TISSUE OPTICS 2011

Monte Carlo Simulations of Light Transport in Tissue

Computer Exercise

Erik Alerstam
Stefan Andersson-Engels

Department of Physics, Lund

March 21, 2011

This document contains instructions for the computer exercise on Monte Carlo simulations of light transport in scattering media. We welcome comments that may improve this document. Contact us at:

`erik.alerstam@fysik.lth.se`
`stefan.andersson-engels@fysik.lth.se`

Contents

1 Introduction

The aim of this exercise is to make you familiar with Monte Carlo simulations of light transport in scattering media. The Monte Carlo method is an approach used to solve forward modeling problems in many different fields of physics, for example neutron transport in nuclear physics. Here it is used to simulate how photons are transported and absorbed in turbid media. The advantage of the Monte Carlo method is that it is a reasonably simple model that can handle arbitrary geometries and that it can provide a direct solution without approximations to the radiative transport equation (RTE). The downside, which you will experience hands on in this exercise, is the simulation time required which makes it less suitable for inverse problems (with a few exceptions, ask your instructor about this if you are interested). The Monte Carlo method is also limited by noise and the fact that the exact geometry rarely is known. Despite the disadvantages the Monte Carlo method is considered the "gold standard" in modeling of light propagation within the field of Biomedical optics.

2 Theory

2.1 Monte Carlo basics

Monte Carlo, is as the name implies based on "throwing the dice", in the sense that individual photon packets are traced as they propagate through a turbid material. The path of each photon packet is determined by the sampling of (pseudo) random numbers and a set of functions or probability distributions describing the likeliness of for example the step length and scattering angles. This way of stochastic simulation relies on the simulation of a large number of photon packets and the solutions provided are more or less noisy depending on the number of photons simulated.

2.2 MCML & conv

During this exercise you will be using an open source Monte Carlo simulation package provided by Dr. Lihong Wang and Prof. Steven L. Jacques. The software (both executables and source code) along with a thorough manual is available at:

`http://omlc.ogi.edu/software/mc/`

Several scientific publications describing and using the Monte Carlo technique is also available on this site. The program, called MCML, has become very popular within the Biomedical optics community and is today the *de facto* standard for Monte Carlo simulations. Although simple simulations of layered media (for which MCML is intended) is of little interest, the source code of MCML is often used as a base when developing new Monte Carlo software and hence the program still carries some value despite its age.

Along with MCML, a program called CONV is also provided. This program will also be used in this exercise to demonstrate spatial convolutions of Monte Carlo simulation results.

2.3 Fluorescence Monte Carlo

In Lund we have developed various Monte Carlo codes to simulate a fluorescence measurement from the tissue surface. The difference with a fluorescence simulation as compared to the normal Monte Carlo algorithm is that as light is absorbed it may give rise to a fluorescence photon. Now you thus need to follow the path also of that photon. This photon is Stoke-shifted (shifted towards longer wavelength) as compared to the excitation photon, meaning that it may experience other optical properties in the transport through tissue as compared to the excitation photon. The fluorescence

Monte Carlo simulations are usually characterized by a very long simulation time, as one now has to follow the emitted photons as well. The concept with photon weights, used in MCML to speed up the simulations drastically, becomes obscure as this concept allows absorption in each interaction between an excitation photon and the tissue. Thus an emission photon should be created in each such interaction, to be followed until it dies. That this acceleration is not useful in fluorescence Monte Carlo, together with the fact that very few of the emitted photons will reach a detector, makes the conventional Fluorescence Monte Carlo simulations very time consuming.

In order to speed up this type of simulations, we invented a novel concept to perform such simulations. This concept employs the symmetry of the problem used in MCML, i.e. that the tissue is layered. This symmetry makes it favourable to simulate the excitation and emission paths separately and then perform a convolution of the two solutions. In the general case this would require one simulation for the excitation light and one simulation per voxel for the emission light (the origin of the emission can be anywhere in the volume). By employing the symmetry of the layered media, it is possible to reduce the simulations for the emission light from one per voxel to one per depth (since simulations of the light emission path originating from a voxel at a certain depth will be identical independent on its lateral position). This will lead to a drastically reduced computation time. It was also shown that this can be further drastically improved, by identifying that the probability for a photon to find its way from position A to B is identical to the probability of the reverse path from B to A. One can thus replace all the emission simulations with one simulation, now starting from the detection fibre. This means that the problem can be solved with two simulations and a convolution.

