in a set of 10 time-points, $t[it]$, where $it = 1 - 10$ is an index that refers to the desired time-point. Let the first time-point, $t[1]$, be 100 ps. Propagation is allowed to continue until the next photon step causes L to exceed the pathlength corresponding to 100 ps,

$$L + s > t[it]/c \tag{5.56}$$

where $t[1] = 100$ ps for this example. At this point, a partial step size, s_1, is taken,

$$s_1 = t[it]/c - L \tag{5.57}$$

The photon is now located exactly at the time-point of 100 ps. The current photon weight w is deposited in the bin $A[iz][ir][it]$, where $[it]$ selects a full 2-D $A[iz][ir]$ array associated with each particular time point. The entire photon weight ($w = 1$) is placed in the local bin, but the weight of the photon is not decremented. The bin $A[iz][ir][it]$ takes a snapshot of the photon's location and weight, but does not affect the photon. There is no absorption. The photon is allowed to continue propagating. The remainder of the step size, $s - s_1$, is taken by the photon. The photon continues to propagate as usual, until a next step causes L to exceed the 2nd time point, $t[2]$. The process of taking a partial step, depositing w into $A[ir][iz][it]$ without changing the w of the photon, completing the step, and resuming propagation is executed. The process continues until L passes the last desired time point, then the photon is terminated and a second photon is launched.

During propagation, when a photon strikes one of the boundaries, a fraction $(1 - R_i)$ of the current photon weight will escape, and the photon weight will decrement. The new photon weight, $R_i w$, will internally reflect and continue propagating. The escaping photon weight will be placed in the bin $R_r[ir][jt]$,

$$R_r[ir][ji] = R_r[ir][ji] + (1 - R_i)w \tag{5.58}$$

where jt is an index that encodes the time of escape, and may be divided into equal time steps, dt [s], that cover the time duration of interest. The current jt is computed:

$$jt = \text{round}(t/dt) + 1 \tag{5.59}$$

As an example of evenly divided time bins, if the time duration of interest was from 10 ps to 1 ns, then dt would be 10 ps and jt would extend from 1 to 100. An alternative approach is to have progressively larger dt bin sizes, so the time base can extend from very short times to very long times, which is not discussed here.

Finally, after N_{photons} have been launched and terminated, it is time to normalize the bins A and R_r for final output. For each time point, $[it]$, conservation of energy in terms of photon weight is summarized:

$$\frac{1}{N_{\text{photons}}}\left(\sum_{iz}\sum_{ir} A[iz][ir][it] + \sum_{jt=1}^{it} R_r[ir][jt]\right) = 1 \tag{5.60}$$

The above calculation is not routinely needed, but is only a check that energy is conserved. The final normalization is

$$T = c\frac{A}{V\,N_{\text{photons}}} \tag{5.61}$$

which has units of $[1/(\text{cm}^2\ \text{s})]$. The time-resolved fluence rate, $[\text{W/cm}^2]$ in response to an impulse of energy Q [J] delivered at time zero is:

$$\phi\,(t) = QT \tag{5.62}$$

The escaping flux density, $R_r[1/(\text{cm}^2\ \text{s})]$, is normalized:

$$R_r = \frac{R_r}{\text{AREA}\ N_{\text{photons}}\ dt} \tag{5.63}$$

Both $A[iz][ir][it]$ and $R_r[ir][it]$ will have good signal-to-noise in regions near the source where most photons spend their time, and poor signal-to-noise in regions far from the source. It is difficult to get good results far from the source even when a large number of photons are launched. Usually, time-resolved Monte Carlo simulation is used to specify results close to a source, and time-resolved diffusion theory is used to specify results at far distances from the source.

Now that the Monte Carlo simulation is completed, absorption can be added to the problem. The attenuation due to absorption is specified by Beer's law which says that photon survival equals $\exp(-\mu_a ct)$, since the ct equals the photon's total pathlength L at any particular time t. Therefore,

$$\phi(t) = QT(t)e^{-\mu_a ct} \tag{5.64}$$

and

$$R_r(t) = \frac{R_r(t)}{\text{AREA}\ N_{\text{photons}}\ dt}e^{-\mu_a ct} \tag{5.65}$$

