Journal of Biomedical Optics

BiomedicalOptics.SPIEDigitalLibrary.org

MCmatlab: an open-source, user-friendly, MATLAB-integrated three-dimensional Monte Carlo light transport solver with heat diffusion and tissue damage

Dominik Marti
Rikke N. Aasbjerg
Peter E. Andersen
Anders K. Hansen

SPIE.

Dominik Marti, Rikke N. Aasbjerg, Peter E. Andersen, Anders K. Hansen, "MCmatlab: an open-source, user-friendly, MATLAB-integrated three-dimensional Monte Carlo light transport solver with heat diffusion and tissue damage," *J. Biomed. Opt.* **23**(12), 121622 (2021), doi: 10.1117/1.JBO.23.12.121622.

Journal of Biomedical Optics 23(12), 121622 (December 2018)

MCmatlab: an open-source, user-friendly, MATLAB-integrated three-dimensional Monte Carlo light transport solver with heat diffusion and tissue damage

Dominik Marti, Rikke N. Aasbjerg, Peter E. Andersen, and Anders K. Hansen*
Technical University of Denmark, Department of Photonics Engineering, Roskilde, Denmark

Abstract. While there exist many Monte Carlo (MC) programs for solving the radiative transfer equation (RTE) in biological tissues, we have identified a need for an open-source MC program that is sufficiently user-friendly for use in an education environment, in which detailed knowledge of compiling or UNIX command-line cannot be assumed. Therefore, we introduce MCmatlab, an open-source codebase thus far consisting of (a) a fast three-dimensional MC RTE solver and (b) a finite-element heat diffusion and Arrhenius-based thermal tissue damage simulator, both run in MATLAB. The kernel for both of these solvers is written in parallelized C and implemented as MATLAB MEX functions, combining the speed of C with the familiarity and versatility of MATLAB. We compare the RTE solver to Steven Jacques' mcxyz, which it is inspired by, and present example results generated by the thermal model. MCmatlab is easy to install and use and can be used by students and experienced researchers alike for simulating tissue light propagation and, optionally, thermal damage. [DOI: 10.1117/1.JBO.23.12.121622]

Keywords: Monte Carlo methods; photothermal effects; three-dimensional modeling; tissue optics; scattering; absorption.

Paper 180443SSRR received Jul. 10, 2018; accepted for publication Nov. 20, 2018; published online Dec. 15, 2018; corrected Jan. 19, 2021.

1 Introduction

When modeling light interaction with biological tissue, the first step is typically to calculate the distribution of light within the tissue, given the optical properties for a given illumination. This light distribution is described by the solution to the radiative transfer equation (RTE), which is often solved with Monte Carlo (MC) methods,[1,2] in which the light is simulated as photon packets that are gradually absorbed as they travel through the medium and will randomly undergo scattering at a rate dependent on the local optical properties of the medium.

Due to the conceptually straightforward nature of the MC methods of solving the RTE, especially using rectangular cuboid voxel meshes,[3–7] many implementations have been made with various strengths and weaknesses and using various meshing schemes. Some solvers assume certain geometrical symmetries (usually cylindrical)[8] or make other numerical approximations[9] to speed up the calculations, which may otherwise in some cases be prohibitively time-consuming. Over the years, many advanced methods have been demonstrated simulating effects such as anisotropic light propagation,[10] light polarization,[11,12] and fluorescence.[13] The simulation speed has been increased using CPU[14] and GPU[3,7,15,16] parallelization and polyhedral meshing of the geometry.[10,17] Various RTE solvers, including Steven Jacques' mcxyz,[4] have been coupled to photoacoustics[18] and other physical phenomena.[19]

