IMPROVING MONTE CARLO PROGRAM FOR LIGHT TRANSPORT IN TISSUE

An Undergraduate Research Scholars Thesis

by

LARS KUSLICH

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Research Advisor: Dr. Michael McShane

May 2015

Major: Biomedical Engineering

TABLE OF CONTENTS

ABSTRACT

Improving Monte Carlo Program for Light Transport in Tissue. (May 2015)

Lars Kuslich
Department of Biomedical Engineering
Texas A&M University

Research Advisor: Dr. Michael McShane
Department of Biomedical Engineering

Monte Carlo Simulations use random number generators and aggregation in order to numerically solve various real-world problems that are too difficult to solve analytically. They are currently the gold standard for simulating photon transport through tissues, and with the adoption of GPU computing, a program by the name of Monte Carlo eXtreme can run these simulations faster than ever. However, this program still lacks several features that would be useful to researchers, such as fluorescent material support and easy simulation of multiple wavelengths. I have added the multi-wavelength feature to MCX, and I have started the process of adding a feature that will allow researchers to easily model fluorescence materials. Although more testing is required, I have laid the foundations of a program that is more readily useful and robust while retaining the speed increases afforded by GPU computing.

CHAPTER I

INTRODUCTION

The Monte Carlo method using GPU computing

The Monte Carlo method is a means of numerically solving various real-world problems that are too difficult to solve analytically[1]. The method involves drawing a large number of pseudo-random numbers from a range of possible values. These numbers are then used as inputs for an algorithm that will deterministically solve an equation or problem. This process is repeated many times, usually thousands of times, in order to get an array of different possible solutions to the problem. These outputs are then aggregated in order to produce a final solution that is typically very similar to the experimentally determined solution.

Due to the repetitive nature of this method and the fact that it follows a set algorithm, Monte Carlo methods are naturally adapted to computational simulations. These Monte Carlo simulations have found use in a wide range of scientific and engineering fields[2], and of particular importance to this proposal, Monte Carlo simulations have become the gold standard for modeling light transportation in tissues[3]. In 2009, Fang and Boas[4] developed a new Monte Carlo program for just this purpose; it is known as Monte Carlo eXtreme (MCX). Their program's most noteworthy feature was their use of Graphics Processing Unit (GPU) computing in order to reduce the processing time needed for each simulation. They were able to achieve processing speeds that were several hundred to one thousand times faster than those possible with pure CPU computing.

Although the creation of this program was a great advancement in terms of reducing Monte Carlo simulation time, there are still several useful features that have not yet been added. For example, there is no support for fluorescent materials[5]. While this does not make simulating fluorescence with MCX an impossible task[6], it does mean that the users will need to either alter the code themselves or use a less sophisticated approximation. Also, as with most other Monte Carlo programs of this type, there is no simple way of simulating multiple wavelengths of light. Presently, one must run multiple separate simulations with MCX in order to accurately simulate multiple wavelengths. This drawback makes fluorescence simulations especially tricky since most fluorophores emit light at multiple wavelengths[7]. While there are workarounds and approximations that researchers can and have used in the past, the resulting simulations are not as accurate as they could be. These and other deficiencies still limit the overall usefulness of MCX. In this thesis, I will describe the results of my attempts to alter the MCX source code in order to give MCX these features, but I will begin with a short overview of MCX as it currently stands.

Overview of MCX

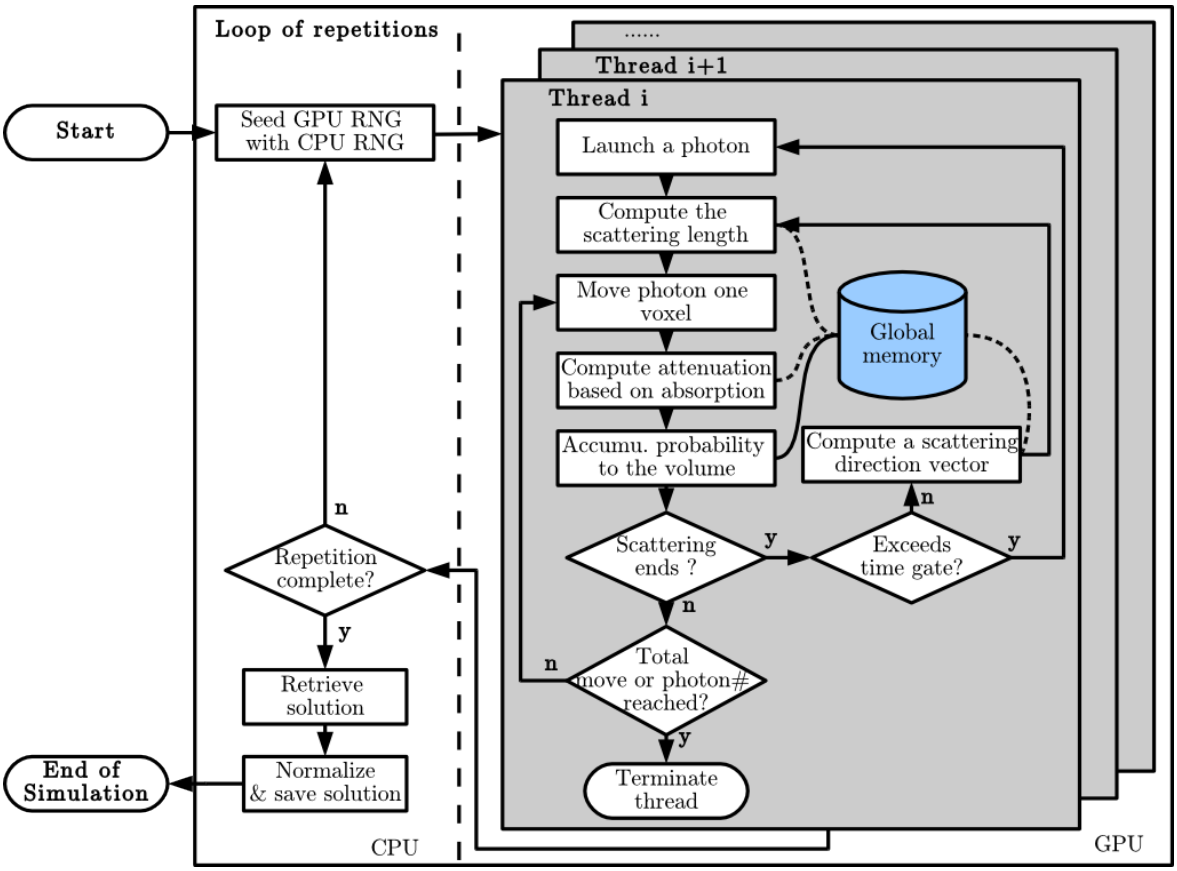
In MCX, three overall inputs are used to determine the various specifications for performing a single wavelength simulation. First, a volume file defines the overall space in which the photons will propagate. Specifically, the volume file defines various shapes, or media, in a 3-D Cartesian coordinate system. These shapes can either be relatively simple, such as a cube, sphere, cylinder, etc., or they can have an arbitrarily defined 3-D shape. The exact dimensions of the shapes are user-defined, and the user is able to define for each shape the index that will determine the

optical properties of that particular shape. As an example of one possible setup, it is possible to create a layered structure by defining successive cuboids that each have their own dimensions and indices, and it is also possible to embed shapes within other shapes. Using these two features, one can create a model of the human skin complete with each individual layer of skin. In this model, one can use embedded shapes to represent such things as blood vessels or even implanted objects depending on the needs of the researcher.