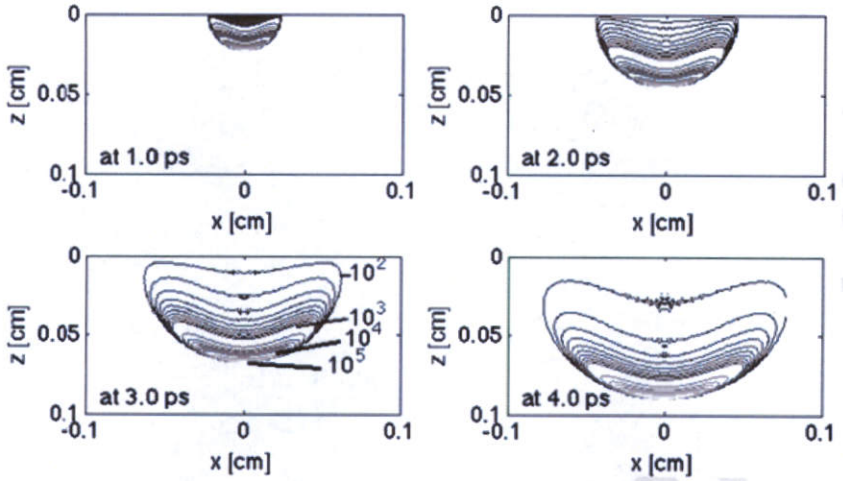

Fig. 5.7 Time-resolved propagation of a collimated laser impulse launched at $r = 0$, $z = 0$ into a 1-mm thickness of a standard tissue. The thickness represents ~10 optical depths for optical properties of $\mu_a = 1\ \text{cm}^{-1}$, $\mu_s = 100\ \text{cm}^{-1}$, $g = 0.90$, $n = 1.4$. The four time points are 1, 2, 3 and 4 ps. The maps are iso-T contours, where T is the time-resolved fractional transport [1/(cm^2 s)]. The influence of μ_a is minor, for example, $\exp(-\mu_a ct)$ is only 0.92 when $t = 4$ ps

In this way, any value of μ_a can be introduced to learn its influence on the time-resolved distribution of $T(t)$ or $R_r(t)$.

To illustrate time-resolved Monte Carlo, an example of snapshots of $T[iz][ir]$ at 4 time-points is presented in Fig. 5.7, which illustrates the early movement of an incident laser pulse into a tissue. The narrow beam laser pulse broadens with time due to scattering.

Figure 5.8 shows an optical fiber embedded within a tissue and terminating at a depth of 500 μm, delivering light toward the surface. As the light reaches the surface, the time-resolved escape of fractional flux density, $R_r(t, r)[1/(\text{cm}^2\ \text{s})]$ is shown.

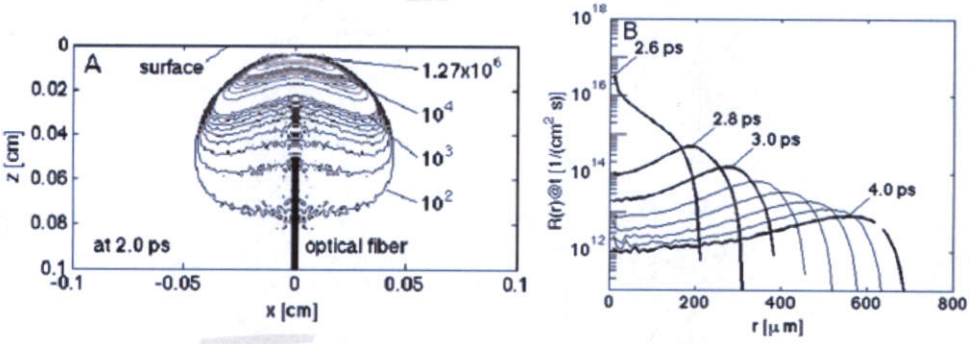

Fig. 5.8 Time-resolved escape of a laser pulse from within tissue. The optical properties were $\mu_s = 100\ \text{cm}^{-1}$, $g = 0.90$, $n_{\text{tissue}} = 1.33$. The impulse is delivered toward the surface from an optical fiber terminated at a depth of 500 μm within the tissue and pointed toward the surface. (**a**) Impulse at time 2 ps, shown as $T\ [1/(\text{cm}^2\text{s})]$, maximum is 1.27×10^6. (**b**) The escaping fractional flux density versus radial position at different times, $R_r(t, r)[1/(\text{cm}^2\ \text{s})]$

5.5 Converting Time-Resolved Results to Frequency-Domain

When the intensity of a light source is modulated at very high frequencies, the ability of the frequency of modulation to transport to some position of observation is described as *frequency domain* light transport. Time-resolved information generated by a time-resolved Monte Carlo simulation can be converted by Fourier Transform into frequency domain information.

Consider the time-resolved escape of fractional flux density, $R_r(t, r)[1/(\text{cm}^2\ \text{s})]$, when light is delivered as a collimated impulse to position $(r, z) = (0, 0)$ on a tissue surface. The light enters the tissue, but due to scattering begins to escape from the tissue after some delay. This time-resolved $R_r(t, r)$ is shown in Fig. 5.9a, for a typical tissue. As the position of observation moves from 1 mm to 6 mm distance from the source, the time delay before onset of escaping flux increases, and the amount of light escaping decreases.