However, most of these programs were not written with user-friendliness in mind or are not free of charge or open-source, and some are outdated and can no longer be compiled or run on modern PCs. Some programs use MATLAB or Octave as the user interface of choice[3,16,17] but most were implemented in low-level programming languages, such as C or C++ to make them very fast, but this usually also makes them difficult to integrate into a larger framework, such as batch execution with programmatical pre- and postprocessing, especially for researchers, engineers, or students who are not themselves experienced C or C++ programmers. Additionally, many implementations are compilable and executable only on a UNIX system or through a UNIX terminal on a Windows or Mac system or rely on having a CUDA-enabled GPU and the corresponding development libraries installed, which in turn requires specialized knowledge on the user side to set up and compile. From our own experience in teaching students on the bachelor and master level and in collaborating with other researchers not familiar with MC methods or software compilation, we have identified a need for an easy to use MC code that does not need to be the fastest, most versatile, or most compatible to other specialized software tools but instead is easy to install, run, and use by nonexperts in programming.

The MATLAB computing environment, although not itself free of charge, is nevertheless available and familiar to many students and researchers worldwide and offers a wealth of functions and interfaces for setting up, visualizing, and analyzing data. Therefore, we have started the project MCmatlab, a codebase that thus far consists of an MC RTE solver inspired by mcxyz and a finite-element thermal diffusion and thermal damage solver, both implemented as MATLAB MEX functions. MEX functions are written in C, C++, or Fortran and, once compiled, can be called in MATLAB as ordinary functions that combine the high speed of those low-level languages with the versatility and data-interfacing of MATLAB. MCmatlab comes

*Address all correspondence to Anders K. Hansen, E-mail: ankrh@fotonik.dtu.dk

with precompiled MEX functions verified to work with recent versions of MATLAB. The code has been written in C in such a way that, if recompiling should prove necessary, it can be done completely within MATLAB in only two steps.

The software is open-source and available at gitlab.gbar.dtu.dk/biophotonics/MCmatlab.

2 MCmatlab's Monte Carlo Solver for Radiative Transfer

The distribution of light within the tissue is found by solving the RTE. MCmatlabs RTE solver is based on and still follows at its core the method of the program mcxyz, developed by Jacques et al.[4] at the Oregon Medical Laser Center. The three main differences for the user are that MCmatlab is entirely controlled through MATLAB, that its RTE solver offers a wider choice of illumination patterns, and that the results are shown using interactive three-dimensional (3-D) volumetric slice plotting. The MCmatlab RTE solver also offers a 17-fold speed increase in comparison with mcxyz.

2.1 *Similarities to mcxyz*

MCmatlab, in the same way as mcxyz, operates in Cartesian coordinates and makes no assumptions as to symmetry in the simulated volume. Its boundary is a rectangular cuboid and its internal volume is uniformly divided into voxels, which are themselves also rectangular cuboids. The user assigns to each voxel a medium or tissue type, described by its absorption coefficient μ_a, its scattering coefficient μ_s, and its Henyey–Greenstein scattering anisotropy factor g at the applied wavelength.

An input light beam is simulated by launching photon packets and calculating their paths in the simulated volume, using pseudorandomly generated numbers to determine initial photon packet position and trajectory as well as path lengths between scattering events and scattering angles. As photon packets propagate from one voxel to the next, they deposit some of their energy ("weight") into the voxel depending on the voxel's absorption coefficient μ_a. This deposited energy is numerically accumulated in a 3-D matrix, which, upon conclusion of the simulation is normalized to the input power, providing the local absorption rate per watt of incident light, in units of $\mathrm{W/cm^3/W.incident}$. Dividing this by the voxels' absorption coefficients yields $F(x, y, z)$, the local fluence rate (irradiance or intensity) per watt of incident light, in units of $\mathrm{W/cm^2/W.incident}$.

