Discrete ordinates CT organ dose simulator (DOCTORS)

Edward T. Norris

DISCRETE ORDINATES CT ORGAN DOSE SIMULATOR (DOCTORS)

by

EDWARD T. NORRIS

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

NUCLEAR ENGINEERING

2017

Approved by

ABSTRACT

Computed tomography (CT) has become pervasive in medical diagnostics as improved imaging techniques and processing algorithms provide higher quality information to doctors. However, the exponentially increasing usage of CT has raised concerns regarding long term low-dose radiological risks. Currently, the dose to patients is computed using Monte Carlo methods and experimental tests. In other areas of radiation transport, deterministic codes have been shown to be much faster than Monte Carlo codes.

Currently, no deterministic methodology exists to automatically generate a spatially distributed dose profile from a CT voxel phantom. This work proposes a new code, Discrete Ordinate CT Organ Dose Simulator (DOCTORS) which utilizes a GPU accelerated raytracer and discrete ordinate solver to compute photon flux in the patient. The flux is then converted to dose.

The DOCTORS code was benchmarked against MCNP6 and found to have good qualitative agreement using both a water phantom and a realistic patient phantom. DOCTORS was also found to be much faster than MCNP6; MCNP takes hours to compute flux profiles that take less than a minute using DOCTORS.

A GPU algorithm was implemented that speeds up the DOCTORS code by a factor of up to nearly 40 for large problems. GPU acceleration was found to benefit smaller problems much less. Speedup was seen in both single precision and double precision problems.

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Liu and all of the other graduate students for their continued support through my graduate career. I appreciate the time they spent helping me identifying problems and providing insight to solutions. I would also like to thank my committee members for their guidance throughout my PhD work.

I would also like to acknowledge the NRC for providing funding for my work. My work is supported by NRC grant NRC-HQ-13-G-38-0026.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Since its introduction in 1973 by Hounsfield [1973], computed tomography (CT) has become pervasive in medical diagnostics as improved algorithms and techniques give doctors access to higher quality information. Increased information results in faster and more accurate diagnosis. However, the exponentially increasing usage of CT and dose patients receive from it has raised concerns of potential long term risks [Brenner and Hall, 2007, Einstein et al., 2007a, Monson et al., 2004, Einstein et al., 2007b, McCollough et al., 2009, Yu et al., 2009]. These concerns are exemplified by the push for low dose CT.

Though concerns have been raised in the medical community about the risks associated with CT, the techniques used to quantify dose are very simple and not patient specific. The dose-length-product (DLP) is used as a measure to quantify the effective dose a patient receives. However, the DLP must be modified by a factor accounting for many variations amongst patients such as age, gender, and size as well as the kVp used in the procedure. Currently, dose estimation relies on *a priori* computation verified with a standardized benchmark.

No methodology currently exists to verify that the dosimetry evaluation was accurate after the patient has undergone the procedure. This work proposes a patient specific methodology by which a patient's CT reconstruction is used to compute the dose received from the radiation beam. This can also help doctors estimate spatial dose distribution in the patient to ensure no specific organ received more dose than permissible.

This work implements and analyzes a new computer code system called Discrete Ordinate CT ORgan dose Simulator (DOCTORS) by which the dose to a patient is computed using a full transport solution inside a CT phantom. The methodology converts a CT mesh into a voxel phantom of materials and densities. The user supplies information about the beam and a discrete ordinate method computes the flux throughout the phantom. The flux is then used to compute the dose using local energy deposition to relieve the need for secondary electron transport.

DOCTORS leverages graphics processing units (GPUs) to accelerate the transport step. GPUs differ from the central processing unit (CPU) in that each of the many cores performs identical instruction to all others at the same time but each with different data. For example, consider a grayscale 1024×768 image. If some pixelwise operation is applied on a CPU, 786,432 operations must occur. Onboard a GPU with 1024 cores, every pixel in the entire row can be computed simultaneously reducing the number of operations to 768. However, issues such as communication to and from the GPU and cache coherency problems can degrade performance. GPU technology and performance is discussed more thoroughly in Section 4.6.

In addition to rapidly computing the patient dose, another goal of DOCTORS is to present the code in a user-friendly fashion. To this end, a graphical user interface (GUI) was developed. The GUI was built using the Qt5 graphics framework for the windows, buttons, and other necessary widgets. The GUI leads the user through the steps necessary to use the code in an intuitive way by using colors to indicate required and completed steps. The output is then plotted graphically as well as sent to an ASCII text file for more advanced postprocessing. Utilization of Qt5 also makes the code portable, it can be compiled on either Linux or Windows operating systems with no changes to the code. More details about the GUI are included in Section 4.5.

Overall, good qualitative agreement was found between the reference code, MCNP6, and DOCTORS. The primary source of error is believed to be the angular treatment of the transport which is causing the flux inside the patient to be underestimated while the flux at the very periphery is overestimated.

In the future, DOCOTRS may be extended to other application domains such as dose estimation of patients receiving radiation therapy. Often, before the procedure, a time dependent, 4D CT scan of the patient is taken so that radiologists can account for breathing patterns during administration of the treatment [Pan et al., 2004]. With further development, this methodology may enable real time dose computation as the treatment is administered. This would be greatly beneficial to both patients and the doctors administering the procedure.

The remainder of this work is organized into five sections. Section 2 summarizes the existing literature pertinent to this work. Section 3 lays the mathematical foundation for each of the major components of DOCTORS. Section 4 gives a overview of the implementation strategies used to transform the mathematical framework in Section 3 into code that can execute on a modern computing platform. Section 5 summarizes the results obtained. It quantifies both the dosimetric accuracy and runtime of the DOCTORS code. Finally, Section 6 summarizes the entire work and provides some concluding remarks. Afterward, an appendix provides additional information that was not included in the main text but may still be helpful to a reader.

2. LITERATURE REVIEW

This section summarizes the literature reviewed in preparation of this work. The literature review is split into multiple sections, each highlighting relevant works pertaining to a particular component or methodology behind DOCTORS. The first two sections cover current dose estimation techniques and steps necessary for preprocessing data. Section 2.3 covers historical and recent work in discrete ordinate solution methodology, the methodology used by DOCTORS. The remaining sections give an overview of GPU hardware and other implementation facets.

2.1. CT DOSE ESTIMATION

As discussed in Section 1, the dose a patient receives from both diagnostic CT procedures and radiation therapy is of concern to the medical community. However, dose quantification is not currently patient specific. Measuring the dose in a patient directly is impossible, instead the air kerma (measured in Gy), which measures the x-ray beam intensity, and can be measured directly with ionization chambers [Wolbarst, 2005]. As the x-ray beam passes through the patient, it deposits energy. The energy deposition per unit mass is the absorbed dose (measured in Gy) which is then weighted by the radiation type to give the equivalent dose (measured in Sv). The equivalent dose quantifies the biological detriment to the patient from the procedure. The effective dose (measured in Sv) accounts for the radiosensitivity of each organ and is the most meaningful measure of risk to the patient. Any two procedures that result in the same effective dose to the patient carry the same acute and long-term risks. Any procedure that gives an effective dose to a patient has the same risk as any procedure that would give the same whole body dose.

Unfortunately, neither the absorbed dose, equivalent dose, nor effective dose to a patient can be measured directly. Instead, the CT Dose Index (CTDI) is measured using a pencil shaped ionization chamber at the center of a standardized acrylic phantom placed at the scanner isocenter [Huda and Mettler, 2011]. The CTDI quantifies the equivalent dose the patient receives from a single slice of the CT scan protocol. The CTDI does *not* capture patient-specific papameters, but rather quantifies the amount of radiation emitted by the source.

The CTDI does not provide the dose from an entire procedure, but rather a single slice. The dose from a procedure is obtained from the dose-length-product (DLP) which is the product of the CTDI and the axial scan length. The DLP is related to the effective dose to the patient for a particular procedure by a conversion factor that is dependent on many variables including the anatomical region scanned, the x-ray tube voltage, and the patient's age, gender, and size [Huda and Mettler, 2011]. For example, Lau et al. [2016] found that severly obese patients received twice as much dose from CT procedures as others on average. Some patients received as much as five times the typical dose.

The need for a complete understanding of the transport and scatter through the patient motivates full transport solutions. Typical solution modalities involve three key steps: (1) interpret the user input and prepare the solver, (2) run the transport solution to compute the scalar flux, and (3) convert the scalar flux to a dose. The first step is completely code dependent. The second and third steps of all major solvers falls into one of a few categories.

Nearly all transport solutions fall into one of two categories: Monte Carlo and deterministic. Monte Carlo codes are generally known for their accuracy; they are able to sample the problem and attain arbitrarily high precision given sufficient runtime. They also take advantage of combinatorial geometry allowing a user to produce geometrically complex 3D structures. Deterministic methods are generally much faster than their Monte Carlo counterparts but do not benefit from excess runtime. They also consume vastly more resources, particularly random-access memory (RAM). However, in regularly structures meshes, deterministic methods can greatly outperform Monte Carlo. A large number of spatial regions slows down the particle transport in Monte Carlo and increases the RAM required to store the numerous voxels.

Codes such as EGS4, its improved versions EGSnrc, and EGS5 [Nelson and Field, 2007], and PENELOPE [Salvat, 2015] are early Monte Carlo that have shown very good comparison to experimental results. However, these codes were developed primarily for shower simulations in high energy physics.

Monte Carlo codes such as TOPAS [Perl et al., 2012], Geant4 [Agostinelli et al., 2003], and MCNP6 [Goorley et al., 2016] are the current gold standard in computational dosimetry [Jia et al., 2012]. This work compares results to those obtained from MCNP6. MCNP6 is a Monte Carlo code developed for general purpose particle transport and is well validated. MCNP6 was selected for comparison because it is so well validated and the input files can be easily generated procedurally. Codes whose input such as Geant4 do not lend themselves easily to procedural code generation are difficult to benchmark precisely against for arbitrary computational phantoms.

The alternative to Monte Carlo is deterministic solutions. The best known solution techniques include the discrete ordinate method, the finite element method, and the method of characteristics. The discrete ordinate method is employed by DOCTORS and the methodology is covered independently in Section 2.3. Other deterministic methods are summarized here.