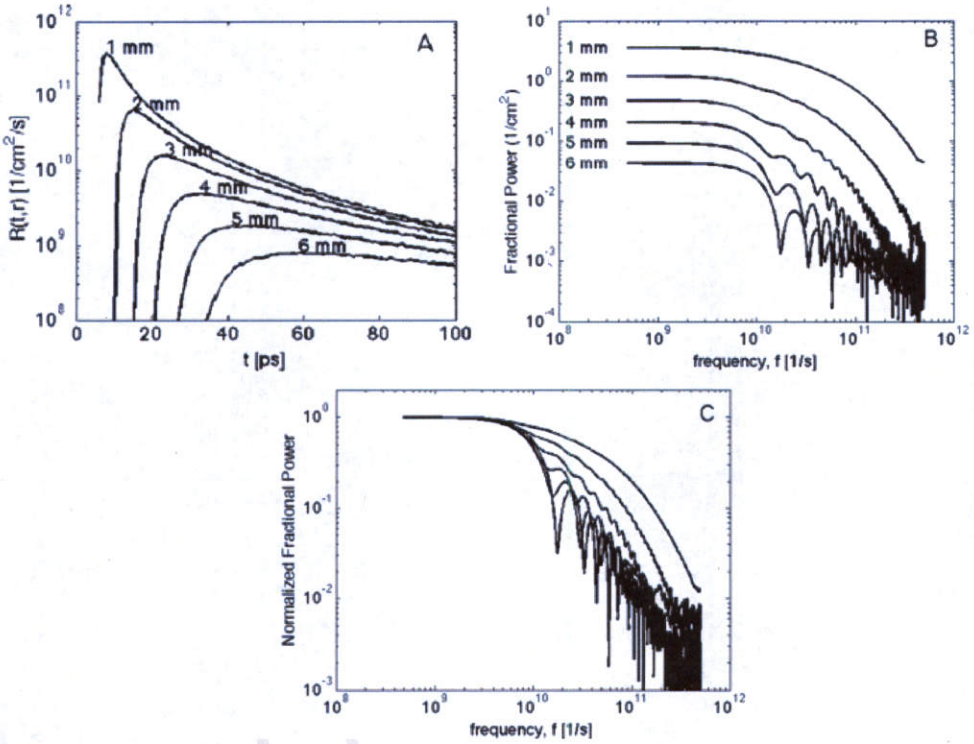

Fig. 5.9 Frequency domain light transport, showing how light modulated at different frequencies and delivered as a collimated beam to position $(r, z) = (0, 0)$, will escape from the tissue as a function of radial position. (**a**) The time-resolved escape of light from the tissue, $R_r[1/(\text{cm}^2\ \text{s})]$ for r = 1 − 6 mm. (**b**) The fractional power spectrum [1/cm^2] versus frequency f [1/s]. The optical properties of the tissue were $\mu_a = 1\text{cm}^{-1}$, $\mu_s = 100\,\text{cm}^{-1}$, $g = 0.90$, $n = 1.4$, and 10^7 photons were launched during a 35-min simulation. The data at the higher frequencies for the most distant radial positions were based on low photon weights and show artifactual oscillations. (**c**) Normalized fractional power versus frequency

The corresponding frequency domain information is obtained by using a Fast Fourier Transform (*fft*) to convert the time-resolved $R_r(t, r)$ escaping at a particular radial position r, denoted as $R_r(t)$, into the frequency domain:

$$F(f) = \text{abs}(fft\,(R_r(t)dt)) \tag{5.66}$$

where dt is the time-step of the time-resolved data. The absolute value converts the imaginary values generated by the *fft*() into real values that correspond to the power spectrum, expressed as the fractional power F [$1/cm^2$]. In other words, if the source was modulated at a frequency f [1/s], the function F would specify the fractional escaping flux, $R_r[1/cm^2]$, that was still modulated at frequency f. Figure 5.9b shows this power spectrum. The limiting values toward low frequencies of modulation correspond to the steady-state reflectance $R_r(r)[1/cm^2]$, which is why the factor dt was included in the above equation.

To illustrate the above equation more specifically, the equivalent programming code written in MATLAB^{TM} notation is listed:

```
mua = 1;     % absorption coefficient [cm^-1]
mus = 100;   % scattering coefficient [cm^-1]
g   = 0.90; % anisotropy of scattering [dimensionless]
n   = 1.5;   % refractive index [dimensionless]
dt = 1e-12; % time step of time-resolved data [s], in this case dt = 1 ps.
t = (1:100)'*dt;        % time base of Monte Carlo data [s], up to 100 ps
r = (1:100)'/100*1.0;  % radial position of Monte Carlo data [cm], up to 1 cm
Rr = getRrMonteCarlo(t,r,mua,mus,g,n); % get Monte Carlo data [1/cm2/s],
      not shown
N = 2048;  % adds zeros to end of time-resolved data, for padding the transform
f = (1:N/2)'/N/dt; % the x-axis frequency of the power spectrum
ir = 10;    % selects one radial position r(ir)
F = abs(fft(Rr(:,ir)*dt, N));  % Rr(:,ir) is the time-resolved Rr(t) at r(ir)
plot(f, F)  % the plot command yields the power spectrum in [1/cm^2]
```