The second type of input is an actual input file, which is where most of the important parameters for a simulation are defined. This input file is where the volume file that the user wants to use for a simulation is chosen, and the optical properties for each particular shape are defined here as well. This second point is important to note as the user does not directly specify the wavelength at which to simulate. Instead, the user specifies the optical properties for the various shapes or materials under investigation, and these optical properties are meant to represent the behavior of the shapes at the desired wavelength. With this method, any number of theoretical or fictional materials can be modeled at any arbitrary wavelength. The user can also define in the input file the number of photons used in the simulation, the position of the photon source, which can be either inside or outside the volume-of-interest, the type of photon source to use, the initial direction of the photons, the start time, end time, and time step size for the simulation, and other important parameters. One particularly noteworthy feature of the input file is the ability to define detectors on the edges of the volume-of-interest. These detectors record every photon that escapes the volume-of-interest through the area where they are defined, and the detectors also record the value of the partial pathlength that the photon traveled before escaping. These detectors are very useful for performing diffuse reflectance studies.

The third type of input is various input parameters that are set when the program is executed. These parameters allow the user to determine which input file to use, the number of GPU threads to use, the name of the output files, whether to take into account internal or external boundary reflections, etc. It is even possible for the user to define new shapes to use in addition to those defined in the volume file. For some cases, the input parameters are able to overwrite the parameters set by the input file, such as in the case of the number of photons used for a simulation.

Once all of the input parameters are set, MCX uses those parameters to initialize the necessary shapes, photons, and boundaries. After initialization, the total number of photons is divided up among the various computer threads located on the GPU. Each thread is responsible for calculating the scattering length, amount absorbed, position, and any other relevant parameters of the photons that it has been assigned. Each thread simulates this photon movement until the photon either exits the volume-of-interest or is absorbed. After that, the process is repeated for the next photon until all photons have been simulated. The results are then compiled and written to two separate output files. One output file is for the photon flux at the end of the simulation, and the other output file contains the information collected by the detectors. The following picture summarizes the general MCX algorithm[4].


Loop of repetitions
.....
Thread i+1
Thread i
Start
Seed GPU RNG with CPU RNG
Launch a photon
Compute the scattering length
Move photon one voxel
Global memory
Compute attenuation based on absorption
Accumu. probability to the volume
Compute a scattering direction vector
Scattering ends ?
Exceeds time gate?
Total move or photon# reached?
Terminate thread
Repetition complete?
Retrieve solution
Normalize & save solution
End of Simulation
n
y
CPU
GPU

CHAPTER II

METHODS & TESTING PROCEDURES

General Procedure

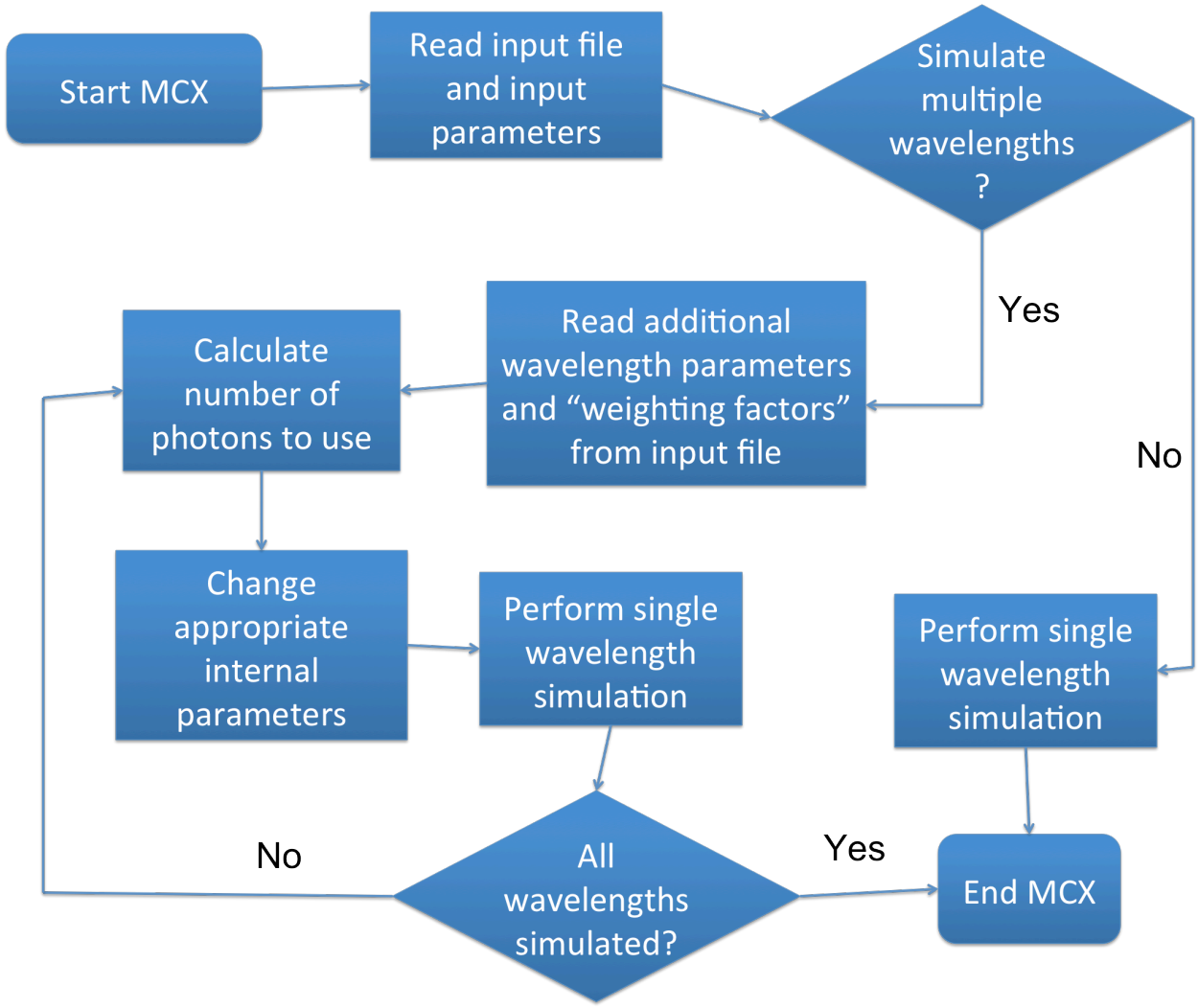
Before any changes could be made to MCX, the first step was to understand how MCX performs the simulations. This was done using a holistic inspection of the myriad source code files. First, the top-most source file that is responsible for calling functions from the other source files was identified. From this file, each sequential function that was called was examined until a sufficient understanding of its purpose was achieved, and any relevant constants, structures, macros, etc. that are located in various header files were located and taken note of as well. This process was repeated until the general computational process behind MCX was understood. This somewhat labor-intensive process was necessary because of the relatively sparse commenting on how certain intricacies of the program are performed. All of this was done on a Dell Precision T3600 Workstation with an x86-64 CPU architecture running Ubuntu 12.04. Most importantly, this computer used an NVIDIA GeForce GTX 770 to perform all GPU-related computations, and the CUDA 6.0 parallel computing platform was used to change the relevant C code for the program.