2.2 *Differences to mcxyz*

There are three main differences between MCmatlab and mcxyz (as of mcxyz version June 1, 2017), mostly centered around improved usability for nonexperts in programming. The most apparent of these is the rewriting of the C code to be compilable and callable in Windows or on Mac from MATLAB as a MEX function, combining the speed of C with the flexibility and ease-of-use of MATLAB. This means that one can perform the entire simulation initialization, execution, postprocessing, visualization, and further interfacing in MATLAB. Therefore, users on Windows or Mac no longer need, e.g., a separate UNIX-emulating console shell to run a compiler and the MC executable, nor is there a need for any intermediate input/output files, since all data are passed in and out of the MEX function through memory. The simulation inputs and the fluence rate matrix output (the solution to the radiative transport equation) are saved as MATLAB ".mat" files, a compressed binary file format allowing for flexible and easy saving and loading and ensuring smaller file sizes compared to uncompressed binary or plain-text file formats.

The second main difference to the mcxyz code is the introduction of a wider variety of beam type definitions. The full selection of beam types is currently pencil beams; isotropically emitting points; infinite plane waves; Gaussian focus, Gaussian far-field beams; Gaussian focus, top-hat far-field beams; top-hat focus, Gaussian far-field beams; top-hat focus, top-hat far-field beams; and Laguerre–Gaussian LG01 beams.

The focal plane of the beams is defined as the plane containing the user-specified focus point with a normal vector given by the user-specified beam center trajectory. For the beams with Gaussian and/or top-hat beam profiles, the initial position and trajectory of each photon are calculated by first randomly sampling, according to the chosen focus beam profile, Gaussian, or top-hat, the point in the focal plane that the photon would hit in the absence of scattering and then randomly sampling the initial trajectory according to the chosen far-field beam profile. The initial position of the photon is then defined as the intersection between the $z = 0$ plane and the line going through the focal plane target point along the photon's initial trajectory. For the Laguerre–Gaussian LG01 beam, the sampling is slightly different; for a focal plane target point at a position r relative to the focus point, the trajectory is chosen only among those directions that are orthogonal to r. This ensures that the intensity is zero everywhere along the center axis of the beam, as expected of an LG01 beam. The focus and far-field widths can be freely specified and could, for instance, be chosen to match that of a diffraction-limited beam.

The third major difference is the visualizations. It can be challenging to properly visualize data in a 3-D volume, especially if the volume contains fine-structured heterogeneities in multiple layers. Therefore, MCmatlab uses interactive 3-D slice plotting, in which the color scale can be switched between linear and logarithmic by checking a box.

Additional changes from mcxyz include: (a) changing pseudorandom number generator from Donald Knuth's subtractive algorithm from 1969[20] to the Fast Mersenne Twister algorithm from 2008,[21] (b) a change of coordinate system from yxz-coordinates to xyz-coordinates, (c) CPU multithread parallelization of the simulation on Windows using OpenMP, (d) minor bug fixes, and (e) significant general speed optimizations.

MCmatlab can also optionally simulate refraction and Fresnel reflection at interfaces between media with different refractive index, provided that the geometry is oriented so that the interface is parallel to the xy-plane. Since MCmatlab does not track polarization, the reflection probabilities are calculated assuming that the light is unpolarized.

2.3 *Example and Comparison of Results*

To test whether the results of our MC RTE solver agrees with prior work, we compared MCmatlab with mcxyz using the example shown on the mcxyz website,[4] a $200 \times 200 \times 200$ voxel model of a cylindrical blood vessel of radius 100 μm embedded 400 μm deep in dermis, with a 60-μm layer of epidermis on the surface. Figure 1 shows the geometry, visualized using the interactive 3-D volumetric slice plotting used for many different plots in MCmatlab. Table 1 shows the parameters used