The original data $R_r(t, r)$ has 100 time-points, but the *fft*() operates best when the number of data points is a multiple of 2. Therefore, zeros are added to the end of the data, which is called *padding*. Adding more zeros causes the result to have more points, so the curves look smoother. In this case 1948 zeros were added to yield a final 2048 data points. The above program yielded Fig. 5.9b. Figure 5.9c normalizes the data by the DC value of R_r, so as to emphasize the shape of the power spectra.

Note in this example that light escaping at 1 mm from the source has higher frequency content than light escaping at 6 mm from the source. In practical frequency domain measurements, the detection of light transport is usually made at least 10 mm from the source, where diffusion theory suffices to describe light transport, and the frequency content of interest is in the hundreds of MHz [1/s]. This example shows how time-resolved Monte Carlo data can be converted to the frequency domain to address questions where diffusion theory is inadequate.

5.6 Summary

Monte Carlo simulations are a relatively simple and flexible method for exploring the behavior of light transport in biological tissues or other media with average absorption and scattering properties. The simulations are like experiments involving a number of photons ($N_{photons}$) and hence the simulations take time to execute. Some simulations take only a minute, and others take hours or even days. If you are trying to fill small bins with rare photons, the simulations take a lot of time. But common problems take about 10 min or less to get a good result.

The strength of Monte Carlo simulations is to treat situations where diffusion theory, or some other analytic expression of light transport, fails. It is not the proper tool for every job.

5.7 Appendix: `mc321.c`

A simplest version of a Monte Carlo simulation, called `mc321.c`, is listed below. The program can be downloaded from the web [8], but since websites may change, the program is listed here as an archival example, and as an easy reference while reading this chapter. This program does not consider escape across boundaries, but shows the basic format of photon propagation, recording and normalization. The output of this program is plotted in Fig. 5.10:

