ValoMC: a Monte Carlo software and MATLAB toolbox for simulating light transport in biological tissue

ALEKSI A LEINO,[*] AKI PULKKINEN, [ID] AND TANJA TARVAINEN

Department of Applied Physics, University of Eastern Finland, P.O. Box 1627, 70211 Kuopio, Finland
[*]*Corresponding author: aleksi.leino@uef.fi*

Abstract: A Monte Carlo method for photon transport has gained wide popularity in biomedical optics for studying light behaviour in tissue. Nowadays, typical computation times range from a few minutes to hours. Although various implementations of the Monte Carlo algorithm exist, there is only a limited number of free software available. In addition, these packages may require substantial learning efforts. To address these issues, we present a new Monte Carlo software with a user-friendly interface. The simulation geometry is defined using an unstructured (triangular or tetrahedral) mesh. The program solves the photon fluence in the computation domain and the exitance at the domain boundary. It is capable of simulating complex measurement geometries with spatially varying optical parameter distributions and supports several types of light sources as well as intensity modulated light. Furthermore, attention is given to ease of use and fast problem set up with a MATLAB (The MathWorks Inc., Natick, MA) interface. The simulation code is written in C++ and parallelized using OpenMP. The simulation code has been validated against analytical and numerical solutions of radiative transfer equation and other Monte Carlo software in good agreement. The software is available for download from the homepage https://inverselight.github.io/ValoMC/ and the source code from GitHub https://github.com/InverseLight/ValoMC.

1. Introduction

Monte Carlo method for photon transport (hereafter referred to as 'Monte Carlo') is used for simulating how photons are scattered and absorbed within a medium. It has been used to simulate light propagation in tissue for a variety of biomedical applications, e. g. near-infrared spectroscopy, diffuse optical imaging, photoacoustic tomography and light-based therapies [1,2]. A key feature of the method, as the name Monte Carlo implies, is utilisation of random numbers in propagating photons in a stochastic manner. Due to its reliability, Monte Carlo has been regarded as the 'gold standard' for simulating light transport in biological tissue. It is generally used as the reference approach for deterministic methods which are typically analytical or numerical solutions of the radiative transfer equation or its approximations such as the diffusion approximation.

Numerous implementations of the Monte Carlo algorithm exist [1,2]. One of the first widely utilised Monte Carlo method in biomedical optical imaging was the photon packet method, introduced for biomedical optics by Prahl et al. [3]. In photon packet method, the efficiency of a Monte Carlo program is increased by propagating many photons, i.e. a photon packet, along each pathway. Instead of considering the absorption of a single photon as a stochastic event, it is treated by reducing the weight (size) of the photon packet. Various implementations of the photon packet method followed.

In the early implementations, photon fluence was computed in geometries with a predefined shape such as layered spheres [4] and multi-layered slab geometries [5]. Later, implementations in other predefined shapes [6,7] and in more general complex geometries followed [8–10]. Typically in complex geometries, Monte Carlo implementations have been implemented using

two-dimensional (2D) pixel or three-dimensional (3D) voxel discretisations [8–15]. It has been shown that approximating boundaries in voxel basis may lead to large errors in the solution of light transport simulation [16]. Therefore, Monte Carlo implementations in unstructured meshes consisting of triangles or tetrahedra have been constructed for better and more accurate solutions in complex heterogeneous geometries [17–24]. In addition to the discretisation, Monte Carlo implementations differ also depending on how they sample scattering length and reduce the weight of the photon packet. For more information on the comparison of different Monte Carlo implementations see e. g. articles [1,25] where it has also been shown that these different implementations are statistically equivalent.

During the past decades, the huge development of computers has enabled parallised and Graphics Processing Unit (GPU) advanced Monte Carlo implementations that can be used to obtain solutions to complex three-dimensional simulation problems in a time-frame from few seconds to a few minutes [12,26–30]. Thus, unlike at the early 1990s, when the available computational resources limited stochastic simulation methods such as Monte Carlo as a reference approach which took hours or even weeks to run, the current Monte Carlo implementations can be easily utilised for various problems. This has made them a tempting alternative for other type of problems as well. Monte Carlo simulations have been used, for example, in studying photoacoustics [31,32], fluorescence tomography [33,34], time resolved diffuse spectroscopy [35,36], statistical models for photon transport [37], biophotonics of corals [38], dose optimization for photodynamic therapy [39] and laser treatment of port-wine stains [40]. Furthermore, the first studies in which Monte Carlo is utilised in the solution of the inverse problem of an optical imaging problem have been implemented [15,41–46].

Although various implementations of Monte Carlo algorithm exists, there is only a limited number of free software available, see e. g. [5,12,14,17,22,29]. In this paper, our Monte Carlo software ValoMC is described. The geometry is defined using unstructured meshes, i. e. triangles in 2D and tetrahedra in 3D. The software consists of the main simulation code, written in C++, and auxiliary MATLAB (The MathWorks Inc., Natick, MA) functions. These functions serve as an easy to use interface for anyone who wants to run Monte Carlo simulations for their problem but are not comfortable with technical programming. The simulation code is based on our earlier Monte Carlo software that has previously been utilised in simulating light propagation for example in [20,23,47–51]. A summary of other Monte Carlo simulation software that have MATLAB interface can be found in Ref. [52].

The rest of the paper is organised as follows. First, the theory is shortly reviewed in Section 2. Then, program description and usage guidelines for the MATLAB interface are given in Section 3, followed by usage examples and validation studies in Section 4. Software and hardware specifications and availability are discussed in Section 5. A summary is given in Section 6.

2.   Theory and computational methods

Four parameters are used to represent the optical properties of medium [53]: absorption coefficient μ_a, scattering coefficient μ_s, scattering anisotropy in the Henyey-Greenstein phase function g and refractive index n, and are all function of the position within the medium. Absorption μ_a and scattering μ_s are defined so that the probability for the photon to be absorbed or scattered within an infinitesimal length dx along the propagation direction are $\mu_a dx$ and $\mu_s dx$, respectively, and assumed to be independent. Further, the Henyey-Greenstein phase function represents the directionality of the scattering [54]. In contrast to e. g. Rayleigh and Mie phase functions, it is not derived from a physical theory. Rather it simply serves as mathematical representation of the scattering medium. For example, when $g = 0$ the scattering is isotropic and when it is close to 1, the photon scattering angles become small. Its numerical value can be determined using experimental data or fits to theoretical considerations. Henyey-Greenstein is the most commonly used phase function for biological tissue but other functions are used as well [55]. The refractive

index determines the amount of light that is reflected or refracted on tissue interface. In this work, the spatial dependence of all the parameters is described using a triangular or tetrahedral mesh, and considering them as piecewise constant values.