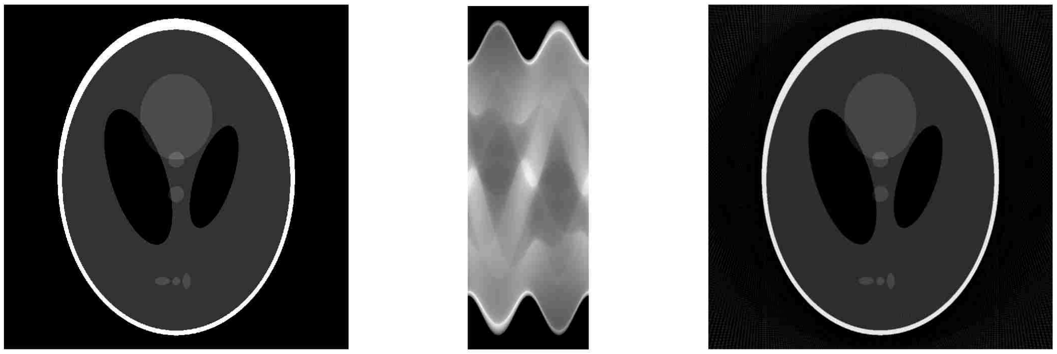
The method of characteristics solves the characteristic form of the Boltzmann transport equation which is defined along a single direction (called the characteristic). Along that direction, the 3-dimensional gradient operator found in the full Boltzmann equation (see Section 3.1.1) becomes a 1-dimensional derivitive. This reduction gives rise to an analytical solution along the characteristic [Askew, 1972]. Sufficient characteristic solutions solved simulateously gives the solution over the entire problem domain. Currently, though, MOC has not been extended to full 3D, instead, hybrid methods such as the NEM-S_N technique employed by DeCART have been used. DeCART uses 2D MOC to solve planar slices of reactor transport problems and links the slices using diffusion [Hursin et al., 2014]. Another MOC implementation is the TRANSMED code which was used to compute external beam therapy dose profiles accurately [Williams et al., 2003]. Method of characteristic solutions have been extended to the time dependent domain by Hoffman and Lee [2016].

However the energy dependent flux is computed, it must afterward be converted to dose. Monte Carlo codes have been used in the past to compute flux-to-dose conversion factors for a reference person [ICRP, 2010]. The values provided by ICRP [2010] are energy dependent conversion factors, $H(E)$ that convert flux to effective dose for a particular orientation. The dose, D is then computed

$$D = \int_0^\infty \varphi(E)H(E)dE \tag{2.1}$$

where φ is the scalar flux. The weakness of this method is that even though the flux may be computed to be patient specific, the weighting factors are evaluated using a standard reference man phantom.

Alternatively, codes can compute the dose deposition directly during the transport solution. The simplest example of this is in Monte Carlo codes which tally the energy deposited by each collision rather than track length in voxels which estimates the absorbed dose. Since the particle type that caused the scatter is always, known, the equivalent dose is simple to compute. However, in order to compute the effective dose, which is organ weighted, the organ to which the voxel of interest belongs must be known. In general, this is a very difficult problem with no fully automated solutions without significant additional information about human anatomy and the scan procedure employed. Therefore, this dose estimation technique is rarely employed in practice.



(a) (b) (c)

Figure 2.1. CT reconstruction of a phantom. (a) The original phantom. (b) The sinogram produced by 360 projections. (c) The filter backprojection reconstruction.

2.2. CT PHANTOM GENERATION

In order for the proposed methodology to be viable, the patient's CT phantom must be available. The phantom is reconstructed from the CT data. Though it is not the focus of this work, a brief summary of reconstruction algorithms is given here.

Fundamentally, all reconstructions algorithms are based on a thin parallel beam of x-rays rotating axially about a patient to produce a sinogram. The inverse radon transform backprojects each row of the sinogram back to physical space. Filtering each row in Fourier space and then compositing all layers together recreates the image [Mersereau and Oppenheim, 1974]. Figure 2.1 shows a phantom, the sinogram created from it, and the final backprojected recreation. This methodology has been extended to fan beams [Besson, 1999] and cone beams [Turbell, 2001]. Besides the filtered backprojection (FBP) method, iterative reconstruction techniques can also be applied to greatly improve the image reconstruction quality [Pontana et al., 2011].

Along any particular ray, monoenergetic particles will attenuate through a homogenous media according to the Beer-Lambert law given in Eq. 2.2 [Lamarsh and Baratta, 2001] resulting in a small fraction of the source particles reaching the detector. This phenomena leads to an image of the attenuating media but also results in the detector receiving information about the attenuation coefficient, μ, along that ray. With sufficient rays of varying directionality, the entire object can be quantified with respect to its spatially distributed attenuation coefficient.

$$I(x) = I_0 e^{-\mu x} \tag{2.2}$$

The reconstruction populates the CT mesh with CT numers, also known as Hounsfield units. These values are related to the attenuation coefficient of the material represented by the voxel. Traditionally, a CT number of zero corresponds to the attenuation of water while -1000 corresponds to dry air though some variations do exist [du Plesis et al., 1998, Saw et al., 2005].

Schneider et al. [1996] proposed a method to utilize different phantom materials to calibrate a HU-to-material conversion. He used only six different materials that descriminated between fat, water, muscle, and three densities of bone. From the calibration curve, he computed proton stopping powers accurate to within 1-2%.

du Plesis et al. [1998] used a methodology similar to Schneider's but instead of performing experiments with phantoms, he used the ITS3 [Halbleib et al., 1992] Monte Carlo code to generate data. He identified 16 major tissue types in the body and classified them into seven dosimetrically equivalent categories. The resulting algorithm transormed HU directly into material and density values. Shortly thereafter, Schneider et al. [2000] completed a similar study but proposed considering all materials as a mixture of tissue and bone to varying degrees. However, the proposed algorithm fails to work well in soft tissue regions due to the presence of three components (water, fat, and muscle).

Many of the earlier studies in HU-to-material conversions were done with high energy (typically 6-8 MeV) photon beams used for treatment [Kim et al., 2014] [Vanderstraeten et al., 2007]. Saw et al. [2005] used a phantom with 17 inserts representing different dosimetric tissues.

Ottosson and Behrens [2011] extended the methodology provided by Schneider et al. [1996] to provide 19 dosimetric groups of materials in the diagnostic energy domain independent of the particular scanner used to generate the data. The DOCTORS code relies on his proposed 19-group model. Table 2.1 shows the mapping of CT number to material composition (reproduced from Ottosson and Behrens [2011]).

2.3. DISCRETE ORDINATES

The dsicrete ordinates method dates back to Chandrasekahar [1950] who used it for radiation transport in atmospheres, yet it remains one of the most prominent solution modalities for radiation transport in use today. A comprehensive review of discrete ordinate methods is given by Lewis and Miller [1993]. Additional historical references include Carlson [1953] Lanthrop and Carlson [1965].

The oldest major discrete ordinate method implementation is DORT which was superceded by its 3D counterpart TORT [Rhoades and Simpson, 1997]. Since TORT, new discrete ordinate method implementations have laregly been extensions allowing more advanced computation or updates to modernize the code.

Discrete ordinate methods are almost always done on a Cartesian grid, though alternative derivations on general geometry do exist [DeHart, 1993]. Extension to unstructured tetrahedral meshes was done by Wareing et al. [1998].

Walters [2015] extended the discrete ordinates method to employ an adaptive collision source. Similarly, Ahrens [2015] wrote an algorithm that allows the quadrature to adapt to the energy group. Both of these methods improve the angular refinement. Ibrahim et al. [2015] added an adaptive mesh refinement. The adaptive mesh refinement allowed the code to automatically refine areas of interest where the flux was rapidly changing. This reduced the error in those regions reducing the computation time needed to reach a particular uncertainty level. An overview of other multigrid reduction schemes is given by Lee [2012]. Efremenko et al. [2013] enumerates additional acceleration techniques available for discrete ordinates that can increase speed by 15-30%.

Denovo is a massively parallel general-purpose discrete ordinate solver developed by Evans et al. [2010] to replace TORT. Denovo uses modern programming standards and utilizes the Exnihilo package for data processing and solution methods [Evans et al., 2006].

To overcome ray effect artifacts (discussed more thouroughly in Section 3.2), discrete ordinate solutions often employ a raytracing algorithm to compute the uncollided flux. The uncollided flux is then computed using the full discrete ordinate method. Raytracing algorithms have been developed for numerous discret ordinate codes include RAY3D [Ying et al., 2015] and ATTILLA [Wareing et al., 1998] which utilizes unstructured tetrahedral meshes.

2.4. RAYTRACING

Many raytracing algorithms are based on the Bresenham line raserizing algorithm which was designed for graph plotting hardware [Bresenham, 1965]. That algorithm was intended to produce lines acceptable for human interpretation and no guarantee is made that all pixels along the line will be identified making this class of algorithms inappropriate for dosimetric raytracing since attenuation through some voxels would be skipped. Some extensions have been added to it that guarantee passage through all voxels though [Liu et al., 2004].

Cleary and Wyvill [1988] proposed a voxel traversal algorithm in which he initially identifies the voxel containing the starting point of the ray, and transport it to the first surface of its bounding voxel it would cross. This is done by computing the distance traveled to cross each enclosing surface. The surface with the least distance is the first surface crossed. Therefore, the particle is advanced to the intersection point and considered inside the next cell, having crossed the surface. This process is repeated until a termination criteria is met.

This work uses the raytracing algorithm proposed by Amanatides and Woo [1987] which is very similar to the method proposed by Cleary and Wyvill [1988] but based on the parametric equation of a line, $\boldsymbol{u} + \boldsymbol{v}t$ which makes implementation in 3D simple. Since its develpment, many improvements and extensions have been made for alternate geometries such as rhombic dodecahedra [He et al., 2013] for volume rendering but the core algorithm remains state of the art for floating-point arithmetic. On systems that are highly efficient at computing integer arithmetic, alternatives exist which are faster [Liu et al., 2004] but they do not outperform on hardware optimized for floating-point operations.

2.5. GPU ACCELERATION

A graphics processing unit (GPU) is similar to a central processing unit (CPU)in that it executes instructions to process data. However, it differs in that a CPU may have up to 22 physical cores, each of which can operate independently, a GPU has up to thousands of cores, but they cannot operate independently. Instead, cores execute simulaneously in groups called warps. All cores in a warp execute the same instruction at the same time as all others.