3 Home Assignment

The home assignment for this computer exercise is to read the scientific paper by Prahl *et al.* on the basics of Monte Carlo simulation for light transport in scattering media. The paper can be found at:

`http://omlc.ogi.edu/pubs/pdf/prahl89.pdf`

After you have read this paper you should be able to describe in detail the principles of the calculations involved in simulating one step of a photon package in a Monte Carlo simulation. Also have a look at the manual for MCML and CONV, the programs which this lab is based upon. This document is available at:

`http://omlc.ogi.edu/pubs/pdf/man_mcml.pdf`

The pages 1-6 and 99-110 are the ones of interest for this lab.

Please read the two above mentioned documents as well as this one and you will be well prepared for the exercise.

4 Instructions

Everything you need for this exercise is available at:

`http://www.atomic.physics.lu.se/fileadmin/atomfysik/Biophotonics/Education/MC.zip`

The zip-file contains the executable MCML and CONV files as well as a set of MATLAB functions to provide an easily useable interface to the old DOS-based programs.

Please create a working directory (folder) on your computer hard-drive (please avoid using your network disk for this task) then download and unzip the necessary files to that folder. The folder should now contain a collections of MATLAB scripts (`.m`) and executable files (`.exe`). Start MATLAB and make your newly created folder your current folder.

> IMPORTANT: It is highly recommended to write your MATLAB code in scripts instead of just typing them in the command prompt. For example, make a script for each part of the exercise, each making all the necessary computations and plots for the current assignments. This way your code may easily be reused and/or modified. If you don't know how to do this, please ask your instructor.

4.1 MCML & conv

Here we will use an unmodified version of MCML to calculate the steady-state light distribution in turbid media featuring different properties.

4.1.1 MCML.m

MCML itself communicates with the user using standard ASCII or binary formated text files. This can be a painstaking process as the input files have to be formated exactly according to a template and the output files are not easily interpreted. To simplify the situation the MATLAB function `MCML.m` provides an interface to MCML where the input parameters are specified as arguments to the function and all the simulation results are summarised in a simple MATLAB structure.

Detailed information on how to use `MCML.m` is available in the Appendix. Briefly, to simulate 50000 photons injected at one point at the surface of a semi infinite (100 cm thick) slab featuring $\mu_s = 75$ [cm^{-1}], $\mu_a = 0.1$ [cm^{-1}], $g = 0.8$ and refractive index $n = 1.4$, we use the commands:

```
>> layers_matrix=[1.4 0.1 75 0.8 100];
>> s = MCML('exercise',50000,layers_matrix);
```

Assignments:

1. Run the above stated command and explore the output structure (in this case called `s`) and make sure you understand what the structure fields are. Some help can be found in the Appendix. Also note the name of the MCML output file stated when the simulation is done. Most likely it will be "exercise.mco". The DOS-console window can be closed as soon as the simulation is done.

2. Plot the Diffuse Reflectance as a function of r. Also display as the absorption and fluence as a function of r and z (use the `imagesc`-command). What does the data tell you? You may also want to try to image the data in logarithmic scale, using `imagesc(log(...))`.

3. Perform one more simulation, similar to the one above, but with another (reasonable) absorption coefficient. Also, use another name for the output structure, for example `s2` to avoid overwriting your previous results. Plot the reflectance vs. r of both simulations to compare the results. Did the computation time change? Discuss and explain the shapes of the curves and why the computation time varies between runs.

4. Discuss your observations and answers with you lab-instructor.

4.1.2 conv.m and Multilayer media

The simulations performed by MCML are performed using an infinitely thin pencil beam of photons incident on the medium. As most beams of light have some spatial extension a program `conv.exe` is provided to convolute the spatial impulse response to accommodate simulation results for other source geometries, *eg.* top-hat (flat-field) or Gaussian distributions. The MATLAB-script `conv.m` will start the CONV-program and read the resulting data file. The program CONV requires manual instructions (as you read in the manual during the home assignment). Help is given by typing `h` and/or in the MCML and CONV manual p. 99-110.