Multi-Wavelength Feature

Proof-of-Concept Implementation

Before implementing the multi-wavelength feature as it was originally envisioned, a simpler implementation that is more congruent with the original MCX set up was performed. This implementation was done to test whether performing multi-wavelength simulations in a certain way would change the output as well as to find any specific internal calculations that would need

to be performed in the final implementation. It was also used to test whether the multi-wavelength feature might slow down the simulation time for a given wavelength. For this implementation, MCX was changed so as to accept multiple sets of optical properties from the input file. Each set of optical properties would represent a different wavelength, and the optical properties in each set would be indexed to the appropriate shapes in exactly the same manner. In addition, a new parameter called the *weighting factor* was added to each set of optical properties. The weighting factor represented the fraction of the chosen number of input photons that would be used for a given wavelength. In this way, the variation in intensity of a light source at certain wavelengths could be modeled. The user could control whether MCX performed a multi-wavelength simulation or not by altering an input parameter that was added for this purpose. The multi-wavelength simulations themselves were performed by re-running the core MCX program for each wavelength. Each set of optical properties were read into the program, and the appropriate internal parameters, e.g. the optical properties and the number of input photons, were altered before the simulation was run again; the actual simulation algorithm was not altered. The following flowchart summarizes how this feature will perform.



For testing, one million photons were simulated at five different wavelengths, and these photons were put into a cube with 60 mm on each side that had homogeneous, arbitrarily defined optical properties for every wavelength. A single multi-wavelength simulation was used to simulate all five wavelengths, and standard MCX simulations performed at all five wavelengths were used as the control. For both the multi-wavelength and the control simulations, each wavelength was simulated three times in order to determine the standard deviation of the results. The recorded results were the photon flux at the time of stopping for each wavelength and the total time required to simulate each wavelength.

Final Implementation

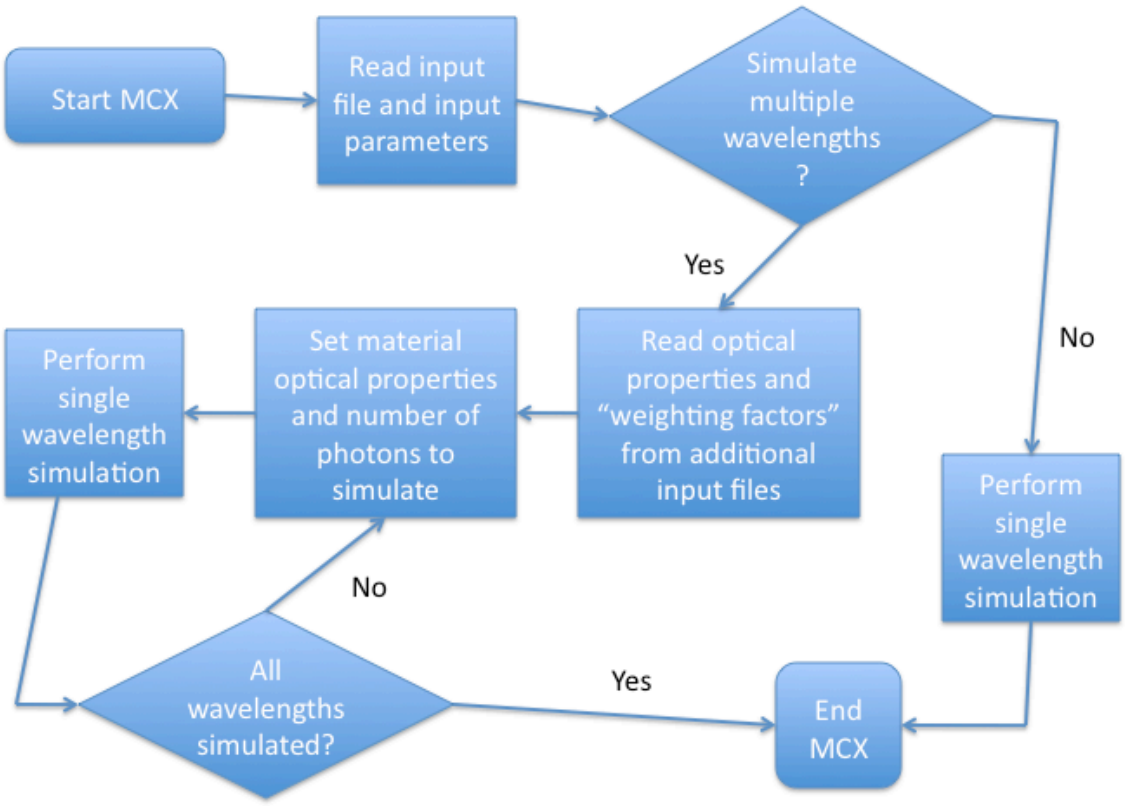
For the final implementation, the standard MCX input file was unaltered. Instead, when the proper input parameter is set, MCX will read the optical properties and weighting factors from user-provided text files. The names of these files are entered as input parameters at the start of MCX. There are two basic types of multi-wavelength input files that will be read: the source file and the files for the optical properties of the shapes inside the volume-of-interest. Both types of input files are simple text files with a series of ordered numbers that represent the various input parameters, and these numbers are arranged such that each line in all of the input files holds information for exactly one wavelength.

The source file is the first multi-wavelength input file that is read, and it contains two numbers in each line. The first number in a given line in the source file contains the wavelength, in nanometers, to be simulated. This number is internally recorded and will be used to find the appropriate optical properties for a given wavelength from the other input files. This number will also be used as the name of the output files for its respective wavelength. The second number in a given line is the weighting factor for that wavelength. This weighting factor works the same way as the weighting factor used in the previous implementation.

For the second type of input file, there are five numbers in each line, and there are as many different input files as there is need for a given set of optical properties at each wavelength. For instance, if one were to create a simple model that has three sequential layers that each have different optical properties, there would need to be three input files that each contain the optical

properties for their respective layers. The first number in one of these files is the wavelength, in nanometers, to be simulated. This number is compared against all of the wavelength numbers obtained from the source file. If there is a match, the corresponding optical properties are saved; this process is repeated for every other input file. The second, third, fourth, and fifth numbers correspond to the refractive index, the absorption coefficient, the scattering coefficient, and the anisotropy factor, respectively, which are the optical properties to be used. If a wavelength number match is found, these numbers are assigned to their appropriate shapes immediately before the corresponding simulation. However, if a match is not found, the corresponding shapes will be assigned a back-up set of optical properties. This back-up set is basically just the set of optical properties that would normally be assigned in a single wavelength simulation, and it is located in the standard MCX input file.