Two major GPU manufacturers control the vast majority of the market, Nvidia and AMD. In 2007, Nvidia released its Compute Unified Device Architecture (CUDA) language. CUDA allows code written in C/C++/FORTRAN to communicate with a GPU with minimal additional code. Therefore, CUDA was selected for the GPU acceleration of DOCTORS. Therefore, only Nvidia GPUs are considered in this work.

In GPU texts, the GPU itself is referred to as a device and the CPU or other hardward running the GPU is referred to as the host. The device has its own instruction set which requires a special compiler. The device also has its own onboard memory so data must be copied from the host to the device and then back.

Conceptually, code run on the device is split into block on a grid, each block then runs multiple concurrent computations, each on a thread. In hardware, threads execute in a warp (32 to 128 streaming multiprocessors depending on the GPU architecture).

Table 2.1. Water Regions

Media	CT Range	H	C	N	O	Na	Mg	P	S	Cl	Ar	K	Ca
Air	-950 to -100			75.7	23.3						1.3		
Lung	-1000 to -950	10.3	10.5	3.1	74.9	0.2		0.2	0.3	0.3		0.2	
Adipose	-100 to 15	11.2	50.8	1.2	36.4	0.1			0.1	0.1			
Connective	15 to 129	10.0	16.3	4.3	68.4	0.4			0.4	0.3			
Bone 1	129 to 200	9.7	44.7	2.5	35.9			2.3	0.2	0.1		1.0	4.5
Bone 2	200 to 300	9.1	41.4	2.7	36.8		0.1	3.2	0.2	0.1		1.0	6.3
Bone 3	300 to 400	8.5	37.8	2.9	37.9		0.1	4.1	0.2	0.1		1.0	8.2
Bone 4	400 to 500	8.0	34.5	3.1	38.8		0.1	5.0	0.2	0.1		1.0	10.0
Bone 5	500 to 600	7.5	31.6	3.2	39.7		0.1	5.8	0.2	0.1			11.6
Bone 6	600 to 700	7.1	28.7	3.4	40.4		0.1	6.6	0.2	0.1			13.1
Bone 7	700 to 800	6.7	26.7	3.5	41.2		0.2	7.2	0.3				14.4
Bone 8	800 to 900	6.3	24.7	3.7	41.8		0.2	7.8	0.3				15.7
Bone 9	900 to 1000	6.0	22.7	3.8	42.4		0.2	8.4	0.3				16.8
Bone 10	1000 to 1100	5.6	20.7	3.9	43.0		0.2	8.9	0.3				17.9
Bone 11	1100 to 1200	5.3	18.7	4.0	43.5		0.2	9.4	0.3				18.9
Bone 12	1200 to 1300	5.1	16.7	4.1	44.0		0.2	9.9	0.3				19.8
Bone 13	1300 to 1400	4.8	15.7	4.2	44.4		0.2	10.3	0.3				20.7
Bone 14	1400 to 1500	4.6	13.7	4.2	44.9		0.2	10.7	0.3				21.5
Bone 15	> 1500	4.3	12.7	4.3	45.3		0.2	11.1	0.3				22.2

3. METHODS

This chapter summarizes the solution methodology employed by DOCTORS. The techniques used to compute the collided and uncollided fluxes are described in separate sections. After the flux solution methodology is explained, the methods used for source generation and flux-to-dose conversion are covered.

3.1. DISCRETE ORDINATE METHODS

The discrete ordinate solver computes the flux distribution inside the CT mesh given a source, physical geometry information, and other associated solver parameters such as quadrature and energy discretization. Discrete ordinates is a deterministic solution to the linear Boltzmann equation.

3.1.1. The Boltzmann Equation. The linear Boltzmann equation (LBE) is generally true of any particle for which outside forces (electromagnetic or gravitational) are either not present or negligible and particle-particle collisions are insignificant. Therefore, the LBE is generally not valid for charged particles or systems whose particle density is on the order of the confining medium. In such systems, particle-particle interactions may not be negligible requiring the non-linear Boltzmann equation.

In general, the LBE is valid for multiplying media (such as neutrons passing through fuel in a reactor), but the variant considered in this work omits such terms since only photons in medical systems are of concern. Solutions to the LBE can also be used to solve criticality eigenvalue problems or to produce adjoint parameters for other codes. Such solutions are not considered in this work.

The steady state form of the LBE can be readily derived from simple intuition. In the steady state, particle production and removal must be equal. In any volume, three production mechanisms are present. Particles can (1) stream into the volume across a bounding surface,

(2) inscatter from another energy-direction component of phase space, or (3) be produced directly by the external source. At the same time, three removal terms are present, particles can (1) stream out of the volume of interest by crossing a bounding surface, (2) be absorbed and vanish entirely, or (3) outscatter to a energy-direction component of phase space other than the one of interest.

The downscatter and absorption removal terms are typically grouped together as

$$\Sigma_t = \Sigma_a + \Sigma_s \tag{3.1}$$

where Σ_t, Σ_a, and Σ_s are the macroscopic total, absorption, and scatter cross sections respectively. The removal of particles from interactions (scatter and absorption) per volume is then

$$\Sigma_t(\boldsymbol{r}, E)\psi(\boldsymbol{r}, E, \hat{\Omega}) \tag{3.2}$$

where ψ is the angular flux. The production and removal streaming operators are combined as

$$\hat{\Omega} \cdot \nabla\psi(\boldsymbol{r}, E, \hat{\Omega}) \tag{3.3}$$

and included as a removal term since the normal is defined pointing outward from the volume. The normal sign convention results in particles streaming out being positive and those streaming in becoming negative.

The inscatter to a particular energy and direction is computed by integrating over all other energies and directions

$$\int_{4\pi}\int_0^\infty \Sigma_s(\boldsymbol{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega})\psi(\boldsymbol{r}, E', \hat{\Omega}')dE'd\hat{\Omega}'. \tag{3.4}$$

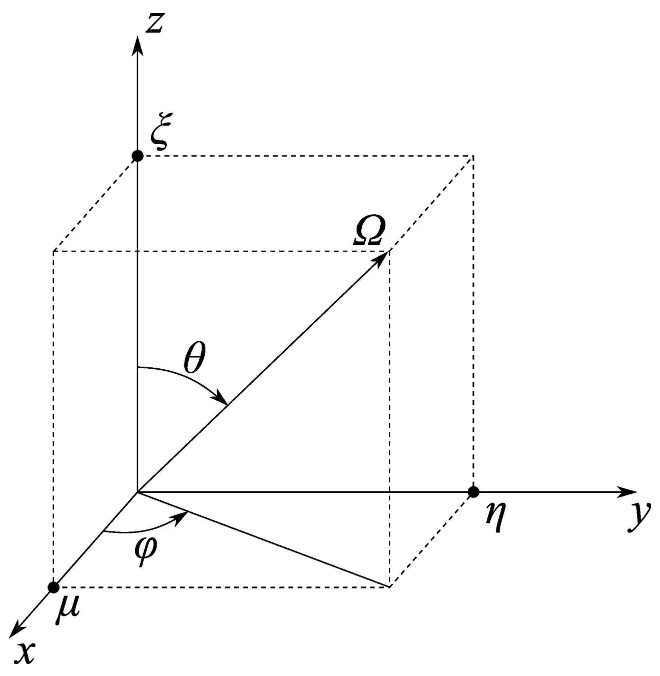


Figure 3.1. The coordinate system used in DOCTORS. Given an arbitrary direction, Ω, μ, η, and ξ are its direction cosines with respect to the x, y, and z axes respectively. φ is the azimuthal angle (with respect to x) and θ is the polar angle (with respect to z).

An external source S produces particles per volume. Combining the removal terms on the left and the production terms on the right yields

$$\begin{aligned}&\left[\hat{\Omega}\cdot\nabla+\Sigma_t(\boldsymbol{r},E)\right]\psi(\boldsymbol{r},E,\hat{\Omega}) = \\ &\int_{4\pi}\int_0^\infty \Sigma_s(\boldsymbol{r},E'\rightarrow E,\hat{\Omega}'\rightarrow\hat{\Omega})\psi(\boldsymbol{r},E',\hat{\Omega}')dE'd\hat{\Omega}'+S(\boldsymbol{r},E,\hat{\Omega})\end{aligned} \tag{3.5}$$

which is the LBE. The following sections show how Eq. 3.5 is discretized and solved computationally.

3.1.2. Angular Discretization. The coordinate system used in DOCTORS is shown in Figure 3.1. A discrete set of N_a angles ($\Omega_a, \quad a = 0\ldots N_a - 1$) is selected to represent continuous directional space. Particles are transported only along these discrete directions from each voxel to adjacent voxels.

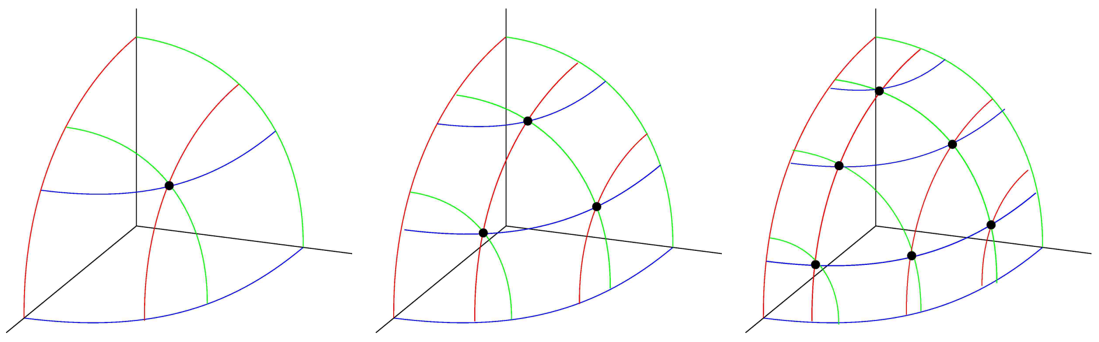


Figure 3.2. Some various S_N quadratures. The black dots indicate the location of a ray. (a) S_2 (b) S_4 (c) S_6

The rotation symmetrical (S_N) quadrature is implemented in DOCTORS. From amongst many different quadrature sets, the rotation symmetrical quadratures were selected because they are easy to implement, rotationally symmetric, and used in production discrete ordinate solvers [Evans et al., 2010] which enables simple, direct comparison to other solvers. Arbitrary quadratures are permissible in a plain text file supplied by the user enabling other quadratures.