2.1.   Geometry description

A coordinate matrix r, a topology matrix H and a boundary topology matrix B_H represent a triangular or a tetrahedral mesh. Examples of such 2D and 3D meshes are shown in Fig. 1. Although we consider here very simple shapes for illustration purposes, general complicated shapes can be constructed and utilised in ValoMC. Let us consider a simple example geometry shown in Fig. 1(b). To set up a quadrilateral region in 2D that consists of two triangles, the following coordinate matrix can be used

$$r = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} = \begin{pmatrix} -0.8 & -0.7 \\ -0.2 & 1.4 \\ 1.6 & 1.2 \\ 1.8 & -1 \end{pmatrix} \tag{1}$$

where each row represent a coordinate. To define the two triangles using the coordinate matrix, an accompanying topology matrix is supplied

$$H = \begin{pmatrix} H_1 \\ H_2 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 1 \end{pmatrix}. \tag{2}$$

Here, each row refers to a single triangle, and each element to a row in r. In addition, the region must be enclosed by boundary elements. In 2D, these are simply lines that are defined in matrix B_H. The structure is similar to that of H. In the current example,

$$B_H = \begin{pmatrix} B_{H_1} \\ B_{H_2} \\ B_{H_3} \\ B_{H_4} \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 1 \end{pmatrix} \tag{3}$$

i. e. each row in B_H refers to a single line segment and each element refers to a row in r. The lines form a quad that encloses the region formed by the two triangles.

A neighbourhood matrix H_N is used to make finding the neighbouring elements and boundary elements for a given element more efficient. This matrix is only used internally and the description is provided here for completeness. Along the same lines, each row in H_N refers to a single element and each element to a neighbouring element.

Negative indices are used to distinguish boundary elements from regular elements. The neighbourhood matrix for the current example is

$$H_N = \begin{pmatrix} -1 & -2 & 2 \\ -3 & -4 & 1 \end{pmatrix} \tag{4}$$

i. e. the first triangle (r_1, r_2, r_3) is surrounded by boundary elements 1, 2 and element 2.

The generalization to a cube in 3D is straightforward, as visualised in Fig. 1. In 3D, the fundamental element is a tetrahedron. The four corners of the tetrahedron are similarly defined using a four-columned matrix H while the boundary elements are triangles. They can be set up using a three column B_H matrix which has the same structure as H in 2D.
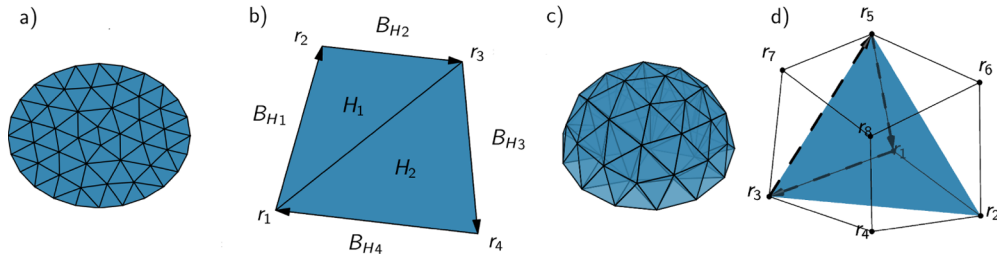
Fig. 1. Illustration of the matrices r (coordinates), H (element topology) and B_H (boundary element topology). (a) Shown is a 2D circle constructed from triangles. (b) The geometry for the simple example discussed in the text. (c) 3D half-sphere constructed from tetrahedrons. (d) A tetrahedron $H_1 = (1\ 3\ 5\ 2)$ indicated with coloured region and a boundary element $B_{H1} = (1\ 3\ 5)$ indicated with a dashed line.

2.2. Algorithm

The flowchart of the algorithm is shown in Fig. 2. Our implementation follows Ref. [3] with additions due to the mesh based geometry and difference in path weighting. Random numbers are generated using the Mersenne-Twister algorithm [56].
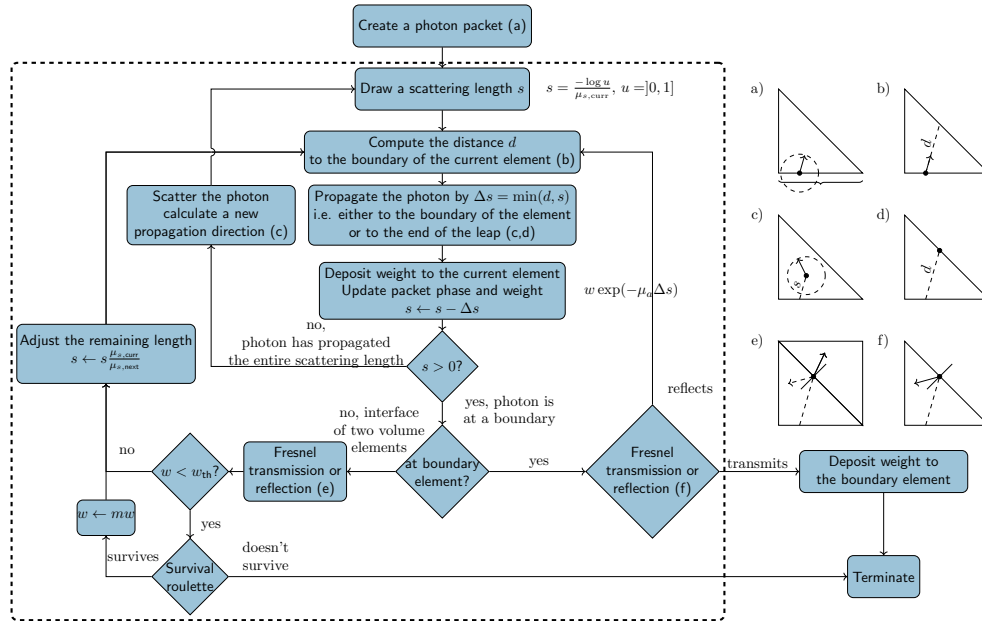

Fig. 2. Flowchart of the algorithm.

The algorithm starts with the creation of a photon packet (Fig. 2(a)). A random location from those boundary elements that act as a light source is selected. Further, the propagation direction (angle) is drawn from a probability density distribution that can be defined for each boundary element independently (see Section 3). The photon packet is initialised with a given weight $w = 1$. This number describes the relative number of photons in the packet.

After the photon packet is initiated, the program enters in the main loop that ends in the termination of the packet. In the loop, a random scattering length s, is drawn so that the drawn

lengths follow an exponential probability density distribution

$$P(s) = \mu_{s,\mathrm{curr}} \exp(-\mu_{s,\mathrm{curr}} s) \tag{5}$$

where $\mu_{s,\mathrm{curr}}$ refers to the scattering coefficient of element the packet is located. The trajectory of the packet is not affected by the absorption coefficient. Instead, absorption coefficient along the path is used only to calculate a weight for the trajectory by Beer-Lambert law. This approach can be formally motivated by considering the solution of the radiative transfer equation for a ray-like initial condition (see e. g. 'microscopic Beer-Lambert law' in Ref. [1]) and noticing that the absorption coefficient enters in an exponential weighting factor for the intensity along the ray.