As mentioned previously, multi-wavelength simulations will be performed when the proper input parameter is set. In addition to turning on the multi-wavelength feature, this input parameter will determine the number of optical property files to search for. For instance, if one has a model with three layers in it, they would need to enter the number three for the multi-wavelength input parameter to find all of the proper input files. If the number is too small or too large, MCX may try to perform an undefined operation and will likely crash. The following flowchart summarizes the entire multi-wavelength simulation process.

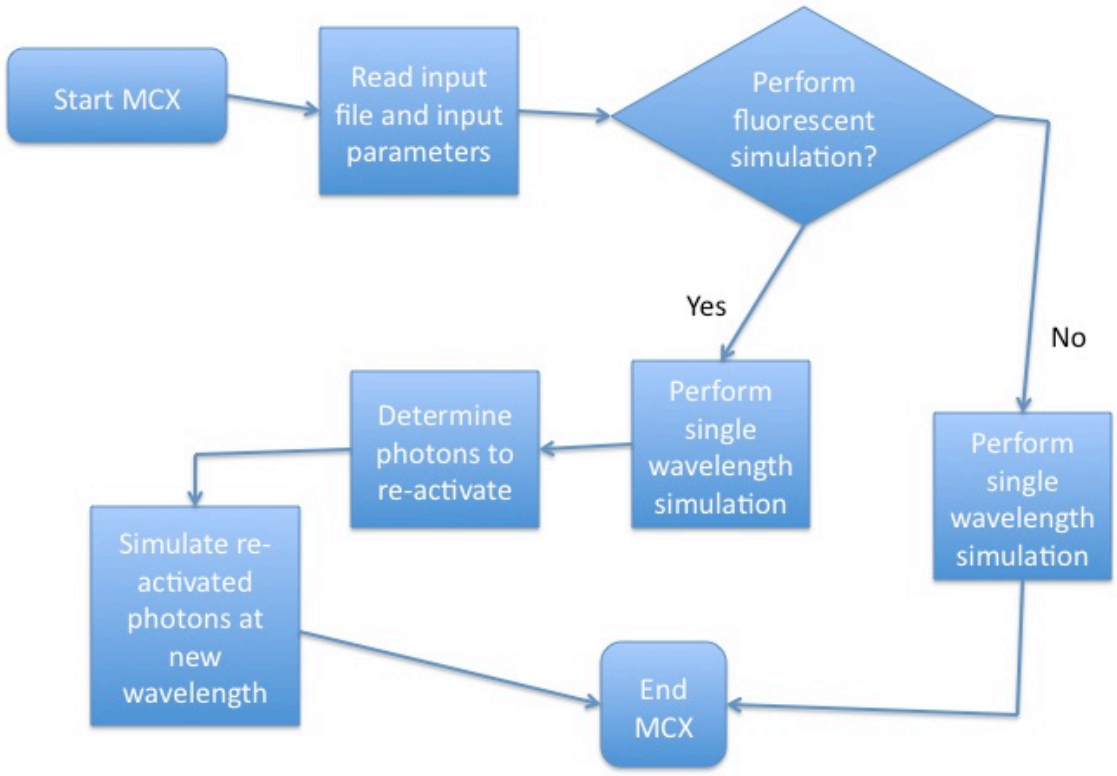


In order to test the final implementation, 53 wavelengths ranging from 400 nm to 800 nm were simulated using the multi-wavelength feature. One million photons were multiplied by the respective weighting factors of each wavelength, and the photons were put into a volume featuring four layered cuboids. The first two cuboids had a depth of 10 mm while the last two cuboids had a depth of 20 mm; each cuboid featured a 60 by 60 mm area. Single wavelength simulations were used as the control for these tests, but they were only performed at certain key wavelengths. The photon fluxes at the end of representative simulations from both the single wavelength simulations and the multi-wavelength simulation were compared for equivalency.

Fluorescent Feature

Unfortunately, the fluorescent feature has not been completed at this time. This section will first describe the parts of the fluorescent feature that have been implemented, and then this section will describe the remainder of the feature's algorithm as it is currently imagined. For the fluorescent feature, the standard MCX input file was altered to contain a new parameter that determines whether a material is fluorescent or not. This parameter is associated with the optical property sets, and one of these parameters is set for every set of optical properties. A parameter value of one indicates that the shapes that receive that particular set of optical properties will be fluorescent, and a parameter value of zero indicates that the shapes will not be fluorescent. If a shape is designated as fluorescent, the location and weight of any photons that are absorbed inside that shape are recorded.

The idea behind this feature is that the photons absorbed in the fluorescent material will be re-activated for the fluorescent simulation. However, a certain number of photons will be randomly selected to not re-activate. This number will be based on the quantum yield of the fluorescent material. The photons that are re-activated will then continue their simulation but with new material optical properties to represent the change in wavelength. The following flowchart shows the general procedure of the fluorescent simulation feature as it is currently envisioned.



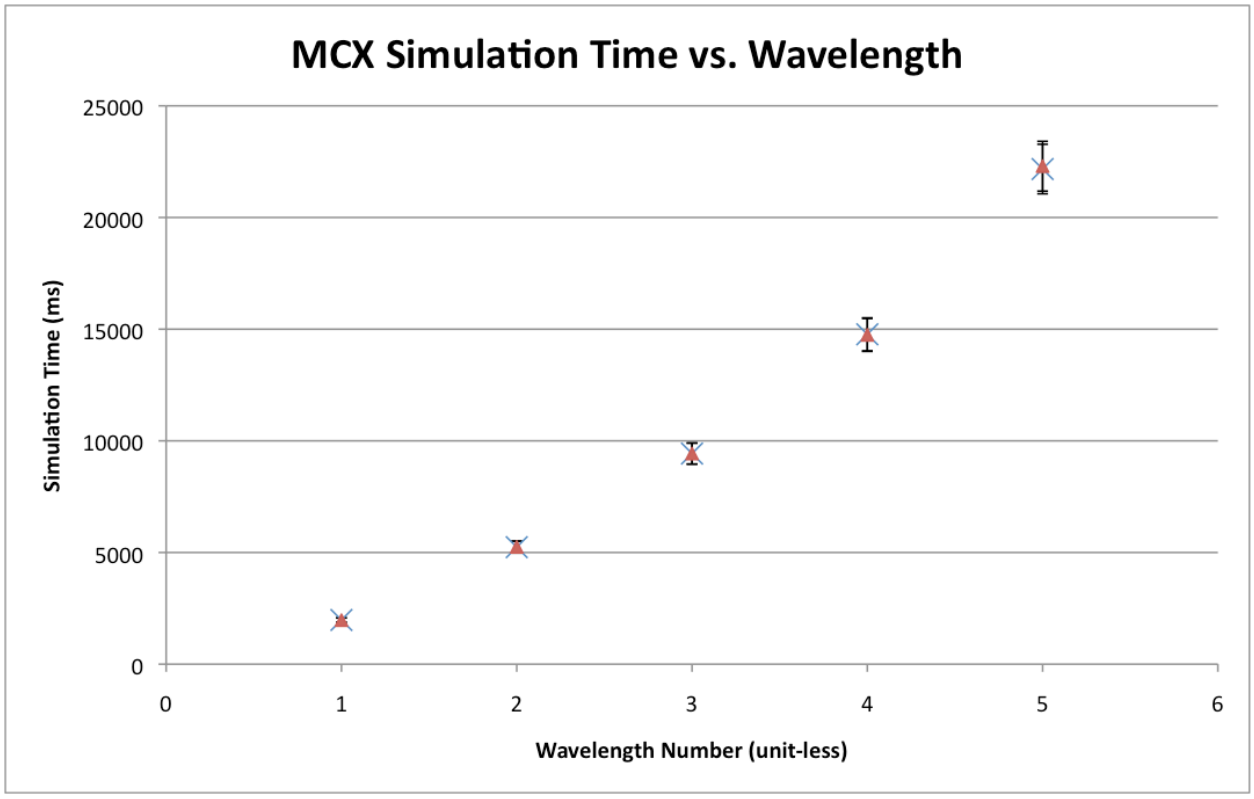
An additional input parameter will be used to determine whether MCX will perform fluorescent simulations. If this parameter is set to zero, MCX will perform a standard simulation. Eventually, it is hoped that this feature will be able to be paired with the multi-wavelength feature so that MCX can produce a multi-wavelength fluorescent output alongside a multi-wavelength input. As this feature has not been completely implemented at the time of this writing, no tests could be performed to determine its accuracy.