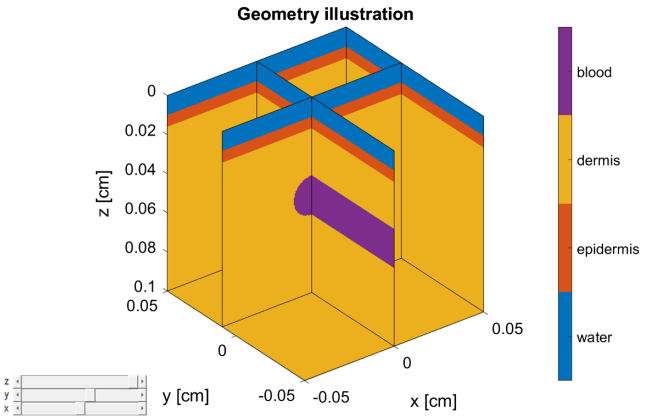

Fig. 1 Screenshot of MCmatlab's illustration of the simulation geometry definition with slices shown at $x = 0.05$ cm, $x = 0$ cm, $y = 0.05$ cm, $y = 0.01$ cm, and $z = 0.1$ cm.

for the demonstration. For comparison purposes, all μ_a, μ_s, and g values were set equal to those used in the mcxyz demonstration example. The thermal properties are used only for the thermal simulations, as shown in Sec. 3. Note that the values are not necessarily accurate for real tissue but are for demonstration purposes only. The illumination beam was a collimated top-hat beam of radius 300 μm.

For this comparison, 6.36×10^7 photon packets were launched in both programs. In Fig. 2, the results of the MCmatlab simulations are shown in terms of the fluence rate and the absorbed power per unit volume. Figure 3 shows a contour-plot comparison to the results of mcxyz in the $y = 0$ and $x = 0$ planes. It can be seen from Fig. 3 that there is good agreement between the results of mcxyz and MCmatlab. There is a bug in mcxyz that affects fluence rates in voxels bordering some of the cuboid boundaries, but this does not affect all inner voxels.

Running in a UNIX-emulating Cygwin terminal on a mid-range laptop PC from 2012 (Dell Latitude E6530) with 64-bit Windows, mcxyz launched 6.29×10^4 photons per minute. When run in MATLAB on the same PC, MCmatlab launched 1.09×10^6 photon packets per minute (a factor of 17 more). MCmatlab's multithreading to the laptop's 8 CPU processors accounts for roughly a factor of 4.5 of the speed increase compared to mcxyz, whereas the remaining factor of 4 is due partly to switching to the Fast Mersenne Twister pseudorandom number generator and partly to many small optimizations throughout the code base.

3 MCmatlab Heat Deposition, Diffusion, and Arrhenius-Based Thermal Damage Solver

As an optional add-on to the MC RTE solver described in Sec. 2, MCmatlab includes a 3-D finite-element time-domain heat deposition and diffusion solver, operating on the same voxel grid as the MC RTE solver. During the simulation, Arrhenius-based thermal chemical changes, such as tissue coagulation, are also calculated. Like the MC solver, the thermal solver is also implemented in C as a MATLAB-compilable and MATLAB-callable MEX function, and, thus, also benefits from combining the speed of C with the flexibility of MATLAB.

Table 1 An overview of the optical properties, the physical thermal properties, and the chemical thermal properties of the four tissues/media used in the example. The values are for demonstration purposes only.

	μ_a (cm^{-1})	μ_s (cm^{-1})	g	c_V (J/cm^3 K)	k (W/(cm K))	E_a (J/mol)	A (s^{-1})
Water	0.00036	10.0	1.00	4.19	0.0058	—	—
Epidermis	16.6	376	0.90	3.76	0.0037	—	—
Dermis	0.459	357	0.90	3.76	0.0037	—	—
Blood	231	94.0	0.90	3.80	0.0052	4.23×10^5	7.6×10^{66}