The scattering length s is exhausted by moving the packet at a straight line either to the interface of the next element or, if the distance to the interface is greater than the remaining length, to the end of the distance (Fig. 2(c,d)). While the packet moves within the medium, it is affected by the optical coefficients in each element on its path. If the scattering coefficient changes along the path, the remaining length is adjusted to account for the change by scaling the remaining length by $\mu_{s,\mathrm{curr}}/\mu_{s,\mathrm{next}}$ where $\mu_{s,\mathrm{next}}$ refers to the scattering coefficient of the element the packet is entering. At the interface of two elements, the photon packet may undergo Fresnel reflection or be transmitted to the neighbouring element with a modified propagation direction due to a change in the refractive index (Fig. 2(e)). Throughout the propagation, the weight of the photon packet is reduced to

$$w \leftarrow w \exp(-\mu_{a,\mathrm{curr}} \Delta s) \tag{6}$$

where $\mu_{a,\mathrm{curr}}$ is the absorption coefficient of element the packet is located, Δs is the distance propagated during the current step.

After the photon packet has propagated the distance s, a new scattering event occurs. In the scattering event, the packet obtains a new direction that is drawn so that the probability of the direction follows Henyey-Greenstein phase function. This is achieved by using the inversion method [57], i.e. uniformly distributed random numbers in the inverse of the Henyey-Greenstein phase function. The current implementation draws the scattering angle in 2D and the cosine of the scattering angle in 3D (similarly as in Ref. [3]). Then, a new scattering length is drawn, and the main loop of the simulation continues.

The photon packet is terminated if the packet exits the computation domain. In addition, the packet can be terminated in a procedure known as 'survival roulette' [3] before drawing a new scattering length. The survival roulette is performed in order to avoid the computation of excessively long propagation paths with a low influence on the computation results. Once the photon packet weight has decreased below a threshold value (by default 10^{-3}), it is either terminated with a probability of $1 - 1/m$ (by default $m = 10$) or its weight is promoted to mw with probability $1/m$. Thereby, the total weight of all photon packets is well conserved despite the terminations.

The main outcome of the simulation is the fluence distribution in the elements of the computation domain and the exitance at the boundary elements. As discussed previously, an initial weight of one is assigned to each packet, and throughout the propagation, the weight of the packet is reduced. Conversely, a weight of $w(1 - \exp(-\mu_a \Delta s))$ is added to the element through which the packet propagates. When the packet reaches a boundary element and exits the computation domain (i.e. does not undergo Fresnel reflection at the boundary, c.f. Fig. 2(f)), the remaining weight is added to the boundary element. Since a photon packet has an initial weight of one, the total weight that is deposited in the computational domain (including its boundary) is equal to the number of photon packets N. The piecewise constant values obtained in this way represent the weight W_i absorbed into the element i and the weight W_j escaped through boundary element j, per unit time. At the end of the simulation, the fluence Φ_i at element i is obtained from

$$\Phi_i = \frac{W_i}{\mu_{a,i} N V_i} \tag{7}$$

and the exitance J_j^+ for boundary element j

$$J_j^+ = \frac{W_j}{N A_j} \tag{8}$$

where V_i and A_j refer volume and area (or area and length in 2D) of the elements. This normalisation fixes the sum of the total absorbed and escaped power to be one unit, in other words, the total power emitted by all light sources is 1 W.

2.3. Frequency domain calculation

Frequency domain Monte Carlo (i.e. intensity modulated light sources) are implemented by using a complex weight for the photon packet [2]. This approach is motivated by noticing that the radiative transfer equation in the frequency domain can be interpreted as time-independent one but with a complex absorption coefficient $\mu_a + \frac{i\omega}{c}$ where ω is the angular frequency of the intensity modulation, c is the speed of light in the medium and i is the imaginary unit. Therefore, given an initial weight w and a distance Δs propagated within an element, the new weight is computed according to

$$w \leftarrow w \exp\left[-\left(\mu_{a,\text{curr}} + \frac{i\omega n_{\text{curr}}}{c_0}\right)\Delta s\right] \tag{9}$$

where c_0 is the speed of light in vacuum, n_{curr} is the refractive index of the element and ω the angular frequency of the light modulation. The weight that is deposited to the elements and boundary elements follow the same formulae as in the previous section, while a complex number is deposited. Likewise, the normalisation of the complex fluence and exitance is done as in Eqs. (7) and (8) while N_i, N_j and $\mu_{a,i}$ are complex numbers. The resulting complex fluence and exitance depict their amplitude and relative phase to the source with an angular frequency ω, and the normalization fixes the boundary integral of the source amplitude to be 1 W.

3. Program description

The program consists of two parts: the simulation code (written in C++) and the MATLAB interface. While the C++ program can be used independently without MATLAB, this section focuses on describing the MATLAB interface part that is intended to facilitate the initialisation of the simulations and the visualisation of the results. Documentation for the C++ code can be found in "cpp/README".

A summary of the currently implemented features of the software is given Table 1. It should be noted ValoMC does not provide a fixed pipeline from meshing to plotting the results. It focuses only on the simulation part. Usage examples on how to construct a custom pipeline are provided. ValoMC stores the mesh and the results in a format that is directly compatible with the plotting tools in MATLAB. Functions to import meshes and export results to specialized software are, in addition, provided. As an output, the computation returns the exitance at each boundary element and the fluence at each element in the computation domain. Therefore, in the current version, detectors (or light sources) with a more complicated spatial or angular dependence can be implemented by making the mesh finer or extending the mesh. Spatial sensitivity profiles of detectors can be implemented by weighting the boundary solution from a finer mesh in post-processing. Angular sensitivity can be implemented by creating vacuum extensions ($\mu_s = 0$, $\mu_a = 0$) to the mesh at the detector locations. Since photons propagate in straight lines in vacuum, the extension can be adjusted so that the boundary elements at the end of the extension collect photons at desired angles only.

Table 1. A summary of currently implemented features in ValoMC.

Implementation			
Method	the photon packet method	Programming language	C++ (simulation code) MATLAB (user interface)
Operating modes	continuous wave intesity modulated	Parallelisation	CPU (OpenMP)
Mesh type	triangular (2D) tetrahedral (3D)	Support for other toolboxes	k-Wave, Toast++
Runs without MATLAB	yes	Results reproducible using a fixed seed	yes
Input parameters			
Mesh generation	functions provided for simple mesh types NetGen VOL file import for complex meshes	Detectors	implemented by meshing
Material parameters (set for each element individually)	μ_a - absorption coefficient μ_s - scattering coefficient g - anisotropy parameter of the Henyey-Greenstein scattering phase function n - refractive index	Boundary conditions (set for each boundary element individually)	light source directivity patterns • unidirectional • isotropic • gaussian • cosinic light source intensity n - refractive index
Output			
Volumetric solution	Piecewise constant value for fluence	Export file formats	ASCII, X3D
Boundary solution	Piecewise constant value for exitance		

In MATLAB, the simulations are initiated with the command

```
solution = ValoMC (vmcmesh, vmcmedium, vmcboundary, vmcoptions)
```

The MATLAB interface consists of auxiliary functions that help to set up the input structures. These structures are documented in Table 2 while a more detailed description with additional features can be found from the homepage. The fields of `vmcmesh` are the same matrices as described in Section 2.1, `vmcmedium` contains the optical coefficients for each element and `vmcboundary` the boundary conditions for each boundary element. A typical modelling task consists of creating a suitable mesh and setting up the optical coefficients of each element in `vmcmedium` and light sources or other boundary conditions of each boundary element in `vmcboundary`. The fields of `vmcoptions` can be used to set a number of global options, for example, to adjust the total number of photon packets launched (10^6 by default) or to enable a frequency domain calculation. If the options structure is not provided, default values will be used.