CHAPTER III

RESULTS & DISCUSSION

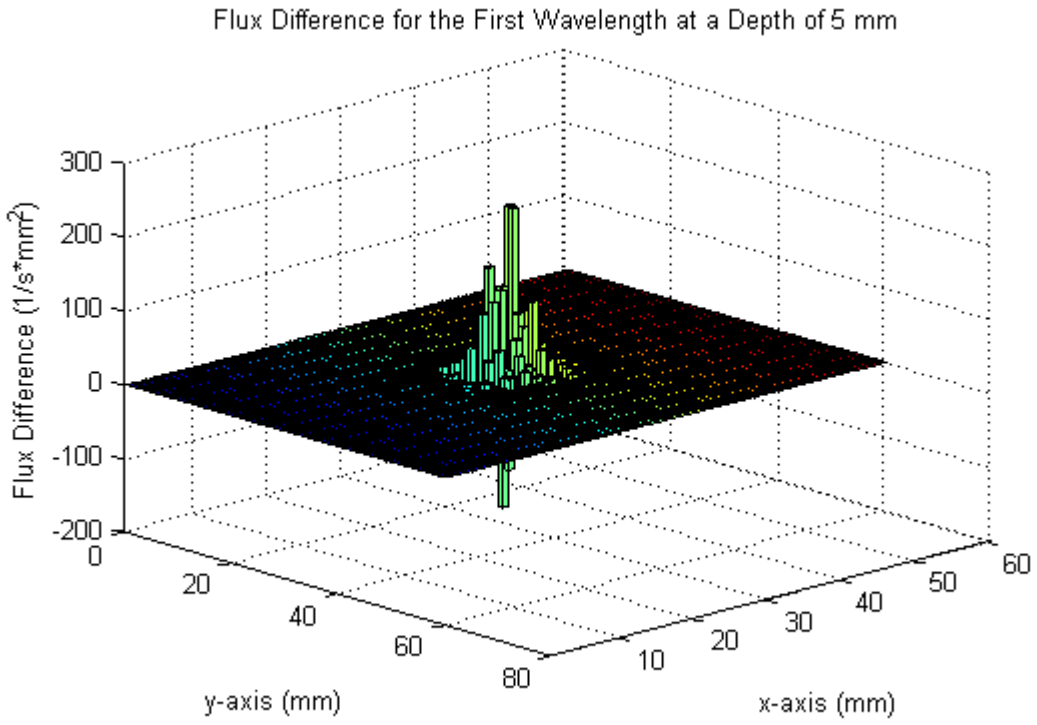
Multi-Wavelength Proof-of-Concept Implementation

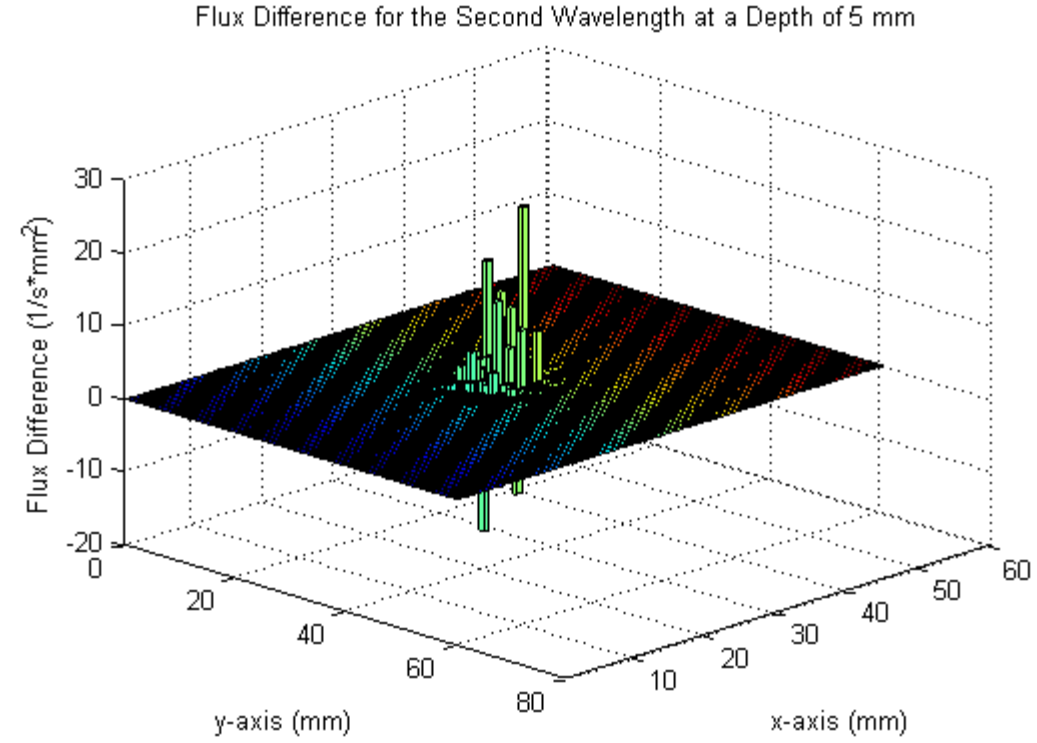
The figure below shows the average time to complete a simulation plotted against the respective wavelength number. The triangles represent the results from the single wavelength simulations while the crosses represent the results from the multi-wavelength simulations. As seen in the figure, there is significant overlap between the simulation times using both single and multi-wavelength features. The average simulation times remain close even when the overall simulation time increases, and even the 95 percent error bars for the simulation times are close for all wavelengths.