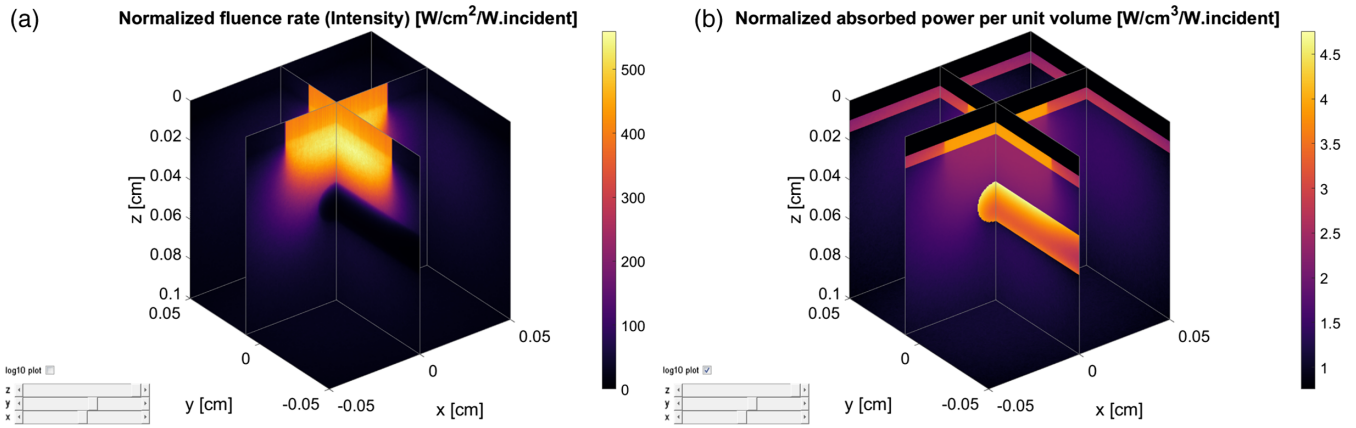

Fig. 2 Screenshots of MCmatlab's illustration of (a) the fluence rate and (b) the logarithm of the absorbed power per unit volume. Slices are shown at the same positions as in Fig. 1.

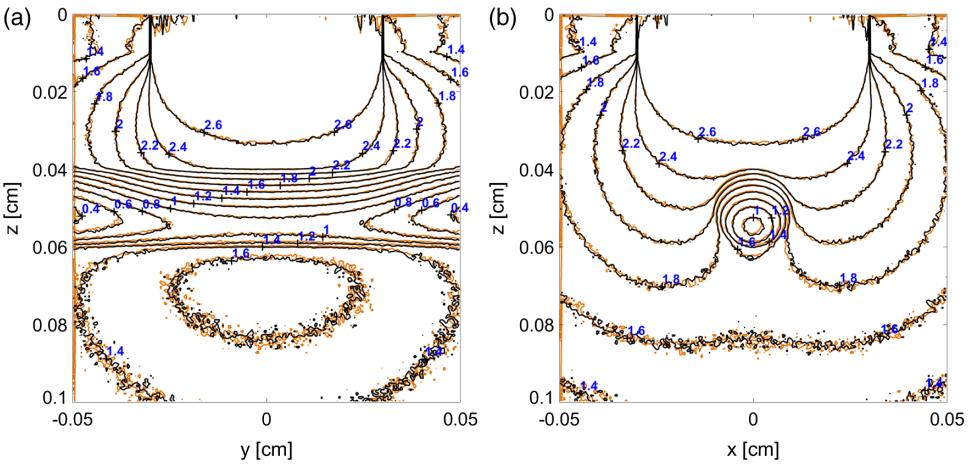

Fig. 3 A comparison of the values of the logarithm of the fluence rate calculated by MCmatlab and mcxyz after launching 6.36×10^7 photon packets. The geometry definition is shown in Fig 1. Black contour curves: MCmatlab, orange: mcxyz. (a) $x = 0$ and (b) $y = 0$.

3.1 *Theory*

The heat deposition and diffusion solver calculates the time-resolved temperature distribution in the simulation volume based on the fluence rate F and the geometry definition of the MC simulation, this time with additional material parameters; the media's physical thermal properties c_V and k and, optionally, chemical thermal properties E_a and A of the considered chemical reaction. Although the physical thermal parameters must be specified for all involved media, the chemical properties may be specified for all, some, or none of them.

The temperature evolution can be simulated for continuous light exposure or for single- or multipulse light exposure with user-specified duty cycle and period. At the end of the simulation, a plot is shown that illustrates the spatial distribution of chemically changed media in the simulation volume. A video can also be automatically generated showing the temperature evolution.