Simulations can also be run outside MATLAB by using the functions `exportValoMC` and `importValoMC`. These functions write an input file for the C++ program or read its output for analysis in MATLAB, respectively, and enable computations, for example, on a computing cluster without MATLAB.

Table 2. List of the mandatory fields in the input and output of the main function of ValoMC. More detailed options are provided in the homepage and documentation.

Field name	Description	Unit	Array size 2D/3D
vmcmesh			
r	node coordinates	mm	$N_r \times 2$ / $N_r \times 3$
H	element topology (indices to nodes)	dimensionless	$N_e \times 3$ / $N_e \times 4$
BH	boundary element topology (indices to nodes)	dimensionless	$N_b \times 2$ / $N_b \times 3$
vmcmedium			
refractive_index n		dimensionless	$N_e \times 1^*$
absorption_coefficient μ_a		1/mm	$N_e \times 1^*$
scattering_coefficient μ_s		1/mm	$N_e \times 1^*$
scattering_anisotropy g		dimensionless	$N_e \times 1^*$
vmcboundary			
lightsource	Cell array of strings that determines the light source type in the element e. g. {'gaussian'} or {'direct'}.		$N_b \times 1$
solution			
element_fluence	Piecewise constant value for fluence Φ	W/mm^2	$N_e \times 1$
boundary_exitance	Piecewise constant value for exitance J^+	W/mm^2	$N_b \times 1$

*When the mesh has been created with createGridMesh, these values can be also given as two or three dimensional pixel or voxel maps. In that case, solution will be returned in solution.grid_fluence.

3.1. Creating a mesh, a medium, and a boundary

ValoMC comes with a few convenient functions to create simple meshes for prototyping purposes, such as createRectangularMesh. For creating more complicated meshes, multiple dedicated freeware tools are available, such as iso2mesh [58] and NetGen [59]. It is possible to import NetGen '.vol' ASCII files using importNetGenMesh. The importer returns the region and boundary definitions in addition to the mesh. Commercial software are available [60] that produce meshes in a format easily convertible for ValoMC (c.f. Table 2) within MATLAB. It should be noted that meshing tools often introduce multiply defined boundaries between different regions of the mesh (i.e. boundaries between connected elements) as a part of the boundary definition. Currently, the overlapping parts must be removed before using them as B_H. Alternatively, the function createBH can be used to create a B_H matrix for a given H matrix and overwrite the boundary given by the meshing tool. Furthermore, ValoMC has been made compatible to mesh formats used in some biomedical optics simulation software such as Toast++ [61] and k-Wave [62].

Function createGridMesh is provided for working with pixel or voxel format data. It creates a structured mesh from a grid so that the mesh pixels or voxels can be filled by two triangles or six tetrahedrons, respectively. When the mesh has been created with createGridMesh, the fields of vmcmedium can be introduced as 2D or 3D arrays. Furthermore, the resulting fluence field is returned as a 2D/3D array in solution.grid_fluence.

Once the mesh has been created, functions createMedium and createBoundary can be used to create an accompanying vmcmedium (used to set material parameters) and vmcboundary (used to set boundary conditions) structure, respectively. They take vmcmesh as an input. If createMedium is given a vmcmedium as the second input parameter, it will

repeat the arrays so that they match in size with the mesh. A convenient way to initialise all elements to uniform values is to give a structure with scalar optical coefficients as an input. `createMedium` will set the refractive index at the boundary so that there is no mismatch at the boundary (see Section 3.3).

3.2. Finding boundaries and regions

All material parameters and boundary conditions are set individually for all elements and boundary elements, respectively. However, it is often convenient to group elements for setting material parameters and boundary conditions using a single instruction. This is done by creating vectors that hold indices to B_H (boundary conditions) or to H (material parameters). These matrices were introduced in Fig. 1(b). If such vectors are not provided by the meshing tool, functions `findBoundaries` and `findElements` can be used. These functions take the mesh as an input, a keyword such as 'circle' and optional arguments. A list of keywords is provided in the code documentation and by invoking 'help findElements'. For example, in 2D all elements within a circle of $r<10$ mm with a centre at the origin (0 0) can be retrieved by invoking

```
circle = findElements (vmcmesh, 'circle', [0 0], 10);
```

The return value, `circle`, is a vector that holds indices to `vmcmesh.H`. These indices could be now used to set e.g. absorption coefficient for the elements within the circle by

```
vmcmedium.absorption_coefficient (circle) = 0.02;
```

Note that `circle` can be a single number as well to set the properties of an individual element. Similarly, `findBoundaries` returns indices to `vmcmesh.BH`, i.e. boundary elements. Using command

```
lightsource1 = findBoundaries (vmcmesh, 'direction',...
                               [0 0], [5 0], 1);  will return boundary
```

elements that fall within a rectangle defined by a line from (0 0) to (5 0) that intesects it in the middle and has a width of 1.

3.3. Light sources and other boundary conditions

Light emitting boundary elements in `vmcboundary` are defined by a cell array `lightsource`. Each element of the array is a string that determines the type of the light source. For example, `vmcboundary.lightsource (lightsource1) = 'direct';` will set a light source with a type 'direct' on each boundary element whose index is contained in vector `lightsource1`. All photons in 'direct' light sources are launched in the same direction (by default, in the direction of the normal).

Light sources can be directed using the field `lightsource_direction`. This is a $N_b \times 2$ / $N_b \times 3$ size matrix where each row contains a direction vector (N_b is the number of boundary elements). The direction can be given either as an absolute direction or relative to the normal. The light source types affect the distribution of initial propagation angles with respect to the direction vector. For example, in a {'cosinic'} light source initial angles are random but follow a cosine distribution and in a {'gaussian'}, a Gaussian.

By default, the initial position of the photon packet is selected from all light emitting boundary elements with a uniform probability. This means that all boundary elements that contain a light source have an equal irradiance. Alternatively, the relative strength of each light source element can be determined using the field `lightsource_irradiance`. This field adjusts the probability for a particular boundary element to be selected. For example, if the value of the

field on a single boundary element is set to 2 and the rest of the elements are 1, the element will emit twice as many photons per unit length than the rest of the elements.

The refractive index at the boundary of the computation domain (i.e. the refractive index of the material surrounding the computation domain) can be set for each boundary element individually using the field `exterior_refractive_index`. Only the photon packets that are exiting the computation domain can be reflected at the boundary. In other words, light sources are considered to lie inside the computation domain (a convention that varies among different software).

3.4. Simulation output and visualisation

The solution of a simulation (fluence at each element and exitance at each boundary element) is returned in `solution.element_fluence` and `solution.boundary_fluence`. Visualisation of the solution arrays is straightforward with MATLAB built-in functions such as `patch`, `trimesh` and `tetramesh`. Various usage examples are provided with the code. Visualisation of large tetrahedral meshes can be inefficient using `tetramesh`. For this purpose, the function `exportX3D` is provided. It converts tetrahedral meshes into triangular meshes so that each four triangular face of tetrahedron will have the same value. The resulting X3D file can be opened and post-processed for example in MeshLab [63]. The triangular representation is also convenient for making even cuts to the mesh for visualisation purposes, e. g. using the Geometry Processing Toolbox [64] for MATLAB.