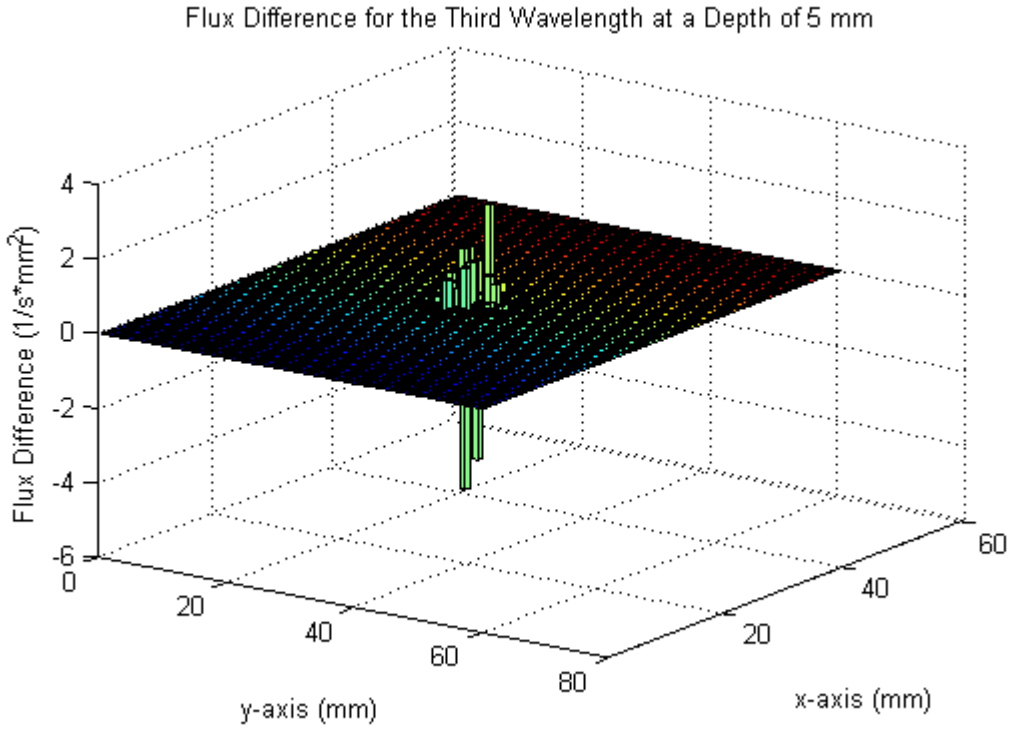


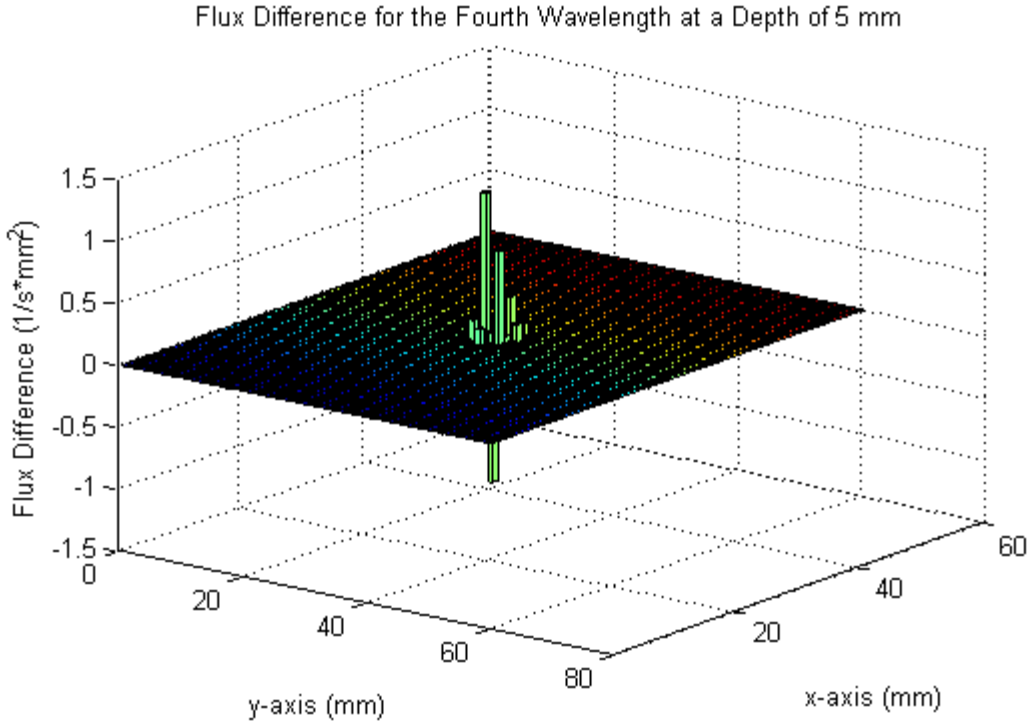
These results mean that using the multi-wavelength feature does not slow down the simulation time of MCX for a given wavelength. This also means that the total simulation time for all wavelengths is the same. In other words, the time it takes to simulate all wavelengths using the multi-wavelength feature is the same as the time it takes to simulate all wavelengths using the single wavelength feature. This is entirely expected as the underlying simulation algorithm, which was only designed to simulate one wavelength at a time, was unchanged. However, the main benefit of the multi-wavelength feature is not in the decreased simulation time. The main benefit of this feature is that it allows a researcher to perform only a single, albeit relatively long, simulation for multiple wavelengths. If one were to simulate the same number of wavelengths using standard single-wavelength MCX, one would need to remain at their computer and manually change the inputs and execute the program for each wavelength. With the multi-wavelength feature, one would only need to define the input parameters once and execute the program once. They would then be free to perform other tasks in the meantime. Therefore, this result is entirely acceptable.

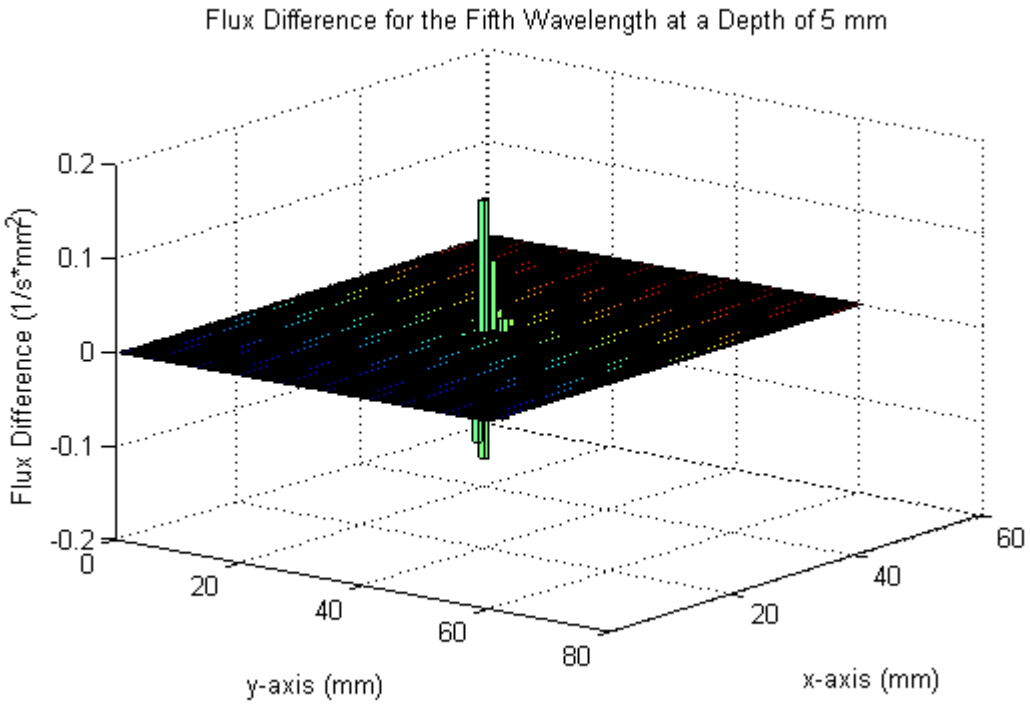
In order to determine the equivalency of the results, the fluxes from both the single and multi-wavelength simulations were compared at a standard depth from the surface of the cube. In this case, the standard depth was chosen to be 5 mm. The differences in flux for each wavelength were calculated and plotted as 3D histograms using Matlab. Theoretically, these differences should be zero if the two types of simulation are totally equivalent. The following figures show the histogram plots for each wavelength. The x and y axes in these plots refer to locations in the volume of interest at the specified depth.