Denoting the local volumetric heat capacity by c_V and thermal conductivity by k, the heat equation in continuous form is given as

$$\frac{\partial T}{\partial t} = \frac{q + \nabla \cdot (k \nabla T)}{c_V},$$

where q is the local rate of heat deposition, which is calculated from multiplying the voxel's fluence rate with its absorption coefficient. The equation is solved with an explicit finite-element method, where the temperature change ΔT after a small time step Δt for the voxel at spatial position indices (i_x, i_y, i_z) is calculated as

$$\begin{aligned}\Delta T = \frac{\Delta t}{c_V}\Bigg(& P\,\mu_a F + (T_{x^-} - T)\frac{2\,k k_{x^-}}{k + k_{x^-}}\frac{1}{dx^2} \\ & + (T_{x^+} - T)\frac{2\,k\,k_{x^+}}{k + k_{x^+}}\frac{1}{dx^2} + (T_{y^-} - T)\frac{2\,k k_{y^-}}{k + k_{y^-}}\frac{1}{dy^2} \\ & + (T_{y^+} - T)\frac{2\,k k_{y^+}}{k + k_{y^+}}\frac{1}{dy^2} + (T_{z^-} - T)\frac{2\,k k_{z^-}}{k + k_{z^-}}\frac{1}{dz^2} \\ & + (T_{z^+} - T)\frac{2\,k k_{z^+}}{k + k_{z^+}}\frac{1}{dz^2}\Bigg),\end{aligned}$$

where P is the incident power, (dx, dy, dz) are the voxel side lengths, k is the thermal conductivity of a voxel, T is its temperature, and the subscripts designate neighboring voxels in the negative and positive x, y, and z directions. The user can specify whether the solution should be calculated by assuming $\Delta T = 0$ on all boundaries, corresponding to constant-temperature (heat-sinked) boundaries, or by simply omitting the above terms that contain temperatures outside the volume, corresponding to insulated boundaries. In the current version of MCmatlab, effects such as cooling by blood perfusion are neglected, as well as any temperature dependence of the optical and thermal properties.

The thermal chemical change is quantified for each voxel in terms of the dimensionless Arrhenius Ω value, which starts at zero and increases over time as the voxel is exposed to elevated temperatures T, depending on the Arrhenius activation energy E_a and Arrhenius pre-exponential factor A:

$$\begin{aligned}\Omega(x, y, z) &= \ln\left(\frac{c_0(x, y, z)}{c_\tau(x, y, z)}\right) \\ &= A\int_0^\tau \exp\left(\frac{-E_a}{R \cdot T(x, y, z, t)}\right)\mathrm{d}t,\end{aligned}$$

where c_0 is the concentration of undamaged molecules at the simulation start, c_τ is the concentration of undamaged molecules at the simulation end, R is the gas constant, t is the simulation time, and τ is the total simulated time duration. Those voxels that have $\Omega > 1$ at the end of the simulation are considered to be "damaged," that is, more than $1 - \frac{1}{e} = 63\%$ of the molecules having undergone a chemical reaction, such as coagulation.

3.2 *Example*

In this demonstration, the blood vessel defined in Sec. 2 was used as an input to the thermal solver, setting the input power to 4 W, running for 5 light pulses of 1 ms on-time and 4 ms off-time each. Video 1 is provided as supplementary material, and a still image is shown here as Fig. 4. The resulting illustration of damaged tissue is shown in Fig. 5.