4. Simulations

The simulation code has been validated against analytical and numerical solutions of the radiative transfer equation and other Monte Carlo software. Below, three example studies are described. We have used

$$E = 100\% \cdot \sqrt{\frac{\int_R (f(\mathbf{r}) - f_{\text{ref}}(\mathbf{r}))^2 d\mathbf{r}}{\int_R f_{\text{ref}}(\mathbf{r})^2 d\mathbf{r}}} \tag{10}$$

to quantify the error or the difference E to a reference solution f_{ref} (exitance or fluence). R is the computation domain (volume or a boundary) and $\mathbf{r}$ is the integration variable (volume or a length).

4.1. Validation of the 2D code

The results of the simulation code were compared against analytical solution of the radiative transfer equation in a 2D semi-infinite layered media. The analytical solution is presented in the article by A. Liemert and A. Kienle [65]. The geometry of the problem and the optical parameters are illustrated in the inset of Fig. 3 (see also Figs. 1 and 5 of Ref. [65]). The slab consisted of two layers with different optical coefficients: the first layer was 5 mm thick while the second was infinite. The optical properties of the medium were $\mu_a = 0.08 \, \text{mm}^{-1}$, $\mu_s = 5 \, \text{mm}^{-1}$, g = 0.9 within the first layer and $\mu_a = 0.01 \, \text{mm}^{-1}$, $\mu_s = 12 \, \text{mm}^{-1}$, g = 0.9 within the second layer. A beam with a spatially Gaussian profile was incident on the slab with an incident angle θ_0 which had the values 0, 45 and 70 degrees.

The incident power per unit length through a line that is perpendicular to θ_0 is $S(y_\perp)$ where S is a Gaussian function (width $\sigma = 0.4 \, \text{mm}$) and $y_\perp$ is a position along that line. Therefore, we computed the strength of the light source (given by the field `lightsource_irradiance`) at y as $S(\cos(\theta_0)y)) \cos(\theta_0)$, where y is a position of the boundary element at the edge (x=0) of the domain. This expression follows from the geometry and preserves the beam profile in the perpendicular direction when tilted. Note that the light source type was {'direct'} and not {'gaussian'} since all photons have the same initial propagation direction.
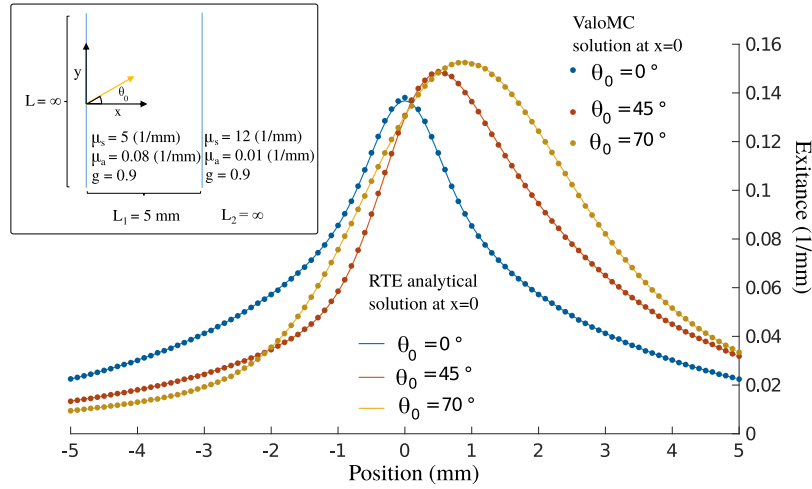
Fig. 3. Comparison of ValoMC against the analytical solution of the radiative transfer equation in a semi-infinte layered medium. The unit of exitance (1/mm) is used for consistency with Ref. [65]. The data for the analytical solution was provided by A. Liemert and A. Kienle [65].

Since infinite domains are not supported in ValoMC, the finite size error was made small by creating a suitably large computation domain with thickness of 55 mm and width of 200 mm. Element width of the rectangular mesh (`createGridMesh`) was 0.1 mm in y-direction and 1 mm in x-direction. 10^8 photon packets were launched into the medium. The relative errors of the Monte Carlo exitance computed according to Eqn. (10) in the domain $y = -5 \cdots 5$ mm were $E(0°) \approx 0.46$ %, $E(45°) \approx 0.35$ %, $E(70°) \approx 0.17$ %. The resulting exitance, given by the field `boundary_exitance` is shown in Fig. 3. The simulations demonstrate an excellent agreement with the analytical solution.

4.2. Validation of the 3D code

The performance of ValoMC was tested in a 3D cubic simulation domain and compared against MMC (mesh-based Monte Carlo) software by Q. Fang et al. [19]. The geometry of the calculation is illustrated in Fig. 4(a) (see also Fig. 3 of Ref. [19]). The domain consisted of a cube as a background (edge length 60 mm, $\mu_a = 0.005$ mm^{-1}, $\mu_s = 1$ mm^{-1}, $g = 0.01$, $n = 1.37$) with a sphere inside of the cube (radius 10 mm, $\mu_a = 0.05$ mm^{-1}, $\mu_s = 5$ mm^{-1}, $g = 0.9$, $n = 1.37$). A pencil beam was incident from the middle of the bottom face. For compatibility of the two results, all MMC simulations were run on a piecewise constant basis and performed using a large time window and step (0.5 s). The mesh had 11853 nodes and 70749 elements and it can be found as a part of the distribution of MMC.

The results were in a good agreement with each other, showing a maximal difference of 1.16×10^{-4} Wmm^{-2} in fluence after 10^8 photon packets were launched into the domain. Differences in the results calculated according to Eqn. (10) over the entire computation domain were 1.57 %, 0.16 % and 0.05 % after 10^5, 10^7 and 10^8 packets launched in to the medium, respectively. A plot of the difference is shown in Fig. 4(b). A comparison of the computation speeds is given in Section 5.2.

To validate the computation of internal reflections due to a refractive index mismatch, we varied the refractive index of the sphere. The refractive index of the background medium and at the boundary of the computation domain were kept unchanged. The total energy absorption rate into the sphere was computed with both codes using 10^7 photon packets for each refractive index
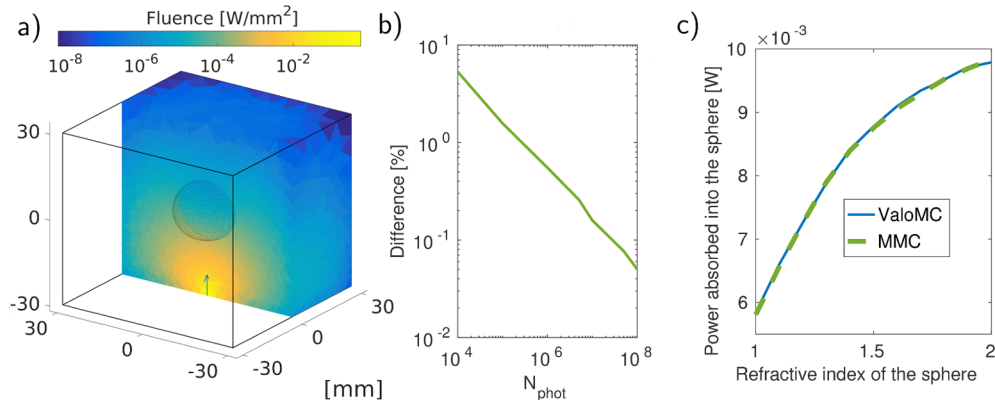
Fig. 4. Comparison of ValoMC against Mesh-based Monte Carlo (MMC) [19]. (a) Fluence distribution when the refractive index of the sphere was set to $n = 1.0$ calculated with ValoMC. (b) Difference (relative error) between fluence obtained using ValoMC and MMC with different number of photon packets calculated using Eqn. (10). (c) Total energy absorption rate in the sphere as a function of its refractive index calculated with ValoMC and MMC.

and is shown in Fig. 4(c). Changing the refractive index did not affect the difference between the results (0.16 ± 0.1 % for all computations).