The S_N quadrature ensures 90 degree symmetry about any axis. In order to ensure the rotational symmetry, the S_N quadrature has a single degree of freedom. The LQ_N quadrature (which is the specific S_N implemented in DOCTORS) uses the remaining degree of freedom to minimize error caused when integrating Legendre polynomials since that class of polynomials are used to represent anisotropy in scattering (see Section 3.1.8). For a more thorough discussion of quadrature generation, see Lewis and Miller [1993]. Figure 3.2 shows the simplest three quadratures implemented in DOCTORS.

To include a user defined quadrature, the plain text file should be formatted such that it contains four columns similar to Table 3.1 (without any headings). No checks are performed to ensure that octants are covered, this allows simple testing quadratures that utilize only a single direction to be used. Values for μ, η, and ξ do not necessarily need to add to unity as they will be automatically normalized by DOCTORS. Likewise, the weights will be normalized such that their summation is unity. However, user defined directions *should not* be parallel to any major axis. Doing so may result in undefined behavior. Also, after normalization, no two rows should have identical values for μ, η, and ξ such as the last two rows of Table 3.2, nor may any weights be negative. The data in Table 3.1 would be read in and renormalized by DOCTORS resulting in data identical to Table 3.2.

Values are read in as a 32-bit IEEE-754 floating point number [IEEE, 2008] which ensures seven significant digits. Significant digits beyond this will be truncated. Note that Table 3.2 rounds four digits after the decimal.

Table 3.1. User Defined Quadrature Input

μ	η	ξ	Weight
1	0	0	1
0.999	0.001	0.001	1
1.0	1.0	1.0	1.3
0.02198	0.2987	.34520	.02226
.001	.001	.999	0.999
-1	1	-1	.2
-1.2	1.2	-1.2	0.1
-1	1	-1	.2

Table 3.2. User Defined Quadrature Interpretation

μ	η	ξ	Weight
1.0000	0.0000	0.0000	0.2074
1.0000	0.0010	0.0010	0.2074
0.5774	0.5774	0.5774	0.2696
0.0481	0.6536	0.7553	0.0207
0.5000	-0.5000	0.7071	0.0046
0.5774	-0.5774	-0.5774	0.2072
-0.5774	0.5774	-0.5774	0.0415
-0.5774	0.5774	-0.5774	0.0415

A given direction, $\hat{\Omega}$, is determined by its three cosine components:

$$\hat{\Omega} = \mu\hat{i} + \eta\hat{j} + \xi\hat{k} \tag{3.6}$$

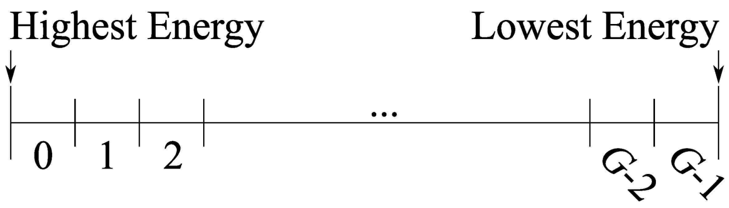


Figure 3.3. The energy grid structure used in DOCTORS. The highest energy group is group 0 and the lowest energy is group $G-1$.

where $\hat{i}$, $\hat{j}$, and $\hat{k}$ are the unit directions in the x, y, and z directions respectively. Therefore, any discrete direction $\hat{\Omega}_a$ can be expressed as $<\mu_a, \eta_a, \xi_a>$. The two forms will be used interchangeably.

Discretizing the LBE (Eq. 3.5) in angular space gives

$$\left[\hat{\Omega}_a \cdot \nabla + \Sigma_t(\boldsymbol{r}, E)\right] \psi_a(\boldsymbol{r}, E) = \\ \sum_{a=0}^{N_a-1} \int_0^\infty \Sigma_{s,a,a'}(\boldsymbol{r}, E' \rightarrow E) \psi_{a'}(\boldsymbol{r}, E') dE' \omega_a + S_a(\boldsymbol{r}, E) \tag{3.7}$$

where the subscript a denotes the direction number and ω_a is a weight associated with that direction. The weights are designed with the angles such that once a discrete quadrature is selected, integration over continuous space becomes integration in discrete space approximated by

$$\int_{4\pi} f(\hat{\Omega}) d\hat{\Omega} \approx \sum_{a=0}^{N_a-1} f(\hat{\Omega}_a) \omega_a = 1. \tag{3.8}$$

3.1.3. Energy Discretization. Continuous energy is discretized into G groups indexed from 0 to $G-1$ as illustrated in Figure 3.3. Therefore, Eq. 3.7 becomes

$$\left[\hat{\Omega}_a \cdot \nabla + \Sigma_t^g(\boldsymbol{r})\right] \psi_a^g(\boldsymbol{r}) = \sum_{a=0}^{N_a-1} \sum_{g'=0}^{G-1} \Sigma_{s,a,a'}^{g,g'}(\boldsymbol{r}) \psi_{a'}^{g'}(\boldsymbol{r}) \omega_a + S_a^g(\boldsymbol{r}) \tag{3.9}$$

where the group number is indexed by superscript g.

The user must provide cross section data in the form of an AMPX formatted binary file (Format details are provided in Section 4.2). The cross section data file contains group structure information. Group averaged cross section values from energy E_1 to E_2 are computed using Eq. 3.10 where $\sigma(E)$ is the continuous microscopic cross section. A weighting function, f, is used to weight the energies. The weighting function can be any of a number commonly used functions. The SCALE data libraries use a uniform function.

$$\sigma_G = \frac{\int_{E_1}^{E_2} f(E')\sigma(E')dE'}{\int_{E_1}^{E_2} f(E')dE'} \tag{3.10}$$

In photon based problems, upscatter is typically negligible. By the time photons become low enough in energy to upscatter significantly, they are no longer of dosimetric concern. Removing upscatter reduces Eq. 3.9 to

$$\left[\hat{\Omega}_a \cdot \nabla + \Sigma_t^g(\boldsymbol{r})\right] \psi_a^g(\boldsymbol{r}) = \sum_{a=0}^{N_a-1} \sum_{g'=g}^{G-1} \Sigma_{s,a,a'}^{g,g'}(\boldsymbol{r})\psi_{a'}^{g'}(\boldsymbol{r})\omega_a + S_a^g(\boldsymbol{r}) \tag{3.11}$$

which nearly identical; only the starting point for the scatter summation changes. Computationally, though, the lack of upscatter makes a significant difference. The highest energy group can be solved independently of all those below it. Once that group is solved, the next group can also be solved and so forth. In the presence of upscatter, the highest group depends on the solution to all groups below it so all energy groups must be solved simultaneously.

The group structure cannot be arbitrarily selected by DOCOTRS, instead it is determined by the external cross section data file loaded by the user. The energy group structure should be selected appropriately for the problem. The cross sections currently included with DOCTORS are those available from the SCALE 6.2 distribution. Those cross sections are designed for light water reactor analysis and are not well suited to medical physics applications. This is one of the limitations of the current version of DOCOTRS. The

SCALE cross section data also contains neutron data which is useless for the CT dosimetry DOCTORS performs. More details of the cross section parsing and generation are included in Section 4.2 and 4.3 respectively.

3.1.4. Spatial Discretization. The problem domain is split into an evenly spaced Cartesian grid. Uniform grid spacing is not required by discrete ordinate methods though it is helpful and can speed up the computation.

The problem domain for a CT voxel phantom is a regular, rectangular parallelepiped of dimension $D_x \times D_y \times D_z$, the values of which are known based on the CT setup provided by the user. The mesh is partitioned into N_x, N_y, and N_z evenly spaced bins along each major direction as shown in Figure 3.4. The total number of voxels, N_V is easily computed from the number of bins in each major direction:

$$N_V = N_x N_y N_z. \tag{3.12}$$

The length of a voxel along the x-direction is computed by

$$\Delta x = \frac{D_x}{N_x}. \tag{3.13}$$

Analogous equations apply in the y and z directions as well.

Voxels can be indexed in one of two ways, either by their component indices or global index. Each voxel in the mesh has three components indices, one in each major direction, i_x, i_y, and i_z that range from 0 to $N_x - 1$, $N_y - 1$, or $N_z - 1$. However, this makes indexing cumbersome, so they are combined into a unique global index, i using

$$i = i_x(N_z + N_y) + i_y N_z + i_z. \tag{3.14}$$

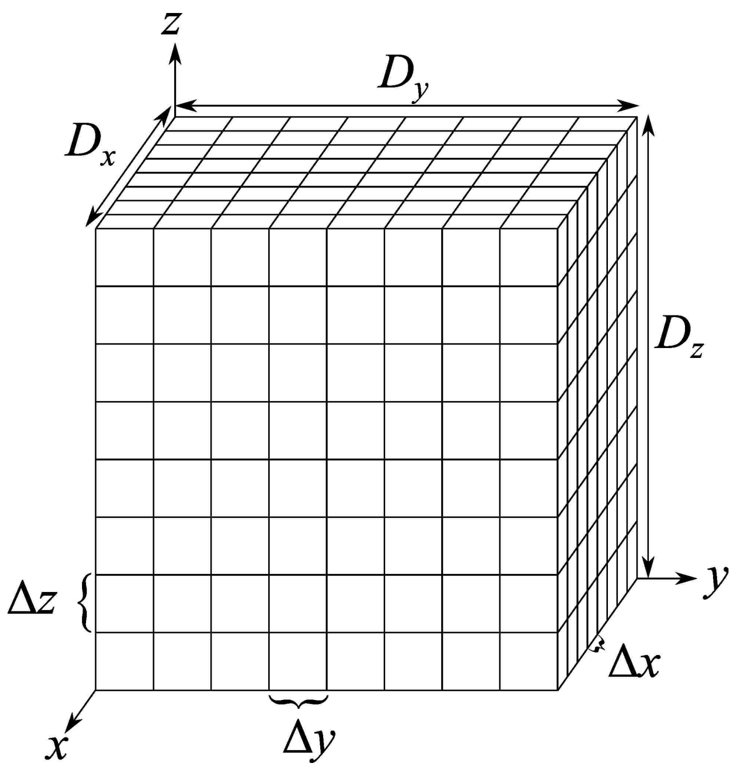


Figure 3.4. The spatial mesh imposed on the problem domain.