Assignments:

1. Enter `cs=conv()` into the MATLAB console window to start the convolution program.
2. Follow the on-screen instructions to load the MCML output data from the first simulation (most likely stored in "exercise.mco") and calculate the fluence resulting from a 1 Joule flat field beam of 0.5 cm diameter. Save the data in the file "exercise.Frz" although any filename will work.
3. Explore the structure `cs` and make sure you understand the output.
4. Compare the fluence as a function of r and z for a point source and the flat field (calculated using `conv`). Also compare the fluence of the two simulations (point source) performed before. Explain the differences. Where is the fluence highest?
5. Compare the results for the fluence and absorption (both vs. r and z) for the first simulation. Explain any differences and similarities.
6. Perform a new simulation with two layers (check the Appendix for how to do this), the first layer 1 mm thick with half the absorption coefficient of the second layer. Again, compare the results for fluence and absorption and explain the results.
7. Discuss your observations and answers with you lab-instructor.

4.1.3 S/N and μ_{eff} extraction

Here we will estimate the Signal to Noise ratio (S/N) of our Monte Carlo simulations and explore one of the drawbacks of numerical (stochastic) solutions compared to convenient analytical solutions, as experienced in the previous computer lab on Photon Diffusion. The noise of our simulation results can be estimated by calculating the standard deviation of the difference between two simulations with identical input parameters. Also we can use diffusion theory to extract a value of μ_{eff} from the simulation results. Useful expressions for this task are:

$$R(r) \propto \frac{e^{-r \cdot \mu_{eff}}}{r^2} \tag{1}$$

$$\mu_{eff} = \sqrt{3\mu_a(\mu_a + 3\mu_s')} \tag{2}$$

where Eqn. 1 is valid far from the source. To calculate the S/N the following code can be useful:

```
>> layers_matrix=[1.4 0.1 75 0.8 100];
>> s = MCML('exercise',50000,layers_matrix);
```

Assignments:

1. Perform two simulations, featuring the same parameters; a 10 cm thick slab, 25000 photons , $\mu_s = 75$ [cm^{-1}], $\mu_a = 0.5$ [cm^{-1}], $g = 0.8$ and $n = 1.4$. Remember not to overwrite the first simulation when performing the second one (simply by storing the results in separate structures).

2. Estimate at what distance from the source the S/N is equal to two for the three different cases. To get a more stable result, calculate the standard deviation of in a window of 10 channels width. Calculate this for a moving window as a function of r, and provide a plot for the S/N ratio as the signal level divided by the standard deviation. Use the MATLAB function `std` to calculate the standard deviation. The following code might be useful:

   ```
   for i=1:1:490
       index=(0:9)+i;
       n(i)=std([s1.refl_r(index)-s2.refl_r(index)]);
       s(i)=mean(s1.refl_r(index));
   end
   snr=s./n;
   ```

   Where `s1` and `s2` are the output structures of the two simulations. The code assumed `nr` for the simulations was set to 500

3. Do the above task for 100000 and 400000 photons, keeping all the other parameters the same.

4. From your findings, estimate how many photons you would have to simulate to get an S/N=2 at 50 mm from the source.

5. Use the above analytical expressions from diffusion theory to evaluate μ_{eff} from the 400000 photon simulation above. Compare this result with the parameters sent to the MCML simulation. Do the results differ? Explain?

6. Discuss your observations and answers with you lab-instructor.

4.2 Fluorescence Monte Carlo

As mentioned in Section 2.3, two specific Monte Carlo simulations have to be performed to model a fluorescence problem, one to map the absorption of the excitation wavelength and one for the light propagation of the fluorescence wavelength. The results of the two simulations are called the excitation and emission matrix respectively.

4.2.1 FLMC.m & conv_FMC.m

Here the process of fluorescence Monte Carlo has been simplified to a single MATLAB-function `FLMC.m` which works in a manner very similar to `MCML.m`. As two simulations are performed the `FLMC.m` takes two layer matrices as input arguments and returns two matrices (instead of a structure):

```
>>[ex_matix,em_matrix]=FLMC('fl_exercise', 50000 ,ex_wl_layer,em_wl_layer);
```

The first two arguments are the the same as for the `MCML.m` function. The third argument `fl_wl_layer` is the layers matrix for the excitation wavelength. It is defined in the same way as the layer matrix used by `MCML.m` but with two extra columns; one describing the yield of the

fluorophore in that layer and one for the absorption coefficient of the fluorophore alone in the tissue. An example matrix (and the one to be used in this exercise) is shown below:

```
>> ex_wl_layer= [1.4 2.0 100 0.8 0.1 0.05 0.1;...
                 1.4 2.0 100 0.8 0.1 0.75 2.0;...
                 1.4 2.0 100 0.8 1000 0.05 0.1]
```

This matrix describes a semi infinite medium featuring $\mu_s = 100$ [cm^{-1}], $\mu_a = 2.0$ [cm^{-1}], $g = 0.8$ and $n = 1.4$. The autofluorescence yield in the bulk tissue is 0.05 and the absorption of the fluorescent chromophores is 0.1 [cm^{-1}]. Situated 1 mm below the tissue surface is a 1 mm thick layer of the same optical properties but with a higher concentration of fluorophore. The yield in this layer is 0.75 and the absorption due to the fluorophore is 2.0 [cm^{-1}].