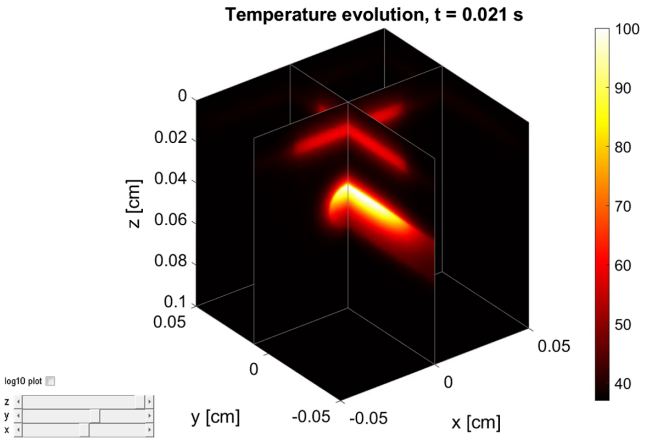

Fig. 4 Still image from a video showing the temperature evolution during five pulses of light incident on the tissue defined in Sec. 2 (Video 1, MP4, 360 KB [URL: https://doi.org/10.1117/1.JBO.23.12.121622.1]).

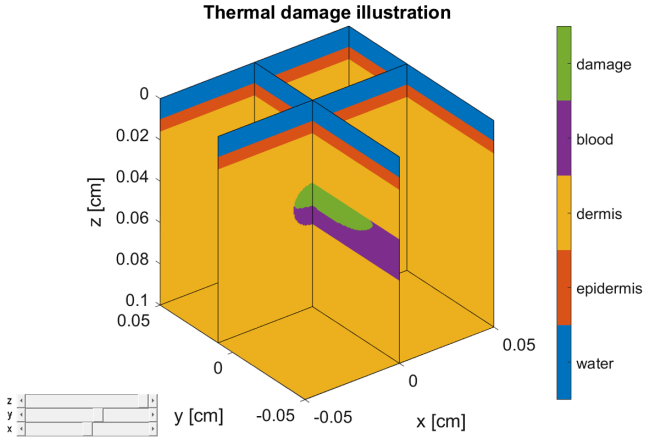

Fig. 5 Screenshot of MCmatlab's thermal damage illustration. The volume of coagulated blood is shown in green.

4 Conclusions

Open-source MC programs, such as mcxyz, are invaluable as tools for researchers and students to model light–tissue interaction and to help understand some of the underlying physics. With MCmatlab, we have taken the core concepts of mcxyz and integrated them into open-source MATLAB MEX functions, making it easier for researchers and students who are not experienced C programmers or users of UNIX systems to join in and use these tools.

The MCmatlab codebase currently consists of the RTE solver for finding the light distribution in complex turbid media and a thermal solver, useful for simulating, e.g., photocoagulation. It is designed to be user-friendly and computationally fast, its RTE solver being roughly 17 times faster than mcxyz.

We expect that students and experienced researchers alike will benefit from this suite of software for both research and teaching activities, and we openly share the code with everyone, encouraging others to add to and modify the code as they wish.

Disclosures

The authors have no relevant financial interests in the article and no other potential conflicts of interest to disclose.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 667933.