In general, this is known as "flattening" a matrix into a one-dimensional vector. More details about the flattening implementation used in DOCTORS is given in Section 4.1. Using global indexing discussed, the fully discretized steady state LBE is

$$\left[\hat{\Omega}_a \cdot \nabla + \Sigma^g_{t,i}\right] \psi^g_{i,a} = \sum_{a=0}^{N_a-1} \sum_{g'=g}^{G-1} \Sigma^{g,g'}_{s,i,a,a'} \psi^{g'}_{i,a'} \omega_a + S^g_{i,a}. \tag{3.15}$$

3.1.5. Solution. To solve the fully discretized form of the Linear Boltzmann Equation given in Eq. 3.15, the gradient operator must first be computed numerically. Recall that Ω_a can be written in vector notation using its cosine components as defined in Eq. 3.6. Also recall the definition of the gradient:

$$\nabla f(x,y,z) = \frac{\partial f(x,y,z)}{\partial x}\hat{i} + \frac{\partial f(x,y,z)}{\partial y}\hat{j} + \frac{\partial f(x,y,z)}{\partial z}\hat{k}. \tag{3.16}$$

To a first order approximation, a partial derivative is computed

$$\frac{\partial f(\zeta)}{\partial \zeta} \approx \frac{f(\zeta + \Delta\zeta/2) - f(\zeta - \Delta\zeta/2)}{\Delta\zeta} \tag{3.17}$$

with respect to ζ. Combining Eq. 3.16 and 3.17, the $\hat{\Omega} \cdot \nabla\psi$ term can be rewritten as:

$$\begin{aligned}\Omega \cdot \nabla\psi &=< \mu, \eta, \xi > \cdot < \frac{\partial\psi}{\partial x}, \frac{\partial\psi}{\partial y}, \frac{\partial\psi}{\partial z} > \\ &= \frac{\partial\psi}{\partial x}\mu + \frac{\partial\psi}{\partial y}\eta + \frac{\partial\psi}{\partial z}\xi \\ &\approx \frac{\psi(x + \Delta x/2, y, z) - \psi(x - \Delta x/2, y, z)}{\Delta x}\mu \\ &+ \frac{\psi(x, y + \Delta y/2, z) - \psi(x, y - \Delta y/2, z)}{\Delta y}\eta \\ &+ \frac{\psi(x, y, z + \Delta z/2) - \psi(x, y, z - \Delta z/2)}{\Delta z}\xi.\end{aligned} \tag{3.18}$$

The flux variables $\psi^g_{i,a,x,in}$, $\psi^g_{i,a,x,out}$, $\psi^g_{i,a,y,in}$, $\psi^g_{i,a,y,out}$, $\psi^g_{i,a,z,in}$, and $\psi^g_{i,a,z,out}$ are *surface* averaged flux values while $\psi^g_{i,a}$ is a *volume* averaged flux value.

The gradient is computed with respect to the direction $\hat{\Omega}$, thus Eq. 3.18 is valid only for the first octant ($\mu, \eta, \xi > 0$). In other octants, a new gradient must be used. Rather than enumerating all eight variants, Eq. 3.18 can be generalized to

$$\Omega \cdot \nabla\psi \approx \frac{\psi_{x,out} - \psi_{x,in}}{\Delta x}\mu + \frac{\psi_{y,out} - \psi_{y,in}}{\Delta y}\eta + \frac{\psi_{z,out} - \psi_{z,in}}{\Delta z}\xi \tag{3.19}$$

where

$$\psi_{x,out} = \begin{cases} \psi(x + \Delta x/2, y, z), & \mu > 0 \\ \psi(x - \Delta x/2, y, z), & \mu < 0 \end{cases} \tag{3.20}$$

and

$$\psi_{x,in} = \begin{cases} \psi(x - \Delta x/2, y, z), & \mu > 0 \\ \psi(x + \Delta x/2, y, z), & \mu < 0 \end{cases} \tag{3.21}$$

and $\psi_{y,out}$, $\psi_{y,in}$, $\psi_{z,out}$, and $\psi_{z,in}$ are similarly defined but with respect to η and ξ respectively.

Using the generalized approximation given in Eq. 3.19, Eq. 3.15 can be rewritten as

$$\left[\frac{\psi^g_{i,a,x,out} - \psi^g_{i,a,x,in}}{\Delta x}\mu + \frac{\psi^g_{i,a,y,out} - \psi^g_{i,a,y,in}}{\Delta y}\eta + \frac{\psi^g_{i,a,z,out} - \psi^g_{i,a,z,in}}{\Delta z}\xi\right] + \Sigma^g_{t,i}\psi^g_{i,a}$$
$$= \sum_{a=0}^{N_a-1}\sum_{g'=g}^{G-1}\Sigma^{g,g'}_{s,i,a,a'}\psi^{g'}_{i,a'}\omega_a + S^g_{i,a}. \tag{3.22}$$

In order for the discretized LBE to be solved using Eq. 3.22, both the incoming and outgoing flux must be known. Therefore, in order to compute the average flux in a cell, some relationship between the average flux and the outgoing surface fluxes must be known. These values can be related by any one of many different mechanisms, the most common of which is the diamond difference approximation.

3.1.6. Diamond Difference Approximation. The diamond difference approximation, relates the incoming and outgoing surface fluxes to the cell averaged flux. The cell averaged flux is assumed to be the average of any two opposite surface fluxes. This is assumed to be the case in all three spatial dimensions leading to:

$$\begin{aligned}\frac{\psi^g_{i,a,x,in} + \psi^g_{i,a,x,out}}{2} &= \psi^g_{i,a}\\ \frac{\psi^g_{i,a,y,in} + \psi^g_{i,a,y,out}}{2} &= \psi^g_{i,a}\\ \frac{\psi^g_{i,a,z,in} + \psi^g_{i,a,z,out}}{2} &= \psi^g_{i,a}.\end{aligned} \tag{3.23}$$

Multiplying both sides of Eq. 3.23 by 2 and subtracting twice the incoming surface flux from each yields:

$$\begin{aligned}\psi^g_{i,a,x,out} - \psi^g_{i,a,x,in} &= 2\psi^g_{i,a} - 2\psi^g_{i,a,x,in}\\ \psi^g_{i,a,y,out} - \psi^g_{i,a,y,in} &= 2\psi^g_{i,a} - 2\psi^g_{i,a,y,in}\\ \psi^g_{i,a,z,out} - \psi^g_{i,a,z,in} &= 2\psi^g_{i,a} - 2\psi^g_{i,a,z,in}.\end{aligned} \tag{3.24}$$

The form of the left hand side of Eq. 3.24 is the same as in Eq. 3.22. This allows Eq. 3.24 to be used to make a substitution in Eq. 3.22 that removes the dependence on the outgoing flux. After the substitution:

$$\begin{aligned}&\left[\frac{2\psi^g_{i,a} - 2\psi^g_{i,a,x,in}}{\Delta x}\mu + \frac{2\psi^g_{i,a} - 2\psi^g_{i,a,y,in}}{\Delta y}\eta + \frac{2\psi^g_{i,a} - 2\psi^g_{i,a,z,in}}{\Delta z}\xi\right] + \Sigma^g_{t,i}\psi^g_{i,a} \\ &= \sum_{a=0}^{N_a-1}\sum_{g'=g}^{G-1}\Sigma^{g,g'}_{s,i,a,a'}\psi^{g'}_{i,a'}\omega_a + S^g_{i,a}.\end{aligned} \tag{3.25}$$

Rearranging Eq. 3.25 to factor out the volume averaged flux term yields:

$$\begin{aligned}&\left[\frac{2\psi^g_{i,a}}{\Delta x}\mu + \frac{2\psi^g_{i,a}}{\Delta y}\eta + \frac{2\psi^g_{i,a}}{\Delta z}\xi\right] - \left[\frac{2\psi^g_{i,a,x,in}}{\Delta x}\mu + \frac{2\psi^g_{i,a,y,in}}{\Delta y}\eta + \frac{2\psi^g_{i,a,z,in}}{\Delta z}\xi\right] + \Sigma^g_{t,i}\psi^g_{i,a} \\ &= \sum_{a=0}^{N_a-1}\sum_{g'=g}^{G-1}\Sigma^{g,g'}_{s,i,a,a'}\psi^{g'}_{i,a'}\omega_a + S^g_{i,a}\end{aligned} \tag{3.26}$$

which can be directly solved for the volume averaged flux:

$$\psi^g_{i,a} = \frac{\sum_{a=0}^{N_a-1}\sum_{g'=g}^{G-1}\Sigma^{g,g'}_{s,i,a,a'}\psi^{g'}_{i,a'}\omega_a + \left[\frac{2\psi^g_{i,a,x,in}}{\Delta x}\mu + \frac{2\psi^g_{i,a,y,in}}{\Delta y}\eta + \frac{2\psi^g_{i,a,z,in}}{\Delta z}\xi\right] + S^g_{i,a}}{\frac{2\mu}{\Delta x} + \frac{2\eta}{\Delta y} + \frac{2\xi}{\Delta z} + \Sigma^g_{t,i}}. \tag{3.27}$$

The diamond difference approximation defined in Eq. 3.23 can also be rearranged to solve for each outgoing flux once the volume averaged flux is computed. The outgoing flux is:

$$\begin{aligned}\psi^g_{i,a,x,out} &= 2\psi^g_{i,a} - \psi^g_{i,a,x,in} \\ \psi^g_{i,a,y,out} &= 2\psi^g_{i,a} - \psi^g_{i,a,y,in} \\ \psi^g_{i,a,z,out} &= 2\psi^g_{i,a} - \psi^g_{i,a,z,in}.\end{aligned} \tag{3.28}$$

The outgoing flux values are then used as the incoming flux values for subsequent voxels.

Repeatedly iteration Eq. 3.27 will eventually yield the solution to the flux distribution regardless of the initial guess. Choosing an accurate guess can greatly accelerate the code and guide convergence on a more accurate solution as discussed in more detail in Section 3.2. Either way, some metric must be used to determine when to stop the iteration process.