The second matrix is a layer-matrix exactly as defined for `MCML.m` describing the optical properties and geometry experienced by the photons emitted by the fluorophore. The optical properties are different as the wavelength has been Stokes-shifted:

```
>> em_wl_layer= [1.4 0.5 50.0 0.84 0.1;...
                 1.4 0.5 50.0 0.84 0.1;...
                 1.4 0.5 50.0 0.84 1000]);
```

A second MATLAB function, `conv_FMC`, is available to convolve (or in this case just multiply) the excitation and emission matrix to combine the two simulation results. The function takes three arguments, the excitation and emission matrices (which we earlier called `a` and `r`) as well as the source-detector separation, `d`:

```
>> conv_FMC(ex_matrix,em_matrix,d);
```

The units of `d` is in voxels (volume pixels) which features a size according to what we defined during simulations. The default voxel length is 0.1 mm. The function will draw a 2D cross-section map describing the spatial origin of the fluorescence signal detected at source-detector separation `d`.

Assignments:

1. Make sure you understand the two matrices discussed above and what they will simulate. Then enter them into MATLAB and run the `FLMC` program.
2. Image the two resulting matrices in both linear and logarithmic scale. Explain the results.
3. Run `conv_FMC` for a 4 mm source-detector separation and find out from where the detected fluorescence originates. What does the map shown by `conv_FMC` represent? Which part of the sample contributes the most to the signal?
4. Use `conv_FMC` to calculate at what source-detector distance the contrast between the fluorescence from the layer and the autofluorescence is maximised. How would this distance change if you change the depth of the fluorescence layer?
5. Discuss your observations and answers with you lab-instructor.

Finally, consider whether, and in that case how, you can utilise these tools for your project.

5 Appendix - MatLab Tools

This appendix provide information on how to use the MATLAB function described in the previous section.

MCML.m (simple version)

Performs a Monte Carlo simulation using MCML using the default options (suitable for this exercise).

Syntax:

```
s=MCML(filename,number_of_photons,layers);
```

Arguments:

`filename`	Name of the input and output files to/from MCML. If the name is taken `MCML.m` will add a number after the string to create a unique file name. The name must be a valid variable name in the MATLAB environment.
`number_of_photons`	Number of photons in the simulation
`layers`	Layers matrix. The rows in this matrix describe the layers of the simulation geometry, starting with the layer closest to the source. The number of layers in the medium modeled is defined by the number of rows in this matrix. Each layer is described by 5 values, stored as columns in the matrix. The values are refractive index n, absorption coefficient μ_a [cm^{-1}], scattering coefficient μ_s [cm^{-1}], anisotropy (g) factor g and layer thickness t [cm]. The order is given by: `[n mua mus g t]`

Output:

`s`	A MATLAB structure containing all simulation results as well as the simulation input parameters. Some of the important fields of this structure are: `s.refl_r` - The reflectance as a function of radial distance from the source (r), `s.abs_rz` - the amount of absorbed light as a function of distance, r, and depth, z, and `s.f_rz` - the fluence as a function r and z.

Examples:

We would like to simulate a slab of tissue, 1 cm thick with high scattering (μ_s=100 [cm^{-1}]) and low absorption (μ_a=0.1 [cm^{-1}]). The refractive index of the tissue is n =1.4 and the anisotropy is $g = 0.8$. The corresponding layers matrix is defined by (remember `[n mua mus g t]`):

```
>> layers_matrix=[1.4 0.1 100 0.8 1];
```

If we would like to add a 1 mm thick layer of lower scattering (μ_s=50 [cm^{-1}]) and higher absorbing (μ_a=2.0 [cm^{-1}]) tissue on top of the slab above the layers matrix is now defined by:

```
>> layers_matrix=[1.4 2.0 50 0.8 0.1; 1.4 0.1 100 0.8 1];
```

The simulation (50000 photon packets) is started by the command:

```
>> s=MCML('example',50000,layers_matrix);
```

The resulting absorption vs. r and z can be viewed using:

```
>> imagesc(s.abs_rz)
```

MCML.m (Complex version)

Performs a Monte Carlo simulation using MCML with the option to override the default parameters (perhaps needed for your project). For more details on the meaning of the optional arguments see the MCML manual, available at:

`http://omlc.ogi.edu/pubs/pdf/man_mcml.pdf`

This manual also provides some insight in all the results given by MCML and hence the structure of the output file and all the field in the resulting MATLAB structure.