4.3. Frequency domain calculation

The use of intensity modulated light sources was simulated following an example given in the book by L.V. Wang [66]. Two light sources, 180 degrees out of phase, were placed adjacent to each other (see Fig. 5) and the resulting amplitude was measured on a line further away. Due to the symmetry of the problem, the photon fluence from the two light sources should be at an opposite phase on a line in half-way between the light sources, which cancels the amplitude on that line.
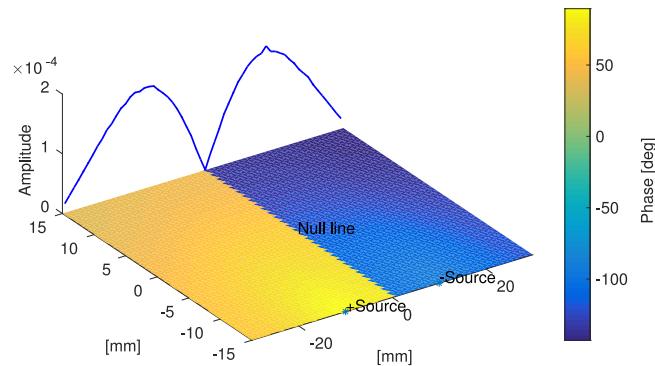

Fig. 5. Frequency domain calculation in 2D. Coloured image shows the phase of the photon fluence. The blue line shows the amplitude of the exitance.

Since the options related to the intensity modulation affect all light sources in ValoMC, two simulation runs were performed to demonstrate the effect. To activate a sinusoidally modulated light source, the field `options.frequency` was set to 200 MHz in both simulations. Using `options.phase0`, the initial phase of the first source was set to $\pi/2$ in the first simulation

and the initial phase of the second source was set to $-\pi/2$ in the second simulation. The total fluence and exitance were formed by summing the individual results from the two simulations. The results are shown in Fig. 5.

5. Software and hardware specifications and availibilty

The simulation code is written in C++ and parallelized using OpenMP. A 64-bit platform is required to run the software. ValoMC has been tested on the platforms listed in Table 3. The software is available for download from the homepage https://inverselight.github.io/ValoMC/ and the source code from GitHub https://github.com/InverseLight/ValoMC. The project is distributed under the MIT-license, excluding the external components. LICENCE file is provided for detailed information. Several examples demonstrating the various features of the code are included.

Table 3. List of the environments ValoMC has been tested on.

Operating system	MATLAB version	Compiler version
Windows 10	2017a	Visual Studio 2017
Windows 10	2016a	TDM-GCC 5.1.0
Ubuntu 16.04	2016a	GCC 4.9

5.1. Installation

Installation instructions are provided in the homepage and in the README.md file. Currently, binaries are not distributed and the program must be compiled before use. We have distributed the program to a small number of test users and found that most problems in installation arose due to MATLAB not supporting the version of the compiler that was installed. These problems are avoided e.g. by using CMake [67] to compile the program and changing the default compiler (instructions provided in the README files).

5.2. Performance

On our Linux test system (Ubuntu GCC 8.1.0, Intel(R) Core(TM) i7-6500U processor, using 4 threads), the computation times of the simulations described in Section 4.2 were somewhat slower than with MMC but similar in magnitude. ValoMC had the computation rate of about 55 000 packets/second, whereas for MMC it was 99 000 packets/second under comparable conditions.

In the current implementation, ValoMC sacrifices some of the optimisations (e.g. by using double precision numbers and standard math functions, whereas MMC uses single precision and custom math functions) for numerical accuracy. Furthermore, ValoMC (unlike MMC) stores by default the boundary solution. On the other hand, MMC performs a time domain calculation. The compiler switches were those reported optimal for MMC in its user manual. All simulations were performed with the stand-alone executable and not in MATLAB. Thus, although while we have made efforts to make the comparison as fair as possible, some differences in the computations remain. The results suggest no significant gain in accuracy by using double precision numbers and standard math functions.

Furthermore, we found that on our Windows test system, the ValoMC binary compiled with Visual Studio 2017 compiler (optimized for speed in the settings) performed considerably better than the GCC 8 compiler without losing accuracy ($\sim 1.25\times$), while we were unable to compile MMC with Visual Studio to perform the comparison also with that. For users wanting to achieve the maximal performance, we therefore recommend trying out multiple compilers. This is straightforward with CMake.

6. Discussion and conclusions

We have developed an open source Monte Carlo software for studying photon transport. While similar programs exist and are readily available, the code packages may require substantial learning efforts. To address this issue, we have also developed a simple MATLAB interface to the simulation code and provide multiple examples demonstrating the features of the software. Functions for facilitating the use of the software with certain biomedical optics simulation software such as Toast++ [61] and k-Wave [62] and meshing software such as NetGen [59] are provided. The simulation code and the interface are suitable for fast prototyping as well as setting up large-scale problems. Near-term feature extension plans include time domain computations, internal light sources (for modelling e.g. fluorescence), coordinate based detectors and optimizations. In particular, we consider that there is still considerable optimization potential left in e.g. improving cache performance, utilising the vector instructions on CPU and math libraries, while GPU implementations are also in consideration.

With this project, we aim to bring high-performance Monte Carlo modelling of light transport accessible to a wide range of researchers, teachers and students, including those not previously familiar with the method or lacking background in low-level programming languages. We vision ValoMC as a lightweight, efficient and flexible tool that is developed according to the principles of open source to suit increasingly diverse uses in biomedical and other applications.

Disclosures

The authors declare that there are no conflicts of interest related to this article.