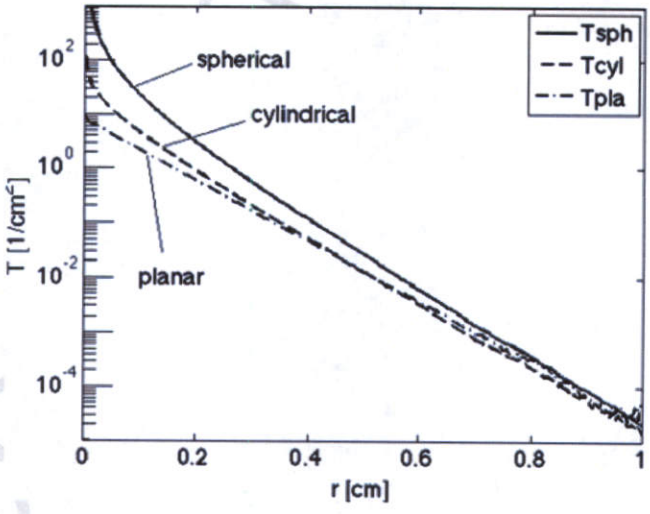

Fig. 5.10 The data in `mc321.out`, produced by `mc321.c`, plotted using a separate graphics program. The spherical transport from a point source, the cylindrical transport from a line source, and the planar transport from a planar source are shown versus the distance, *r*, from the source

```
/*****************************************
 *  mc321.c    , in ANSI Standard C programming language
 *
 *  Monte Carlo simulation yielding spherical, cylindrical, and planar
 *    responses to an isotropic point source (equivalent to a plane source,
```

```
1216  *     line source, and point source, respectively) in an infinite homogeneous
1217  *     medium with no boundaries. This program is a minimal Monte Carlo
1218  *     program scoring photon distributions in spherical, cylindrical,
1219  *     and planar shells.
1220  *
1221  *  by Steven L. Jacques based on prior collaborative work
1222  *     with Lihong Wang, Scott Prahl, and Marleen Keijzer.
1223  *     partially funded by the NIH (R29-HL45045, 1991-1997) and
1224  *     the DOE (DE-FG05-91ER617226, DE-FG03-95ER61971, 1991-1999).
1225  *
1226  *  A published report illustrates use of the program:
1227  *     S. L. Jacques: "Light distributions from point, line, and plane
1228  *     sources for photochemical reactions and fluorescence in turbid
1229  *     biological tissues," Photochem. Photobiol. 67:23-32, 1998.
1230  **********/
1231
1232  #include <math.h>
1233  #include <stdio.h>
1234
1235  #define Nbins           500
1236  #define Nbinsp1         501
1237  #define PI          3.1415926
1238  #define LIGHTSPEED    2.997925E10 /* in vacuo speed of light [cm/s] */
1239  #define ALIVE       1                   /* if photon not yet terminated */
1240  #define DEAD        0                   /* if photon is to be terminated */
1241  #define THRESHOLD   0.01                /* used in roulette */
1242  #define CHANCE      0.1                 /* used in roulette */
1243  #define COS90D      1.0E-6
1244       /* If cos(theta) <= COS90D, theta >= PI/2 - 1e-6 rad. */
1245  #define ONE_MINUS_COSZERO 1.0E-12
1246       /* If 1-cos(theta) <= ONE_MINUS_COSZERO, fabs(theta) <= 1e-6 rad. */
1247       /* If 1+cos(theta) <= ONE_MINUS_COSZERO, fabs(PI-theta) <= 1e-6 rad. */
1248  #define SIGN(x)           ((x)>=0 ? 1:-1)
1249  #define InitRandomGen     (double) RandomGen(0, 1, NULL)
1250       /* Initializes the seed for the random number generator. */
1251  #define RandomNum         (double) RandomGen(1, 0, NULL)
1252       /* Calls for a random number from the randum number generator. */
1253
1254  /* DECLARE FUNCTION */
1255  double RandomGen(char Type, long Seed, long *Status);
1256       /* Random number generator */
1257  main() {
1258
1259  /* Propagation parameters */
1260  double  x, y, z;     /* photon position */
```

```
1261  double  ux, uy, uz; /* photon trajectory as cosines */
1262  double  uxx, uyy, uzz;  /* temporary values used during SPIN */
1263  double  s;          /* step sizes. s = -log(RND)/mus [cm] */
1264  double  costheta;   /* cos(theta) */
1265  double  sintheta;   /* sin(theta) */
1266  double  cospsi;     /* cos(psi) */
1267  double  sinpsi;     /* sin(psi) */
1268  double  psi;        /* azimuthal angle */
1269  double  i_photon;   /* current photon */
1270  double  W;          /* photon weight */
1271  double  absorb;     /* weighted deposited in a step due to absorption */
1272  short   photon_status;  /* flag = ALIVE=1 or DEAD=0 */
1273
1274  /* other variables */
1275  double  Csph[Nbinsp1];  /* spherical   photon concentration CC[ir=0..100] */
1276  double  Ccyl[Nbinsp1];  /* cylindrical photon concentration CC[ir=0..100] */
1277  double  Cpla[Nbinsp1];  /* planar      photon concentration CC[ir=0..100] */
1278  double  Fsph;       /* fluence in spherical shell */
1279  double  Fcyl;       /* fluence in cylindrical shell */
1280  double  Fpla;       /* fluence in planar shell */
1281  double  mua;        /* absorption coefficient [cm^-1] */
1282  double  mus;        /* scattering coefficient [cm^-1] */
1283  double  g;          /* anisotropy [-] */
1284  double  albedo;     /* albedo of tissue */
1285  double  nt;         /* tissue index of refraction */
1286  double  Nphotons;   /* number of photons in simulation */
1287  short   NR;         /* number of radial positions */
1288  double  radial_size;  /* maximum radial size */
1289  double  r;          /* radial position */
1290  double  dr;         /* radial bin size */
1291  short   ir;         /* index to radial position */
1292  double  shellvolume;  /* volume of shell at radial position r */
1293  double  CNT;        /* total count of photon weight summed over all bins */
1294
1295  /* dummy variables */
1296  double  rnd;        /* assigned random value 0-1 */
1297  short   i, j;       /* dummy indices */
1298  double  u, temp;    /* dummy variables */
1299  FILE*   target;     /* point to output file */
1300
1301  /**** INPUT