References

1. B. C. Wilson and G. Adam, "A Monte Carlo model for the absorption and flux distributions of light in tissue," *Med. Phys.* **10**(6), 824–830 (1983).
2. S. T. Flock et al., "Monte Carlo modeling of light propagation in highly scattering tissues. I. Model predictions and comparison with diffusion theory," *IEEE Trans. Biomed. Eng.* **36**(12), 1162–1168 (1989).
3. Q. Fang and D. A. Boas, "Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units," *Opt. Express* **17**(22), 20178–20190 (2009).
4. S. Jacques, T. Li, and S. Prahl, "mcxyz.c, a 3D Monte Carlo simulation of heterogeneous tissues," omlc.org/software/mc/mcxyz (2017).
5. D. A. Boas et al., "Three dimensional Monte Carlo code for photon migration through complex heterogeneous media including the adult human head," *Opt. Express* **10**(3), 159–170 (2002).
6. I. Kawrakow, M. Fippel, and K. Friedrich, "3D electron dose calculation using a voxel based Monte Carlo algorithm (VMC)," *Med. Phys.* **23**(4), 445–457 (1996).
7. A. Bjorgan, M. Milanic, and L. L. Randeberg, "YMC3D: GPU-accelerated 3D Monte Carlo photon tracking in tissue inclusions," https://github.com/ntnu-bioopt/ymc3d (3 October 2018).
8. S. L. Jacques and L. Wang, "Monte Carlo modeling of light transport in tissues," in *Optical-Thermal Response of Laser-Irradiated Tissue*, Vol. **47**, No. 2, pp. 73–100, Springer US, Boston, Massachusetts (1995).
9. L. Wang and S. L. Jacques, "Hybrid model of Monte Carlo simulation and diffusion theory for light reflectance by turbid media," *J. Opt. Soc. Am. A* **10**(8), 1746–1752 (1993).
10. C. J. Zoller et al., "Parallelized Monte Carlo software to efficiently simulate the light propagation in arbitrarily shaped objects and aligned scattering media," *J. Biomed. Opt.* **23**(6), 065004 (2018).
11. G. W. Kattawar and G. N. Plass, "Radiance and polarization of multiple scattered light from haze and clouds," *Appl. Opt.* **7**(8), 1519–1527 (1968).
12. J. C. Ramella-Roman, S. A. Prahl, and S. L. Jacques, "Three Monte Carlo programs of polarized light transport into scattering media: part I," *Opt. Express* **13**(12), 4420–4438 (2005).
13. A. J. Welch et al., "Propagation of fluorescent light," *Lasers Surg. Med.* **21**(2), 166–178 (1997).
14. A. Colasanti et al., "Multiple processor version of a Monte Carlo code for photon transport in turbid media," *Comput. Phys. Commun.* **132**(1–2), 84–93 (2000).
15. E. Alerstam, T. Svensson, and S. Andersson-Engels, "Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration," *J. Biomed. Opt.* **13**(6), 060504 (2008).
16. 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).
17. A. Leino, A. Pulkkinen, and T. Tarvainen, "InverseLight/ValoMC: Monte Carlo software for simulating light propagation," 2018, https://github.com/InverseLight/ValoMC (14 November 2018).
18. S. L. Jacques, "Coupling 3D Monte Carlo light transport in optically heterogeneous tissues to photoacoustic signal generation," *Photoacoustics* **2**(4), 137–142 (2014).
19. Y. Liu et al., "OptogenSIM: a 3D Monte Carlo simulation platform for light delivery design in optogenetics," *Biomed. Opt. Express* **6**(12), 4859–4870 (2015).
20. D. E. Knuth, *The Art of Computer Programming, Volume II: Seminumerical Algorithms*, 1st ed., Addison-Wesley Massachusetts (1969).
21. M. Saito and M. Matsumoto, "SIMD-oriented Fast Mersenne Twister: a 128-bit pseudorandom number generator," in *Monte Carlo Quasi-Monte Carlo Methods 2006*, pp. 607–622, Springer, Berlin, Heidelberg (2008).

Dominik Marti received his MSc and PhD degrees in two-photon fluorescence technologies from the University of Bern, Switzerland. He then joined DTU Fotonik, Denmark, as a researcher in the field of biomedical imaging with a focus on multiphoton microscopy, OCT/OCM, and multimodal imaging, and the translation thereof to clinics.

Rikke N. Aasbjerg received her MSc degree from the Technical University of Denmark in 2018, using Monte Carlo methods to simulate photocoagulation treatment of diabetic eye diseases.

Peter E. Andersen received his MSc and PhD degrees from the Technical University of Denmark, Denmark, 1991 and 1994, respectively. His research interests include laser systems, optical coherence tomography, nonlinear microscopy, and multimodal imaging for biomedical applications.

Anders K. Hansen received his PhD from Aarhus University, Denmark, in 2013 in the field of laser cooling of atomic and molecular ions and has since worked at the Technical University of Denmark on development and applications of laser systems, especially diode lasers and especially within biophotonics. He also works on numerical methods for optics simulations and phase retrieval.