References

1. A. Sassaroli and F. Martelli, "Equivalence of four Monte Carlo methods for photon migration in turbid media," J. Opt. Soc. Am. A **29**(10), 2110–2117 (2012).
2. C. Zhu and Q. Liu, "Review of Monte Carlo modeling of light transport in tissues," J. Biomed. Opt. **18**(5), 050902 (2013).
3. S. A. Prahl, M. Keijzer, S. L. Jacques, and A. J. Welch, "A Monte Carlo model of light propagation in tissue," in "*SPIE Proceedings of Dosimetry of Laser Radiation in Medicine and Biology*," G. Müller and D. Sliney, eds. (1989), vol. IS 5, pp. 102–111.
4. M. Hiraoka, M. Firbank, M. Essenpreis, M. Cope, S. Arridge, P. van der Zee, and D. Delpy, "A Monte Carlo investigation of optical pathlength in inhomogeneous tissue and its application to near-infrared spectroscopy," Phys. Med. Biol. **38**(12), 1859–1876 (1993).
5. L. Wang, S. Jacques, and L. Zheng, "MCML – Monte Carlo modeling of photon transport in multi-layered tissues," Comput. Methods Programs Biomed. **47**(2), 131–146 (1995).
6. L. Wang, R. Nordquist, and W. Chen, "Optimal beam size for light delivery to absorption-enhanced tumors buried in biological tissues and effect of multiple-beam delivery: a Monte Carlo study," Appl. Opt. **36**(31), 8286–8291 (1997).
7. V. Periyasamya and M. Pramanik, "Monte Carlo simulation of light transport in turbid medium with embedded object - spherical, cylindrical, ellipsoidal, or cuboidal objects embedded within multilayered tissues," J. Biomed. Opt. **19**(4), 045003 (2014).
8. T. Pfefer, J. K. Barton, E. Chan, M. Ducros, B. Sorg, T. Milner, J. Nelson, and A. Welch, "A three-dimensional modular adaptable grid numerical model for light propagation during laser irradiation of skin tissue," IEEE J. Sel. Top. Quantum Electron. **2**(4), 934–942 (1996).
9. D. Boas, J. Culver, J. Stott, and A. Dunn, "Three dimensional Monte Carlo code for photon migration through complex heterogeneous media including the adult human head," Opt. Express **10**(3), 159–170 (2002).
10. J. Heiskala, I. Nissilä, T. Neuvonen, S. Järvenpää, and E. Somersalo, "Modeling anisotropic light propagation in a realistic model of the human head," Appl. Opt. **44**(11), 2049–2057 (2005).

11. Y. Fukui, Y. Ajichi, and E. Okada, "Monte Carlo prediction of near-infrared light propagation in realistic adult and neonatal head models," Appl. Opt. **42**(16), 2881–2887 (2003).
12. Q. Fang and D. Boas, "Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units," Opt. Express **17**(22), 20178–20190 (2009).
13. S. Patwardhan, A. Dhawan, and P. Relue, "Monte Carlo simulation of light-tissue interaction: Three-dimensional simulation for trans-illumination-based imaging of skin lesions," IEEE Trans. Biomed. Eng. **52**(7), 1227–1236 (2005).
14. Y. Liu, S. Jacques, M. Azimipour, J. Rogers, R. Pashaie, and K. Eliceiri, "OptogenSIM: a 3D Monte Carlo simulation platform for light delivery design in optogenetics," Biomed. Opt. Express **6**(12), 4859–4870 (2015).
15. R. Hochuli, S. Powell, S. Arridge, and B. Cox, "Quantitative photoacoustic tomography using forward and adjoint Monte Carlo models of radiance," J. Biomed. Opt. **21**(12), 126004 (2016).
16. T. Binzoni, T. S. Leung, R. Giust, D. Rüfenacht, and A. H. Gandjbakhche, "Light transport in tissue by 3D Monte Carlo: Influence of boundary voxelization," Comput. Methods Programs Biomed. **89**(1), 14–23 (2008).
17. D. Côte and I. Vitkin, "Robust concentration determination of optically active molecules in turbid media with validated three-dimensional polarization sensitive Monte Carlo calculations," Opt. Express **13**(1), 148–163 (2005).
18. E. Margallo-Balbás and P. French, "Shape based Monte Carlo code for light transport in complex heterogeneous tissues," Opt. Express **15**(21), 14086–14098 (2007).
19. Q. Fang, "Mesh-based Monte Carlo method using fast ray-tracing in Plücker coordinates," Biomed. Opt. Express **1**(1), 165–175 (2010).
20. T. Tarvainen, V. Kolehmainen, A. Pulkkinen, M. Vauhkonen, M. Schweiger, S. Arridge, and J. Kaipio, "An approximation error approach for compensating for modelling errors between the radiative transfer equation and the diffusion approximation in diffuse optical tomography," Inv. Probl. **26**(1), 015005 (2010).
21. H. Shen and G. Wang, "A tetrahedron-based inhomogeneous Monte Carlo optical simulator," Phys. Med. Biol. **55**(4), 947–962 (2010).
22. J. Cassidy, A. Nouri, V. Betz, and L. Lilge, "High-performance, robustly verified Monte Carlo simulation with FullMonte," J. Biomed. Opt. **23**(8), 1 (2018).
23. A. Pulkkinen and T. Tarvainen, "Truncated Fourier-series approximation of the time-domain radiative transfer equation using finite elements," J. Opt. Soc. Am. A **30**(3), 470–478 (2013).
24. R. Yao, X. Intes, and Q. Fang, "Generalized mesh-based Monte Carlo for widefield illumination and detection via mesh retessellation," Biomed. Opt. Express **7**(1), 171–184 (2016).
25. C. Hayakawa, J. Spanier, and V. Venugopalan, "Comparative analysis of discrete and continuous absorption weighting estimators used in Monte Carlo simulations of radiative transport in turbid media," J. Opt. Soc. Am. A **31**(2), 301–311 (2014).
26. T. Leung and S. Powell, "Fast Monte Carlo simulations of ultrasound-modulated light using a graphics processing unit," J. Biomed. Opt. **15**(5), 055007 (2010).
27. Q. Fang and D. Kaeli, "Accelerating mesh-based Monte Carlo method on modern CPU architectures," Biomed. Opt. Express **3**(12), 3223–3230 (2012).
28. S. Powell and T. Leung, "Highly parallel Monte-Carlo simulations of the acousto-optic effect in heterogeneous turbid media," J. Biomed. Opt. **17**(4), 045002 (2012).
29. O. Yang and B. Choi, "Accelerated rescaling of single Monte Carlo simulation runs with the Graphics Processing Unit (GPU)," Biomed. Opt. Express **4**(11), 2667–2672 (2013).
30. L. Yu, F. Nina-Paravecino, D. Kaeli, and Q. Fang, "Scalable and massively parallel Monte Carlo photon transport simulations for heterogeneous computing platforms," J. Biomed. Opt. **23**(1), 1 (2018).
31. Y. Liu, H. Jiang, and Z. Yuan, "Two schemes for quantitative photoacoustic tomography based on Monte Carlo simulation," Med. Phys. **43**(7), 3987–3997 (2016).
32. S. L. Jacques, "Coupling 3D Monte Carlo light transport in optically heterogeneous tissues to photoacoustic signal generation," Photoacoustics **2**(4), 137–142 (2014).
33. D. Ancora, A. Zacharopoulos, J. Ripoll, and G. Zacharakis, "Fluorescence diffusion in the presence of optically clear tissues in a mouse head model," IEEE Trans. Med. Imaging. **36**(5), 1086–1093 (2017).
34. J. Chen, V. Venugopal, and X. Intes, "Monte Carlo based method for fluorescence tomographic imaging with lifetime multiplexing using time gates," Biomed. Opt. Express **2**(4), 871–886 (2011).
35. J. J. Selb, D. A. Boas, S. T. Chan, K. C. Evans, E. M. Buckley, and S. A. Carp, "Sensitivity of near-infrared spectroscopy and diffuse correlation spectroscopy to brain hemodynamics: simulations and experimental findings during hypercapnia," Neurophotonics **1**(1), 015005 (2014).
36. D. Ancora, L. Qiu, G. Zacharakis, L. Spinelli, A. Torricelli, and A. Pifferi, "Noninvasive optical estimation of CSF thickness for brain-atrophy monitoring," Biomed. Opt. Express **9**(9), 4094 (2018).
37. F. Martelli, T. Binzoni, A. Pifferi, L. Spinelli, A. Farina, and A. Torricelli, "There's plenty of light at the bottom: statistics of photon penetration depth in random media," Sci. Rep. **6**(1), 27057 (2016).
38. D. Wangpraseurt, S. L. Jacques, T. Petrie, and M. Kühl, "Monte Carlo modeling of photon propagation reveals highly scattering coral tissue," Front. Plant. Sci. **7**, 1404 (2016).
39. J. Cassidy, V. Betz, and L. Lilgem, "Treatment plan evaluation for interstitial photodynamic therapy in a mouse model by Monte Carlo simulation with FullMonte," Front. Phys. **3**, 6 (2015).