1302     Input the optical properties
1303     Input the bin and array sizes
1304     Input the number of photons
1305  *****/
```

```
1306  mua          = 1.673;      /* cm^-1 */
1307  mus          = 312.0;   /* cm^-1 */
1308  g            = 0.90;
1309  nt           = 1.33;
1310  Nphotons     = 10000; /* set number of photons in simulation */
1311  radial_size = 2.0;    /* cm, total range over which bins extend */
1312  NR           = Nbins;      /* set number of bins.  */
1313  /* IF NR IS ALTERED, THEN USER MUST ALSO ALTER THE ARRAY DECLARATION TO A
1314       SIZE = NR+1. */
1315  dr           = radial_size/NR;  /* cm */
1316  albedo       = mus/(mus + mua);
1317
1318  /**** INITIALIZATIONS
1319  *****/
1320  i_photon = 0;
1321  InitRandomGen;
1322  for (ir=0; ir<=NR; ir++) {
1323     Csph[ir] = 0;
1324     Ccyl[ir] = 0;
1325     Cpla[ir] = 0;
1326     }
1327
1328
1329  /**** RUN
1330     Launch N photons, initializing each one before progation.
1331  *****/
1332  do {
1333
1334  /**** LAUNCH
1335     Initialize photon position and trajectory.
1336     Implements an isotropic point source.
1337  *****/
1338  i_photon += 1;  /* increment photon count */
1339  if ( fmod(i_photon, Nphotons/10)  == 0)
1340      printf("%0.0f%% done\n", i_photon/Nphotons*100);
1341
1342  W = 1.0;                    /* set photon weight to one */
1343  photon_status = ALIVE;      /* Launch an ALIVE photon */
1344
1345  x = 0;                      /* Set photon position to origin. */
1346  y = 0;
1347  z = 0;
1348
1349  /* Randomly set photon trajectory to yield an isotropic source. */
1350  costheta = 2.0*RandomNum - 1.0;
```

```
1351  sintheta = sqrt(1.0 - costheta*costheta);     /* sintheta is always positive */
1352  psi = 2.0*PI*RandomNum;
1353  ux = sintheta*cos(psi);
1354  uy = sintheta*sin(psi);
1355  uz = costheta;
1356
1357  /* HOP_DROP_SPIN_CHECK
1358     Propagate one photon until it dies as determined by ROULETTE.
1359  *******/
1360  do {
1361
1362  /**** HOP
1363     Take step to new position
1364     s = step size
1365     ux, uy, uz are cosines of current photon trajectory
1366  *****/
1367    while ((rnd = RandomNum) <= 0.0);   /* yields 0 < rnd <= 1 */
1368    s = -log(rnd)/(mua + mus);          /* Step size.  Note: log() is base e */
1369    x += s * ux;                        /* Update positions. */
1370    y += s * uy;
1371    z += s * uz;
1372  /**** DROP
1373     Drop photon weight (W) into local bin.
1374  *****/
1375     absorb = W*(1 - albedo);       /* photon weight absorbed at this step */
1376     W -= absorb;                   /* decrement WEIGHT by amount absorbed */
1377
1378     /* spherical */
1379     r = sqrt(x*x + y*y + z*z);     /* current spherical radial position */
1380     ir = (short)(r/dr);            /* ir = index to spatial bin */
1381     if (ir >= NR) ir = NR;         /* last bin is for overflow */
1382     Csph[ir] += absorb;            /* DROP absorbed weight into bin */
1383
1384     /* cylindrical */
1385     r = sqrt(x*x + y*y);           /* current cylindrical radial position */
1386     ir = (short)(r/dr);            /* ir = index to spatial bin */
1387     if (ir >= NR) ir = NR;         /* last bin is for overflow */
1388     Ccyl[ir] += absorb;            /* DROP absorbed weight into bin */
1389
1390     /* planar */
1391     r = fabs(z);                   /* current planar radial position */
1392     ir = (short)(r/dr);            /* ir = index to spatial bin */
1393     if (ir >= NR) ir = NR;         /* last bin is for overflow */
1394     Cpla[ir] += absorb;            /* DROP absorbed weight into bin */
1395
```

```
1396  /**** SPIN
1397     Scatter photon into new trajectory defined by theta and psi.
1398     Theta is specified by cos(theta), which is determined
1399     based on the Henyey-Greenstein scattering function.
1400     Convert theta and psi into cosines ux, uy, uz.
1401  *****/
1402    /* Sample for costheta */
1403    rnd = RandomNum;
1404       if (g == 0.0)
1405          costheta = 2.0*rnd - 1.0;
1406       else {
1407          double temp = (1.0 - g*g)/(1.0 - g + 2*g*rnd);
1408          costheta = (1.0 + g*g - temp*temp)/(2.0*g);
1409          }
1410    sintheta = sqrt(1.0 - costheta*costheta); /* sqrt() is faster than sin(). */
1411    /* Sample psi. */
1412    psi = 2.0*PI*RandomNum;
1413    cospsi = cos(psi);
1414    if (psi < PI)
1415      sinpsi = sqrt(1.0 - cospsi*cospsi);     /* sqrt() is faster than sin(). */
1416    else
1417      sinpsi = -sqrt(1.0 - cospsi*cospsi);
1418
1419    /* New trajectory. */
1420    if (1 - fabs(uz) <= ONE_MINUS_COSZERO) {      /* close to perpendicular. */
1421      uxx = sintheta * cospsi;
1422      uyy = sintheta * sinpsi;
1423      uzz = costheta * SIGN(uz);   /* SIGN() is faster than division. */
1424      }
1425    else {                                        /* usually use this option */
1426      temp = sqrt(1.0 - uz * uz);
1427      uxx = sintheta * (ux * uz * cospsi - uy * sinpsi) / temp + ux * costheta;