Flux Difference for the First Wavelength at a Depth of 5 mm
Flux Difference (1/s*mm²)
300
200
100
0
-100
-200
0
20
40
60
80
y-axis (mm)
10
20
30
40
50
60
x-axis (mm)


Flux Difference for the Second Wavelength at a Depth of 5 mm
Flux Difference (1/s*mm²)
30
20
10
0
-10
-20
0
20
40
60
80
y-axis (mm)
10
20
30
40
50
60
x-axis (mm)


Flux Difference for the Third Wavelength at a Depth of 5 mm
Flux Difference (1/s*mm²)
4
2
0
-2
-4
-6
0
20
40
60
80
y-axis (mm)
20
40
60
x-axis (mm)


Flux Difference for the Fourth Wavelength at a Depth of 5 mm
Flux Difference (1/s*mm²)
1.5
1
0.5
0
-0.5
-1
-1.5
0
20
40
60
80
y-axis (mm)
20
40
60
x-axis (mm)

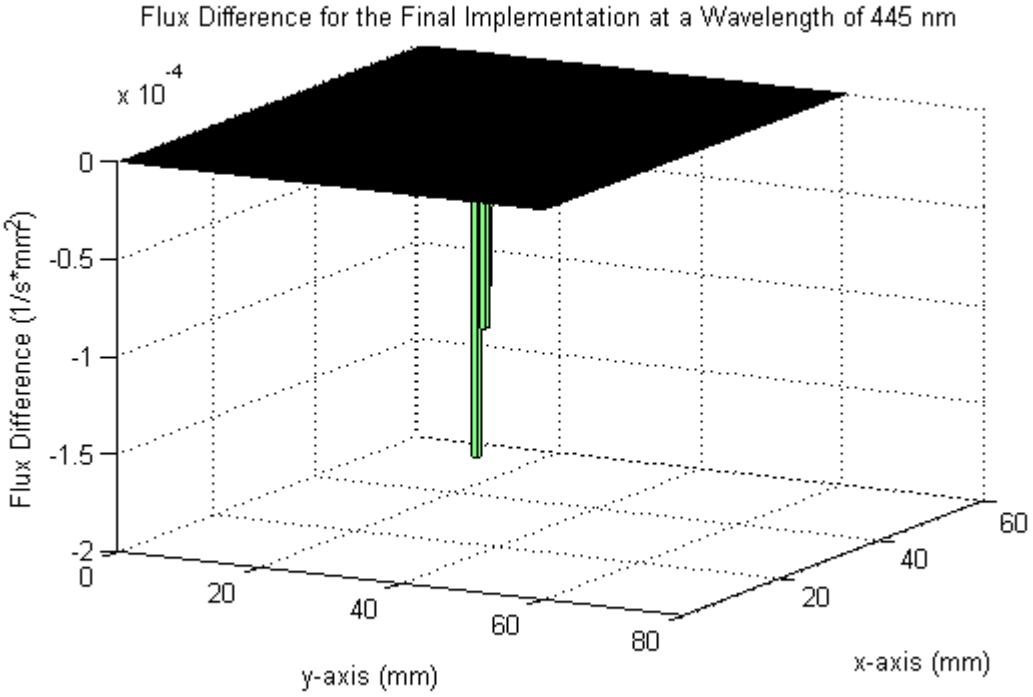


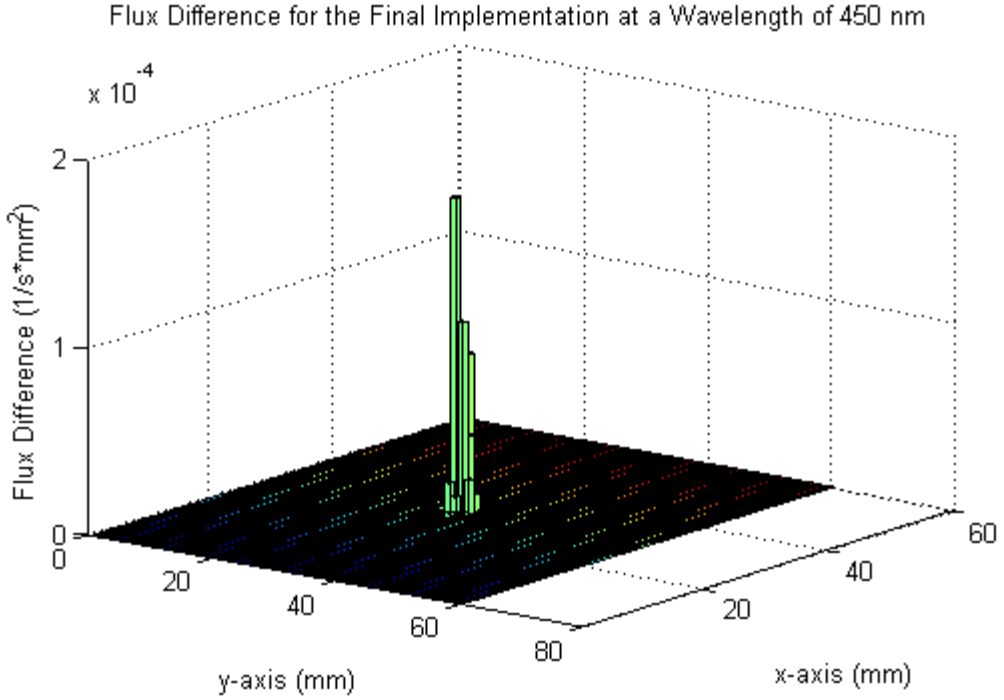
According to the plots above, the difference in flux between the two types of simulation decrease as the wavelength increases. More importantly, the flux differences for the first and second wavelengths appear to be fairly large in some locations. However, these differences are not very large relative to the original values of the flux. According to the table of sample values below, there are roughly two orders of magnitude between the original values and the difference between the values. Although these results are worse than the theoretical results, the several order of magnitude differences mean that the results from the multi-wavelength feature are similar enough to the single wavelength simulation results to be useful in a practical sense. In other words, these results show that the multi-wavelength feature produces acceptably accurate results.