3.1.7. Convergence Criteria. The typical convergence criteria used to determine whether or not to terminate further iteration is the maximum relative error from the previous iteration to the current iteration given in Eq. 3.29 over the phase space $\mathbb{R}$ where t is the current iteration number. Once the relative error in the parameter of interest, ϕ, is below some threshold, ϵ, the iteration stops and the solution is considered converged.

$$\max_{\mathbb{R}} \left\{ \frac{\phi_t - \phi_{t-1}}{\phi_t} \right\} < \epsilon \tag{3.29}$$

In some problems, the convergence criteria given in Eq. 3.29 is not reached for many iterations. In these cases, terminating the solution early rather than waiting for full convergence is appropriate. This is enforced by a second criterial given by Eq. 3.30. Whenever the number of iterations, t, exceeds the total permissible number of iterations, T, the iteration is terminated.

$$t > T \tag{3.30}$$

Under further investigation, the cause of the convergence failure that results in Eq. 3.30 being exercised often stems from floating point error. The flux magnitude fluctuates wildly in regions of very flow flux, such as the gantry. In these regions, the relative uncertainty remains much higher than in regions with smoother flux distributions.

In order to measure the uncertainty due to floating-point error, another convergence criteria was added. Rather than considering the relative change from iteration to iteration, the total absolute change is measured:

$$\varepsilon = \sum_{\mathbb{R}} |\phi_i - \phi_{i-1}|. \tag{3.31}$$

The termination criteria is given by Eq. 3.32. Computation of a value for ε requires two previous iteration, thus this condition cannot be met until the third iteration and thereafter.

$$\varepsilon_i > \varepsilon_{i-1} \tag{3.32}$$

3.1.8. Anisotropy Treatment. Scatter is assumed to be a function of only the scatter angle θ_s between the initial, $\hat{\Omega}_a$, and scattered, $\hat{\Omega}_{a'}$, directions as illustrated in Figure 3.5. The cosine of the scatter angle, μ_s is related by

$$\mu_s = \Omega_a \cdot \Omega_{a'} = \cos(\theta_s), \quad 0 \le \theta_s \le \pi. \tag{3.33}$$

Typically, at this point, discrete ordinate methods will describe the discrete angular flux, ψ using flux moments, ϕ. The motivation for using flux moments is that they improve computational performance and reduce memory overhead.

The data files containing the scatter cross sections store the information using Legendre polynomial expansion coefficients. The use of a Legendre expansion removes the dependence on the quadrature from the data. This allows any quadrature to work with any cross section dataset. A l-order Legendre polynomial is denoted by P_l. Details of the Legendre polynomials can be found in Appendix 7.2. The anisotropic scatter cross section is rewritten using a Legendre expansion as

$$\Sigma_{s,i,a,a'}^{g,g'} = \sum_{l=0}^{L} \frac{2l+1}{4\pi} \Sigma_{s,i,l}^{g,g'} P_l(\mu_s). \tag{3.34}$$

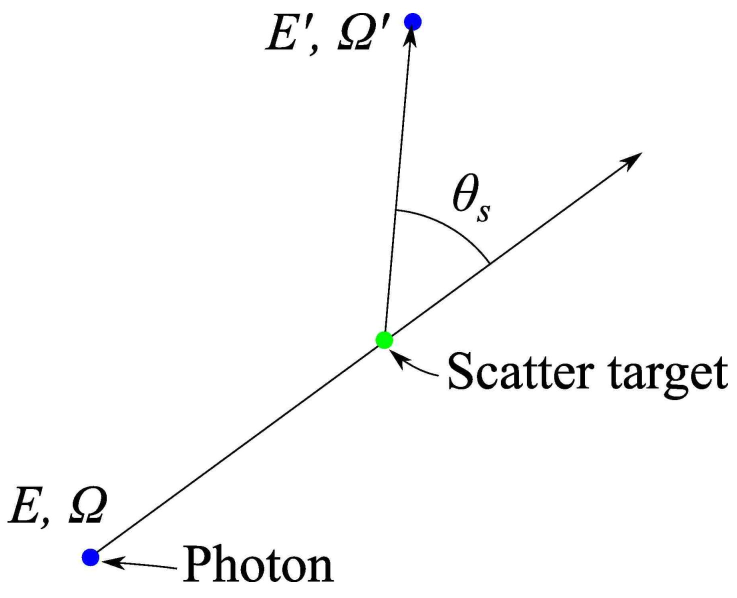


Figure 3.5. The scatter angle. A photon (blue) at energy E traveling in direction Ω, hits a stationary atom (green) and scatters into a new direction Ω' with a new energy E'.

An alternate anisotropy treatment is to use the Klein-Nishina formula [Shultis and and, 1950]. The Klein-Nishina formula describes the scattering cross section as

$$\sigma(E,\mu) \propto q^3 + q\mu^2 \tag{3.35}$$

where

$$q(E,\mu) = \frac{1}{1+(E/E_0)(1-\mu)}. \tag{3.36}$$

Figure 3.6 shows a comparison of the Klein-Nishina scattering cross section as a function of scatter angle versus the computed cross sections in DOCTORS using the Klein-Nishina formula given in Eq. 3.35. The slight discrepancy between the two is due to the postprocessing used to merge data from the DOCTORS output. The underlying trends are identical.

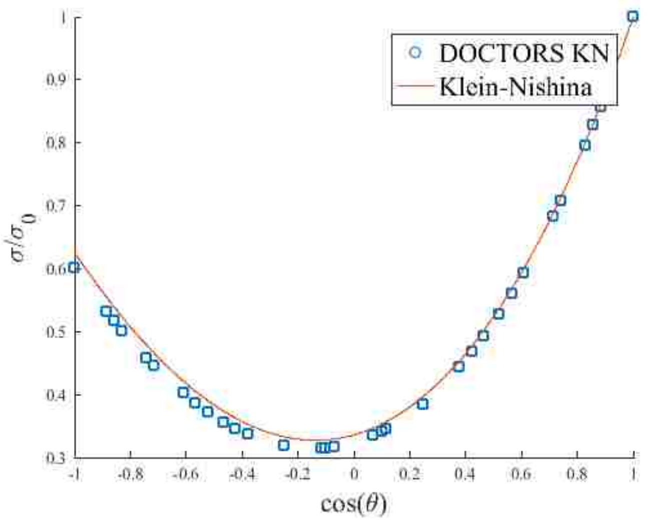


Figure 3.6. Comparison of the Klein-Nishina formula to the computed cross section values in DOCTORS using the Klein-Nishina anisotropy treatment.

3.2. UNCOLLIDED SOLUTION METHODS

One of the earliest problems with the discrete ordinate methods is the presence of ray effect. Ray effect arises due to the restriction of particle transport to a set of discrete directions. To illustrate ray effect, consider a point source streaming particles isotropically into vacuum. Since no scatter or absorption takes place in vacuum conditions, the particles can only stream and particles must be conserved. However, if particles are transported along a single discrete direction only as illustrated in Figure 3.7 ($\hat{\Omega} =< \sqrt{2}/2, \sqrt{2}/2 >$ in Figure 3.7 and 3.8) the phenomena observed in Figure 3.8 is observed. As the discretization becomes more refined, the isotropic source becomes more heavily biased in the $\hat{\Omega}$ direction. This phenomena is called ray effect and has been thoroughly studied by many [Mathews, 1999] [Tencer, 2016].

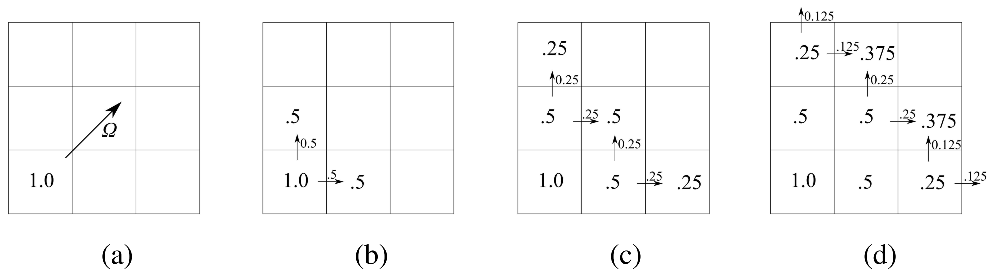


Figure 3.7. A simple illustration of how ray effect emerges from discrete angles. (a) Particles start in the lower left cell and travel along Ω. (b) Half travel to the cell above and half to the right. (c) Half of the particles in each cell move up and half to the right. The center cell gets particles from the cells below and left of it. (d) Some particles escape across the vacuum boundary.

All particles must stream from one voxel to adjacent voxels. The streaming is considered (in 2D) along $< \sqrt{2}/2, \sqrt{2}/2 >$ as shown in Figure 3.7. All particles in the initial cell must stream outward into the two cells touching in the $\hat{\Omega}$ direction. The particles in each voxel split, half go to the voxel above and the other half travel to the right. This process is illustrated in Figure 3.7.