1428      uyy = sintheta * (uy * uz * cospsi + ux * sinpsi) / temp + uy * costheta;
1429      uzz = -sintheta * cospsi * temp + uz * costheta;
1430      }
1431
1432    /* Update trajectory */
1433    ux = uxx;
1434    uy = uyy;
1435    uz = uzz;
1436
1437  /**** CHECK ROULETTE
1438     If photon weight below THRESHOLD, then terminate photon using Roulette
1439        technique.
1440
```

```
1441    Photon has CHANCE probability of having its weight increased by factor of
1442       1/CHANCE,
1443    and 1-CHANCE probability of terminating.
1444 *****/
1445 if (W < THRESHOLD) {
1446    if (RandomNum <= CHANCE)
1447       W /= CHANCE;
1448    else photon_status = DEAD;
1449    }
1450 } /* end STEP_CHECK_HOP_SPIN */
1451 while (photon_status == ALIVE);
1452
1453   /* If photon dead, then launch new photon. */
1454 } /* end RUN */
1455 while (i_photon < Nphotons);
1456
1457 /**** SAVE
1458    Convert data to relative fluence rate [cm^-2] and save to file called
1459      "mcmin321.out".
1460 *****/
1461 target = fopen("mc321.out", "w");
1462
1463 /* print header */
1464 fprintf(target, "number of photons = %f\n", Nphotons);
1465 fprintf(target, "bin size = %5.5f [cm] \n", dr);
1466 fprintf(target, "last row is overflow. Ignore.\n");
1467
1468 /* print column titles */
1469 fprintf(target, "r [cm] \t Fsph [1/cm2] \t Fcyl [1/cm2] \t Fpla [1/cm2]\n");
1470
1471 /* print data:  radial position, fluence rates for 3D, 2D, 1D geometries */
1472 for (ir=0; ir<=NR; ir++) {
1473     /* r = sqrt(1.0/3 - (ir+1) + (ir+1)*(ir+1))*dr; */
1474     r = (ir + 0.5)*dr;
1475     shellvolume = 4.0*PI*r*r*dr; /* per spherical shell */
1476     Fsph = Csph[ir]/Nphotons/shellvolume/mua;
1477     shellvolume = 2.0*PI*r*dr;   /* per cm length of cylinder */
1478     Fcyl = Ccyl[ir]/Nphotons/shellvolume/mua;
1479     shellvolume = dr;            /* per cm2 area of plane */
1480     Fpla =Cpla[ir]/Nphotons/shellvolume/mua;
1481     fprintf(target, "%5.5f \t %4.3e \t %4.3e \t %4.3e \n", r, Fsph, Fcyl, Fpla);
1482     }
1483
1484 fclose(target);
1485
```

```
1486  } /* end of main */
1487
1488  /* SUBROUTINES */
1489
1490  /*****************************************************************************
1491   *      RandomGen
1492   *      A random number generator that generates uniformly
1493   *      distributed random numbers between 0 and 1 inclusive.
1494   *      The algorithm is based on:
1495   *      W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P.
1496   *      Flannery, "Numerical Recipes in C," Cambridge University
1497   *      Press, 2nd edition, (1992).
1498   *      and
1499   *      D.E. Knuth, "Seminumerical Algorithms," 2nd edition, vol. 2
1500   *      of "The Art of Computer Programming", Addison-Wesley, (1981).
1501   *
1502   *      When Type is 0, sets Seed as the seed. Make sure 0<Seed<32000.
1503   *      When Type is 1, returns a random number.
1504   *      When Type is 2, gets the status of the generator.
1505   *      When Type is 3, restores the status of the generator.
1506   *
1507   *      The status of the generator is represented by Status[0..56].
1508   *
1509   *      Make sure you initialize the seed before you get random
1510   *      numbers.
1511   ****/
1512  #define MBIG 1000000000
1513  #define MSEED 161803398
1514  #define MZ 0
1515  #define FAC 1.0E-9
1516
1517  double RandomGen(char Type, long Seed, long *Status){
1518    static long i1, i2, ma[56];   /* ma[0] is not used. */
1519    long        mj, mk;
1520    short       i, ii;
1521
1522    if (Type == 0) {              /* set seed. */
1523      mj = MSEED - (Seed < 0 ? -Seed : Seed);
1524      mj %= MBIG;
1525      ma[55] = mj;
1526      mk = 1;
1527      for (i = 1; i <= 54; i++) {
1528        ii = (21 * i) % 55;
1529        ma[ii] = mk;
1530        mk = mj - mk;
          if (mk < MZ)
```

```
1531          mk += MBIG;
1532        mj = ma[ii];
1533      }
1534      for (ii = 1; ii <= 4; ii++)
1535        for (i = 1; i <= 55; i++) {
1536          ma[i] -= ma[1 + (i + 30) % 55];
1537          if (ma[i] < MZ)
1538            ma[i] += MBIG;
1539        }
1540      i1 = 0;
1541      i2 = 31;
1542    } else if (Type == 1) {       /* get a number. */
1543      if (++i1 == 56)
1544        i1 = 1;
1545      if (++i2 == 56)
1546        i2 = 1;
1547      mj = ma[i1] - ma[i2];
1548      if (mj < MZ)
1549        mj += MBIG;
1550      ma[i1] = mj;
1551      return (mj * FAC);
1552    } else if (Type == 2) {       /* get status. */
1553      for (i = 0; i < 55; i++)
1554        Status[i] = ma[i + 1];
1555      Status[55] = i1;
1556      Status[56] = i2;
1557    } else if (Type == 3) {       /* restore status. */
1558      for (i = 0; i < 55; i++)
1559        ma[i + 1] = Status[i];
1560      i1 = Status[55];
1561      i2 = Status[56];
1562    } else
1563      puts("Wrong parameter to RandomGen().");
1564    return (0);
1565  }
1566  #undef MBIG
1567  #undef MSEED
1568  #undef MZ
1569  #undef FAC
1570
1571
```