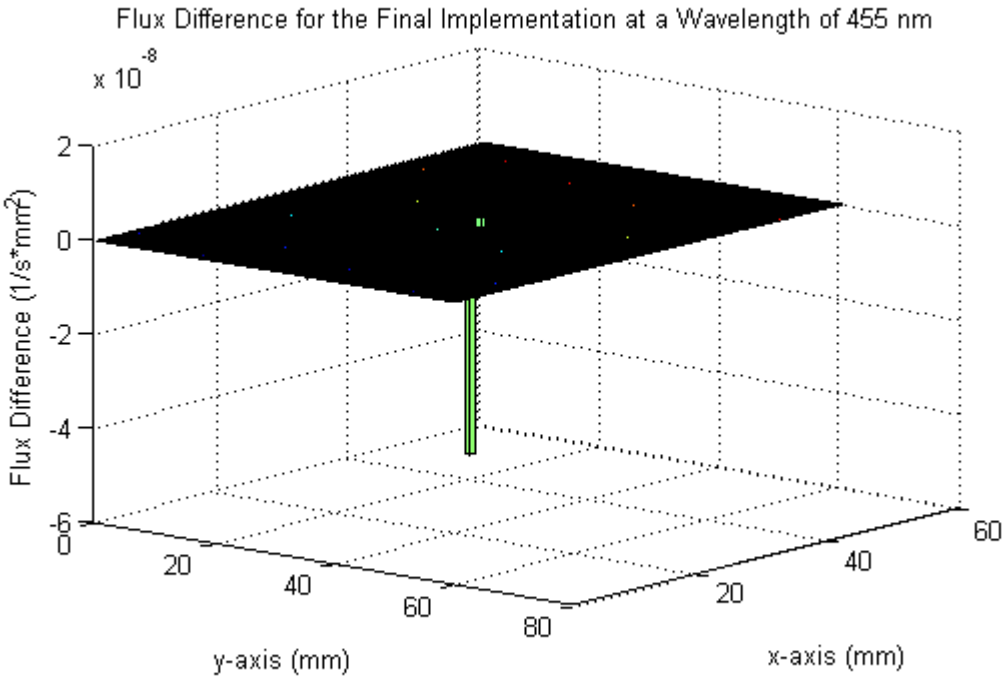
Sample Flux at 1st Wavelength	64,888 1/mm^2*sec
Sample Flux at 1st Wavelength	65,001 1/mm^2*sec
Sample Flux at 2nd Wavelength	9758.7 1/mm^2*sec
Sample Flux at 2nd Wavelength	9767.7 1/mm^2*sec
Sample Flux at 3rd Wavelength	1200.3 1/mm^2*sec
Sample Flux at 3rd Wavelength	1195.3 1/mm^2*sec
Sample Flux at 4th Wavelength	121.77 1/mm^2*sec
Sample Flux at 4th Wavelength	122.95 1/mm^2*sec
Sample Flux at 5th Wavelength	13.44 1/mm^2*sec
Sample Flux at 5th Wavelength	13.3 1/mm^2*sec

Multi-Wavelength Final Implementation

Similar to the previous implementation's equivalency test, 3D histograms of the differences in flux between the multi-wavelength simulation and various single wavelength simulations were created using Matlab. Again, these histograms used the same standard depth for comparison, which was again chosen to be 5 mm. The following figures show these histogram plots for each representative wavelength. The 445 nm, 450 nm, and 455 nm wavelengths were chosen as the representative wavelengths for comparison. The x and y axes in these plots refer to locations in the volume of interest at the specified depth.


Flux Difference for the Final Implementation at a Wavelength of 445 nm
x 10^-4
0
-0.5
-1
-1.5
-2
Flux Difference (1/s*mm^2)
0
20
40
60
80
y-axis (mm)
20
40
60
x-axis (mm)


Flux Difference for the Final Implementation at a Wavelength of 450 nm
x 10^-4
2
1
0
Flux Difference (1/s*mm^2)
0
20
40
60
80
y-axis (mm)
20
40
60
x-axis (mm)



These plots are similar to those generated for the previous implementation in that not all of the actual flux differences are zero. Unlike the results for the previous implementation, there is no clear difference in scale between the original fluxes and the differences between the fluxes. The following table of sample flux values gives an idea of the scale of the original fluxes, and from this table, it can be seen that the original fluxes are much closer to zero than for the previous implementation. The reason for these lower fluxes is because of the chosen optical properties for these wavelengths. With the amount of time the simulation was allowed to run, which was 100 nanoseconds, the tissue should have absorbed practically all of the photons after the simulation is finished. This means that the corresponding fluxes should be zero or close to zero, and the given results are therefore very accurate despite the fact that the flux differences are on the same scale as the original flux values. The reason for certain non-zero flux values is likely because of the way that floating point numbers are handled by the computer. Overall, these results are

acceptable, but the relatively small number of total results, i.e. the small number of fluxes/depths checked, means that equivalency between the multi-wavelength feature and the single wavelength feature cannot definitively be proven.

Sample Flux at 445 nm	-0.00006352 1/mm^2*sec
Sample Flux at 445 nm	0.00010465 1/mm^2*sec
Sample Flux at 450 nm	0.00010465 1/mm^2*sec
Sample Flux at 450 nm	-0.00006352 1/mm^2*sec
Sample Flux at 455 nm	-0.00037922 1/mm^2*sec
Sample Flux at 455 nm	-0.000042292 1/mm^2*sec

CHAPTER IV

CONCLUSION & FUTURE WORK

I was able to add a multi-wavelength feature to MCX by altering its source code, and I have laid the foundation for the addition of a fluorescent material feature as well. Unfortunately, I was not able to conclusively prove that the multi-wavelength simulation results are equivalent to the single wavelength simulation results. However, I was able to show that, for a given wavelength, the use of the multi-wavelength feature does not slow down MCX relative to the standard single wavelength MCX.

In terms of future work, I plan on performing tests to conclusively determine whether the multi-wavelength simulation results are equivalent to single wavelength simulation results. If the multi-wavelength feature does give equivalent results, I can add more internal error-checking mechanisms both to this feature and to MCX as a whole. I will attempt to finish adding the fluorescent material feature as well, and I could update MCX's GUI to reflect these new features. I hope to eventually share these new features of MCX with its original creators so that researchers would be able to take advantage of them.

REFERENCES

1. http://mathworld.wolfram.com/MonteCarloMethod.html

2. J. G.Amar, "The Monte Carlo Method in Science and Engineering". University of Toledo website.

3. C. Zhu and Q. Liu, "Review of Monte Carlo Modeling of Light Transport in Tissues," Journal of Biomedical Optics 18(5), 050902 (May 2013).

4. Q. Fang and D. A. Boas, "Monte Carlo Simulation of Photon Migration in 3D Turbid Media Accelerated by Graphics Processing Units," Optics Express, vol. 17, issue 22, pp. 200178-20190 (2009).

5. TODO.txt (To-do file that came with the MCX binary; lists probable future developments for MCX)

6. L. Zhao et al., "The Integration of 3-D Cell-Printing and Mesoscopic Fluorescence Molecular Tomography of Vascular Constructs within Thick Hydrogel Scaffolds," Biomaterials. Jul. 2012; 33(21): 5325-5332.

7. http://www.olympusmicro.com/primer/techniques/confocal/fluorophoresintro.html