Syntax:

```
s=MCML(...,n_above,n_below,dz,dr,number_of_dz,number_of_dr,number_of_da);
```

Arguments:

`filename`	See previous page
`number_of_photons`	See previous page
`layers`	See previous page
`n_above`	(optional, default value 1) Refractive index of the medium above.
`n_below`	(optional, default value 1) Refractive index of the medium below.
`dz`	(optional, default value 0.01 [cm]) Spatial resolution of detection grid, z-direction [cm].
`dr`	(optional, default value 0.01 [cm]) Spatial resolution of detection grid, r-direction [cm]
`number_of_dz`	(optional, default value 200) Number of grid elements, z-direction.
`number_of_dr`	(optional, default value 500) Number of grid elements, r-direction.
`number_of_da`	(optional, default value 1) Number of grid elements, angular-direction.

Output:

`s`	See previous page

Examples:

Assuming we have a layers matrix defined and we would like to simulate a slab immersed in water (n=1.33) we start the simulation by using:

```
>> s=MCML('slab_immersed_in_water',50000,layers_matrix,1.33,1.33);
```

Let's say we would like to change the number of grid elements in the depth (z) direction to 42 for the simulation above we have to give values of all the earlier arguments (in this case `dr` and `dz`):

```
>> s=MCML('slab_immersed_in_water',50000,layers_matrix,1.33,1.33,0.01,0.01,42);
```

conv.m

Starts the DOS program `CONV.exe` and will automatically identify and import the resulting file. A detailed manual on `CONV.exe` is available on p. 99-110 in the MCML and CONV manual:

`http://omlc.ogi.edu/pubs/pdf/man_mcml.pdf`

Syntax:

```
cs=conv();
```

Output:

`cs`	A structure containing the fields `r` - a vector of r, `z` - a vector of z and `data` - a matrix containing the convolved data requested during `CONV.exe` execution.

FLMC.m

Performs a Fluorescence Monte Carlo simulation comprising two actual simulations, one for the excitation light and one for the emitted fluorescent light.

Syntax:

```
[ex em]=FLMC(filename,number_of_photons,ex_layers,em_layers...);
```

Arguments:

`filename`	Same as `MCML.m`.
`number_of_photons`	Same as `MCML.m`, used for both simulations.
`ex_layers`	Layers matrix for the excitation photons. Is similar to the layers matrix used in `MCML.m` with the addition of two columns describing the fluorescence yield, `yi`, and the absorption due to the flourophore alone, `fmua`. The extended layers matrix looks like: `[n mua mus g t yi fmua]`
`em_layers`	Layers matrix for the photons emitted by the fluorophore. Exactly the same as the layers matrix used by `MCML.m`.
`...`	The optional parameters, as well as their default values are the same as in `MCML.m`. Any options used apply to both simulations.

Output:

`ex`	A matrix of the excitation light absorbed by fluorophores vs. r and z.
`em`	A matrix describing photon density vs. r and z for the emisson wavelength.

Examples:

An example of layer matrices and the use of FLMC is given in the exercise. The use of options is demonstrated on a previous page (on `MCML.m`).

conv_FMC.m

Multiplies the matrices resulting from a Fluorescence Monte Carlo simulation using an offset corresponding to a source-detector separation. Gives a 2D map of the origin of the detected fluorescent light.

Syntax:

```
conv(ex,em,d);
```

Arguments:

`ex`	From `FLMC.m`. A matrix of the excitation light absorbed by fluorophores vs. r and z.
`em`	From `FLMC.m`. A matrix describing photon density vs. r and z for the emisson wavelength.
`d`	The source-detector separation in voxel (volume pixel) lengths (r-direction).

Output:

`conv_FMC.m` will draw (using the MATLAB command`imagesc`) a 2D cross-section map describing the spatial origin of the fluorescence signal detected at source-detector separation `d`.