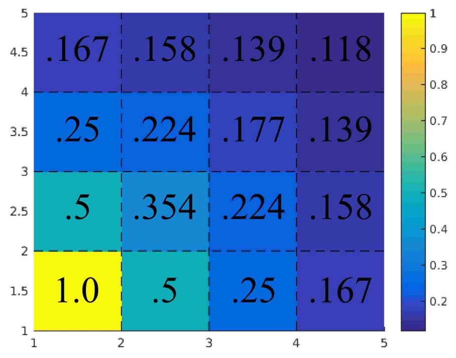
The solution to prevent ray effect is to utilize an analytical uncollided source term. The full flux solution is split into two subproblems. Equation 3.5 is rewritten as two equations solving for the uncollided flux, ψ_u, and collided flux, ψ_c independently:

$$\psi = \psi_u + \psi_c. \tag{3.37}$$

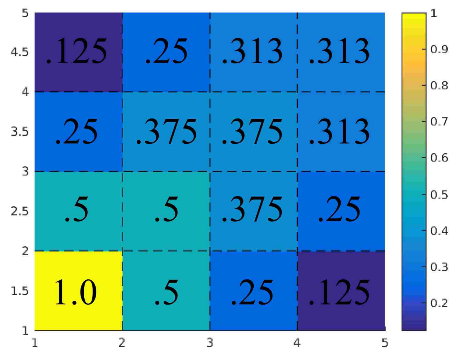
Both ψ_u and ψ_c obey the LBE. The uncollided flux

$$\left[\hat{\Omega} \cdot \nabla + \Sigma_t(\boldsymbol{r}, E)\right] \psi_u(\boldsymbol{r}, E, \hat{\Omega}) = S(\boldsymbol{r}, E, \hat{\Omega}) \tag{3.38}$$

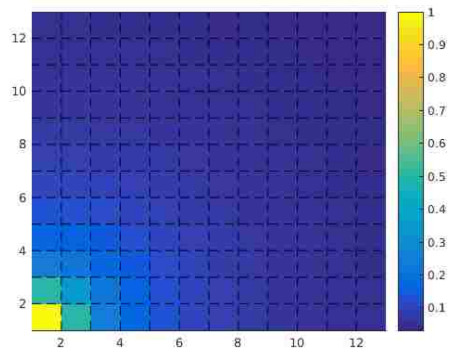
has no scatter term since scattered particles are not considered in the uncollided flux. The lack of a scatter term makes the uncollided flux (whose analytical solution is derived shortly) computable with a raytracing algorithm. The uncollided flux is then used to compute the



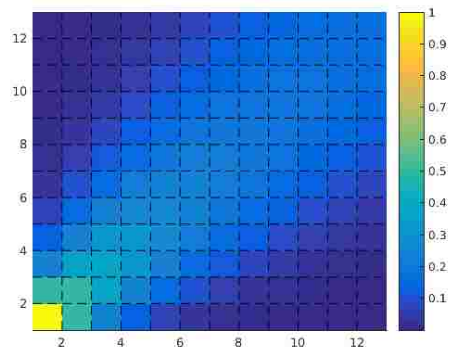
(a)



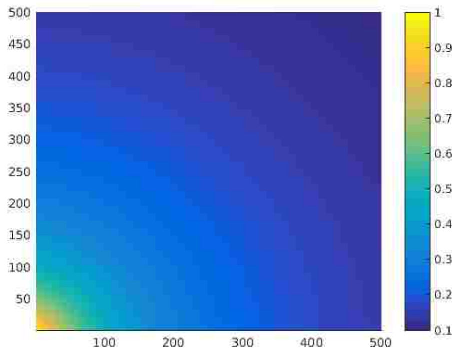
(b)



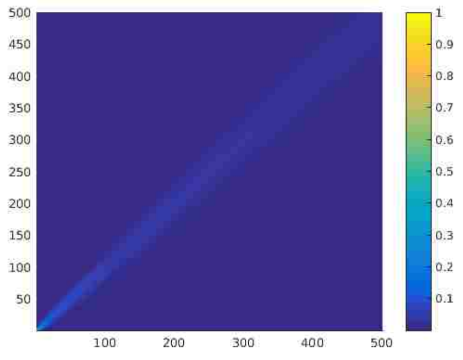
(c)



(d)



(e)



(f)

Figure 3.8. The ray effect artifact. The subfigures on the left (a, c, and e) show the correct solution modeled with $1/r^2$. The right subfigures (b, d, and f) show the corresponding solution with ray effect generated by the process described in Figure 3.7. The top row of subfigures uses 4 cells, the middle row uses 12, and the bottom row uses 500.

first collision source, S_u:

$$S_u = \int_{4\pi}\int_0^{\infty} \Sigma_s(\boldsymbol{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega})\psi(\boldsymbol{r}, E', \hat{\Omega}')dE'd\hat{\Omega}'. \tag{3.39}$$

which is then used to drive the collided flux in the same way the external source drives the uncollided flux distribution:

$$\begin{aligned}&\left[\hat{\Omega}\cdot\nabla + \Sigma_t(\boldsymbol{r}, E)\right]\psi_c(\boldsymbol{r}, E, \hat{\Omega}) = \\ &\int_{4\pi}\int_0^{\infty} \Sigma_s(\boldsymbol{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega})\psi_c(\boldsymbol{r}, E', \hat{\Omega}')dE'd\hat{\Omega}' + S_u(\boldsymbol{r}, E, \hat{\Omega}).\end{aligned} \tag{3.40}$$

Substituting Eq. 3.39 into S_u in Eq. 3.40 simplifies it to

$$\begin{aligned}&\left[\hat{\Omega}\cdot\nabla + \Sigma_t(\boldsymbol{r}, E)\right]\psi_c(\boldsymbol{r}, E, \hat{\Omega}) \\ &= \int_{4\pi}\int_0^{\infty} \Sigma_s(\boldsymbol{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega})\psi_c(\boldsymbol{r}, E', \hat{\Omega}')dE'd\hat{\Omega}' + \\ &\int_{4\pi}\int_0^{\infty} \Sigma_s(\boldsymbol{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega})\psi_u(\boldsymbol{r}, E', \hat{\Omega}')dE'd\hat{\Omega}' \\ &= \int_{4\pi}\int_0^{\infty} \Sigma_s(\boldsymbol{r}, E' \to E, \hat{\Omega}' \to \hat{\Omega})\psi(\boldsymbol{r}, E', \hat{\Omega}')dE'd\hat{\Omega}'.\end{aligned} \tag{3.41}$$

A rigorous derivation for the uncollided flux is not provided in this work. Instead, an intuitive approach is used. First, consider an isotropic point source in a homogeneous media. The (scalar) flux is well known to be

$$\varphi(r, E) = \frac{S(E)e^{-\mu(E)r}}{4\pi r^2} \tag{3.42}$$

where S is the source strength, μ is the attenuation coefficient in the material, and r is the distance traveled by the photon [CITE]. Expanding the problem to a full 3D model, the flux at $\boldsymbol{r}$ from a point source at $\boldsymbol{s}$ is

$$\varphi(\boldsymbol{r}, E) = \frac{S(E)e^{-\mu(E)|\boldsymbol{r}-\boldsymbol{s}|}}{4\pi|\boldsymbol{r}-\boldsymbol{s}|^2}. \tag{3.43}$$

The only non-intuitive step is the introduction of anisotropy. The 4π solid angle term in the denominator of Eq. 3.43 is generally expressed by

$$\int_{4\pi}\int_{0}^{\infty} S(E,\hat{\Omega})dEd\hat{\Omega} \tag{3.44}$$

but DOCTORS assumes S is normalized such that the integral in Eq. 3.44 is 4π. The flux is nonzero *only* in the direction of travel, namely in the unit direction $\boldsymbol{r}-\boldsymbol{s}$. Therefore, a Kronecker delta function (defined in Eq. 3.46) is used to produce the angular flux:

$$\psi(\boldsymbol{r},E,\hat{\Omega}) = \frac{S(E,\hat{\Omega})e^{-\mu(E)|\boldsymbol{r}-\boldsymbol{s}|}}{4\pi|\boldsymbol{r}-\boldsymbol{s}|^2}\delta\left(\hat{\Omega},\frac{\boldsymbol{r}-\boldsymbol{s}}{|\boldsymbol{r}-\boldsymbol{s}|^2}\right). \tag{3.45}$$

$$\delta(\boldsymbol{A},\boldsymbol{B}) = \begin{cases} 1, & \text{if } \boldsymbol{A}=\boldsymbol{B} \\ 0, & \text{otherwise} \end{cases} \tag{3.46}$$

Equation 3.45 can be expanded to accomodate any number of spatially distributed sources of arbitrary directionality given in Eq. 3.47 by integrating over the entire spatial domain, V

$$\psi_u(\boldsymbol{r},E,\hat{\Omega}) = \iiint_V S(\boldsymbol{s},E,\hat{\Omega})\frac{e^{-\sum_i \Sigma_{t,i}x_i}}{|\boldsymbol{r}-\boldsymbol{s}|^2}\delta\left(\hat{\Omega},\frac{\boldsymbol{r}-\boldsymbol{s}}{|\boldsymbol{r}-\boldsymbol{s}|^2}\right)d\boldsymbol{s}. \tag{3.47}$$

3.3. SOURCE GENERATION

All photon transport problems considered in the DOCTORS code are ultimately driven by an external source, namely the medical equipment producing the photon beam. These sources can come in a variety of geometric shapes and energy spectra. Currently, DOCTORS has a number of built-in beam types available for analysis including point sources, fan beams, and cone beams. Fan beams and cone beams can be arranged about a

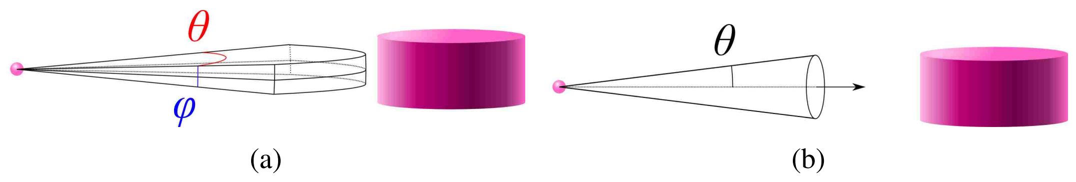


Figure 3.9. The geometry of fan and cone beams in DOCTORS. (a) A fanbeam is defined by θ in the polar coordinate and φ azimuthally. (b) A cone beam is described only be θ.

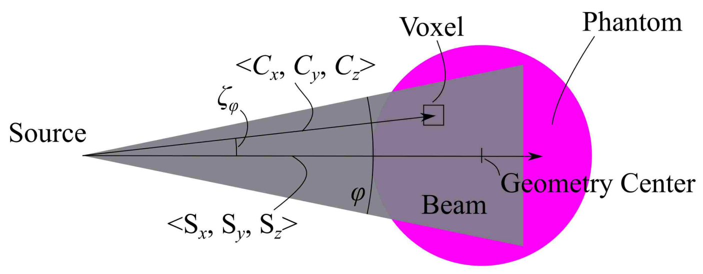


Figure 3.10. Typical DOCTORS phantom geometry viewed axially with the phantom. The beam (gray) is centered on $\boldsymbol{s}$ and the vector from the source to the voxel of interest $\boldsymbol{c}$ is used to determine whether that voxel should be accepted or rejected. In the case of isotropic point sources, $\boldsymbol{s}$ is undefined.

phantom to produce an axial slice. Currently, helical paths are not supported, though they could be added with little additional effort. The geometry of fan beams and cone beams in DOCTORS is shown in Figure 3.9.