40. H. Xiang, B. Chen, W. Wu, Y. Zhang, and H. Jia, "An integral MPS model of blood coagulation by laser irradiation: Application to the optimization of multi-pulse Nd:YAG laser treatment of port-wine stains," Int. J. Heat Mass Transfer **114**, 1220–1233 (2017).

41. J. Chen and X. Intes, "Comparison of Monte Carlo methods for fluorescence molecular tomography-computational efficiency," Phys. Med. Biol. **38**, 5788–5798 (2011).

42. Y. Marzouk, I. Langmore, and G. Bal, "Bayesian inverse problems with Monte Carlo forward models," Inv. Probl. Imag. **7**(1), 81–105 (2013).

43. L. Vinckenbosch, C. Lacaux, S. Tindel, M. Thomassin, and T. Obara, "Monte Carlo methods for lightpropagation in biological tissues," Math. Biosci. **269**, 48–60 (2015).

44. B. A. Kaplan, J. Buchmann, S. Prohaska, and J. Laufer, "Monte-Carlo-based inversion scheme for 3D quantitative photoacoustic tomography," in "*Photons Plus Ultrasound: Imaging and Sensing 2017, Proc. of SPIE*," A. Oraevsky and L. Wang, eds. (2017), vol. 10064, pp. 100645J–1.

45. J. Buchmann, B. A. Kaplan, S. Prohaska, and J. Laufer, "Experimental validation of a Monte-Carlo-based inversion scheme for 3D quantitative photoacoustic tomography," in "*Photons Plus Ultrasound: Imaging and Sensing 2017, Proc. of SPIE*," A. Oraevsky and L. Wang, eds. (2017), vol. 10064, p. 1006416.

46. A. Correia, P. Hanselaer, H. Cornelissen, and Y. Meuret, "Radiance based method for accurate determination of volume scattering parameters using GPU-accelerated Monte Carlo," Opt. Express **25**(19), 22575–22586 (2017).

47. O. Lehtikangas, T. Tarvainen, V. Kolehmainen, A. Pulkkinen, S. Arridge, and J. Kaipio, "Finite element approximation of the Fokker-Planck equation for diffuse optical tomography," J. Quant. Spectrosc. Radiat. Transfer **111**(10), 1406–1417 (2010).

48. P. Mohan, T. Tarvainen, M. Schweiger, A. Pulkkinen, and S. Arridge, "Variable order spherical harmonic expansion scheme for the radiative transport equation using finite elements," J. Comput. Phys. **230**(19), 7364–7383 (2011).

49. T. Tarvainen, A. Pulkkinen, B. Cox, J. Kaipio, and S. Arridge, "Image reconstruction in quantitative photoacoustic tomography using the radiative transfer equation and the diffusion approximation," in "*Opto-Acoustic Methods and Applications, Proc. of OSA Biomedical Optics-SPIE*," V. Ntziachristos and C. Lin, eds. (2013), vol. 8800, pp. 880006–1.

50. A. Pulkkinen, V. Kolehmainen, J. Kaipio, B. Cox, S. Arridge, and T. Tarvainen, "Approximate marginalization of unknown scattering in quantitative photoacoustic tomography," Inv. Probl. Imag. **8**(3), 811–829 (2014).

51. O. Lehtikangas, T. Tarvainen, A. Kim, and S. Arridge, "Finite element approximation of the radiative transport equation in a medium with piece-wise constant refractive index," J. Comput. Phys. **282**, 345–359 (2015).

52. V. Periyasamy and M. Pramanik, "Advances in Monte Carlo simulation for light propagation in tissue," IEEE Rev. Biomed. Eng. **10**, 122–135 (2017).

53. A. Ishimaru, *Wave Propagation and Scattering in Random Media*, vol. 1 (Academic, 1978).

54. L. G. Henyey and J. L. Greenstein, "Diffuse radiation in the galaxy," Astrophys. J. **93**, 70–83 (1941).

55. A. Kienle, F. K. Foster, and R. Hibst, "Influence of the phase function on determination of the optical properties of biological tissue by spatially resolved reflectance," Opt. Lett. **26**(20), 1571–1573 (2001).

56. M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," ACM Trans. Model. Comput. Simul. **8**(1), 3–30 (1998).

57. L. Devroye, "General Principles in Random Variate Generation," in "*Non-Uniform Random Variate Generation*," (Springer New York, New York, 1986), pp. 27–82.

58. Q. Fang and D. A. Boas, "Tetrahedral mesh generation from volumetric binary and gray-scale images," in "*Proc. IEEE Int. Symp. Biomed. Imaging*," (IEEE Press, 2009), ISBI'09, pp. 1142–1145.

59. J. Schöberl, "NETGEN an advancing front 2d/3d-mesh generator based on abstract rules," Comput. Visualization Sci. **1**(1), 41–52 (1997).

60. "Comsol Multiphysics, COMSOL AB, Stockholm, Sweden," http://www.comsol.com/.

61. M. Schweiger and S. R. Arridge, "The Toast++ software suite for forward and inverse modeling in optical tomography," J. Biomed. Opt. **19**(4), 040801 (2014).

62. B. Treeby and B. Cox, "k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields," J. Biomed. Opt. **15**(2), 021314 (2010).

63. Visual Computing Lab ISTI - CNR, "MeshLab," http://meshlab.sourceforge.net/.

64. A. Jacobson, *et al.*, "gptoolbox: Geometry processing toolbox," (2018). http://github.com/alecjacobson/gptoolbox.

65. A. Liemert and A. Kienle, "Analytical approach for solving the radiative transfer equation in two-dimensional layered media," J. Quant. Spectrosc. Radiat. Transfer **113**(7), 559–564 (2012).

66. L. V. Wang, *Diffuse Optical Tomography* (Wiley-Blackwell, 2012), chap. 11, pp. 249–281.

67. Kitware, Inc., "CMake," http://cmake.org (2018).