References

1572
1573
1. Wilson BC and Adam G. A Monte Carlo model for the absorption and flux distributions of light in tissue. *Med. Phys.* 10:824–830 (1983).
1574
1575
2. Keijzer M, Jacques SL, Prahl SA, and Welch AJ. Light distributions in artery tissue: Monte Carlo simulations for finite-diameter laser beams. *Lasers Surg. Med.* 9:148–154 (1989).

3. Prahl, SA, Keijzer, Jacques SL, and Welch AJ. A Monte Carlo model of light propagation in tissue. In: G Müller and D Sliney (eds) *Dosimetry of laser radiation in medicine and biology*, SPIE Series, Vol. IS 5, pp. 102–111 (1989).
4. Wang L-H, Jacques SL, and Zheng L-Q. MCML – Monte Carlo modeling of photon transport in multi-layered tissues. *Comput. Methods Programs Biomed.*, 47:131–146 (1995).
5. Jacques, SL, http://omlc.ogi.edu/software/mc/mcml, Oregon Health & Science University, 2010. This site includes a 178-page manual on MCML. Also, a convolution program, CONV, is available for convolving the point spread functions generated by MCML.
6. Wang, LV, http://labs.seas.wustl.edu/bme/Wang/mc.html, Washington University in St. Louis, 2010. Alternative site for obtaining MCML, CONV and the manual.
7. Jacques SL. Light distributions from point, line, and plane sources for photochemical reactions and fluorescence in turbid biological tissues. *Photochem. Photobiol.* 67:23–32 (1998).
8. Jacques, SL, http://omlc.ogi.edu/software/mc/mc321, Oregon Health & Science University, 2010. This site lists the minimal Monte Carlo program mc321.m, used in Ref. 7.
9. Jacques SL. Monte Carlo simulations of fluorescence in turbid media, Ch. 6. In: MA Mycek and BW Pogue (eds) *Handbook of biomedical fluorescence*. Marcel-Dekker, New York, NY (2003).
10. Jacques, SL, http://omlc.ogi.edu/software/mc/mcsub, Oregon Health & Science University, 2010. This site lists the subroutine mcsub() that can be called by c programs to run a Monte Carlo simulation.
11. Ramella-Roman JC, Prahl SA, and Jacques SL. Three Monte Carlo programs of polarized light transport into scattering media: part I. *Opt. Express* 13(12):4420–4438(2005).
12. Ramella-Roman JC, Prahl SA, and Jacques SL, http://omlc.ogi.edu/software/mc/polarizedlight, Oregon Health & Science University, 2010. This site lists the program for Monte Carlo simulation of polarized light, reported in Ref. 11.

Ashley J. Welch · Martin J.C. van Gemert
Editors

Optical-Thermal Response of Laser-Irradiated Tissue

Second Edition

Springer

Editors
Dr. Ashley J. Welch
University of Texas, Austin
Dept. Biomedical Engineering
University Station 1
78712 Austin Texas
ENS12, Campus Code C0800
USA
Welch@mail.utexas.edu

Dr. Martin J.C. van Gemert
University of Amsterdam
Academic Medical Centre
Laser Centre
Meibergdreef 9
1105 AZ Amsterdam
Netherlands
m.j.vangemert@amc.uva.nl

ISBN 978-90-481-8830-7 e-ISBN 978-90-481-8831-4
DOI 10.1007/978-90-481-8831-4
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: xxxxx

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)