3.3.1. Point Source. The simplest source implemented is an isotropic point source. This source is defined only by its position and energy distribution. Given a vector from the source to the center of the geometry, $\boldsymbol{s}$, and a vector from the source to the center of a cell, $\boldsymbol{c}$,the uncollided flux is computing using Eq. 3.43. The collided flux is then computed using the discrete ordinate solution to the LBE given in Eq. 3.41. The vectors are shown in Figure 3.10.

3.3.2. Fan Source. In addition to the values defining a point source (position $\boldsymbol{s}$ and the energy distribution) a fan source is defined by two additional parameters, φ and θ which describe the azimuthal and polar angles subtended by the fan beam respectively. Figure 3.9a shows φ and θ with respect to the geometry setup. The raytracing algorithm used for the fan beam is identical to the point source raytracer except for one difference: before the raytracing is done, whether the ray falls within the fan beam is determined and any ray falling outside the fan beam is not traced, instead its uncollided flux is immediately set to zero. Some voxels are partially enclosed inside the beam which is not accounted for resulting in artifacts at the edges of the beam.

To determine whether a ray is accepted or rejected, the ray direction is first projected onto the xy plane giving $< s_x, s_y, 0 >$ and $< c_x, c_y, 0 >$. Then the cosine of the angle between them, ζ_φ, is computed using

$$\cos(\zeta_\varphi) = \frac{< S_x, S_y, 0 >}{\sqrt{S_x^2 + S_y^2}} \cdot \frac{< C_x, C_y, 0 >}{\sqrt{C_x^2 + C_y^2}} = \frac{S_x C_x + S_y C_y}{\sqrt{S_x^2 + S_y^2}\sqrt{C_x^2 + C_y^2}}. \tag{3.48}$$

Once ζ_φ is computed,

$$|\zeta_\varphi| < \frac{\varphi}{2} \tag{3.49}$$

is evaluated. If the condition is met, the angle is accepted and raytraced normally, otherwise, it is rejected immediately.

Treatment for the θ variable is similar to that of φ except the polar angle is required instead of the azimuthal angle. The polar angle of an arbitrary vector $\boldsymbol{\xi}$ can be computed using

$$\cos(\theta_\xi) = \frac{\xi_z}{\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}}. \tag{3.50}$$

This is used to compute the polar angle for both $\boldsymbol{s}$ and $\boldsymbol{c}$. The difference between the polar angles of the two vectors

$$\zeta_\theta = \theta_s - \theta_c \tag{3.51}$$

is used to evaluate

$$|\zeta_\theta| < \frac{\theta}{2}. \tag{3.52}$$

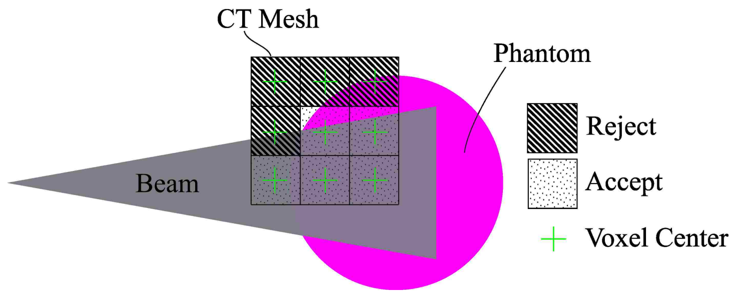


Figure 3.11. The fan beam rejection. Voxels whose center lies outside the beam are rejected. This may lead to artifacts along the edges of the beam.

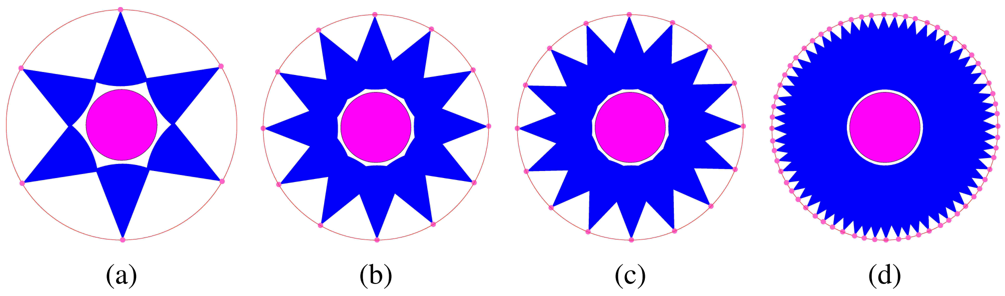


Figure 3.12. Different numbers of fan beam projections (N). (a) $N = 6$. (b) $N = 12$. (c) $N = 16$. (d) $N = 64$.

If the condition is not met, the ray is rejected and no raytrace is performed. Figure 3.11 shows a simple example demonstrating cases when a voxel would be accepted or rejected.

3.3.3. Multi-fan Source. The focal point about which the CT system rotates is assumed to be the center of the xy plane and at the same z-elevation as the first user specified point. Instead of integrating the path of the beam mathematically, a set of samples are taken as illustrated in Figure 3.12. The focal spot defines the center of each fan beam.

Rotation of an arbitrary point (x, y) about the origin θ radians to a new position (x', y') is performed by

$$x' = x\cos\theta - y\sin\theta \tag{3.53}$$

and

$$y' = y\cos\theta + x\sin\theta. \tag{3.54}$$

Rotation about an arbitrary point (namely the focal point), (x_f, y_f) requires translating the system such that the focal point is at the origin, performing the rotation given in Eq. 3.53 and 3.54, and then translating back to the original coordinates. This is done by

$$x' = (x - x_f)\cos\theta - (y - y_f)\sin\theta + x_f \tag{3.55}$$

and

$$y' = (y - y_f)\cos\theta + (x - x_f)\sin\theta + y_f. \tag{3.56}$$

3.3.4. Cone Source and Multi-cone Source. Cone sources are identical to fan beam sources except for the rejection criteria used to determine whether or not a voxel is inside the beam. The rejection criteria is a function of only a single parameter, θ, the maximum permissible angle between the center of the beam, $\boldsymbol{s}$, and a ray toward the point of interest, $\boldsymbol{c}$. The angle between $\boldsymbol{s}$ and $\boldsymbol{c}$ is

$$\cos(\zeta) = \frac{< S_x, S_y, S_z >}{\sqrt{S_x^2 + S_y^2 + S_z^2}} \cdot \frac{< C_x, C_y, C_z >}{\sqrt{C_x^2 + C_y^2 + C_z^2}} \tag{3.57}$$

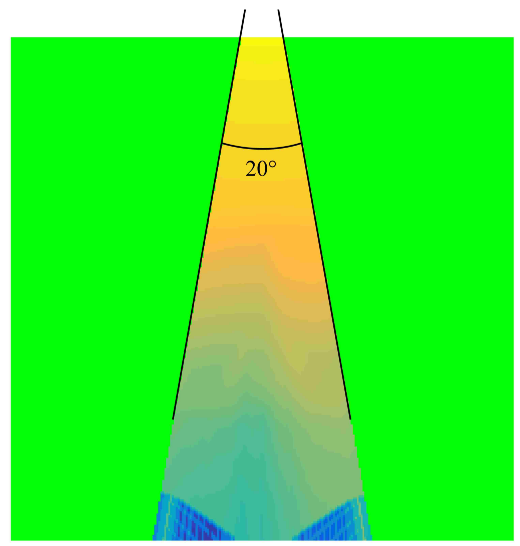
which, if $\boldsymbol{s}$ and $\boldsymbol{c}$ are unit vectors, simplifies to

$$\cos(\zeta) = S_x C_x + S_y C_y + S_z C_z. \tag{3.58}$$

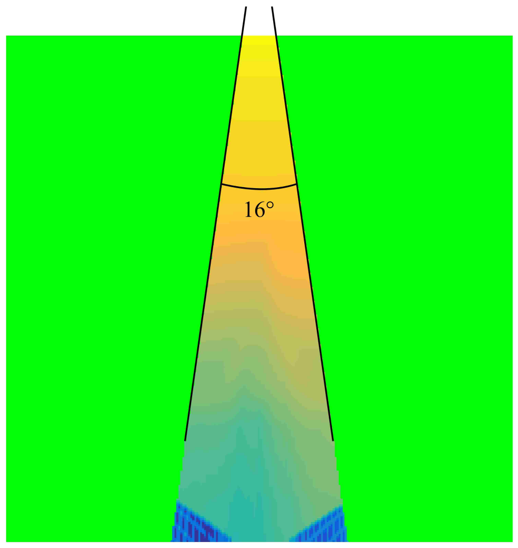
Once the angle is computed, the condition

$$|\zeta| < \theta \tag{3.59}$$

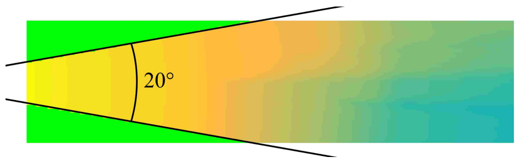
is evaluated to determine whether the ray is accepted or rejected. Figure 3.13 shows a comparison of the cone beam and fan beam shapes. The most noticable difference is in the xz plne where the cone beam is circular, but the fan beam produces a rectangular cross section.



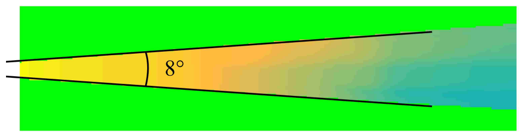
(a) Cone xy



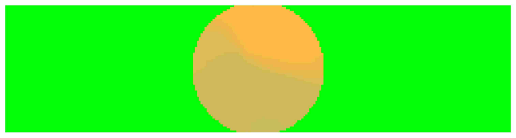
(b) Fan xy



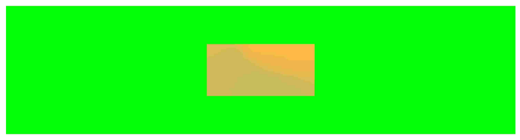
(c) Cone yz



(d) Fan yz



(e) Cone xz



(f) Fan xz

Figure 3.13. Comparison of a cone beam (a), (c), and (e) to a fan beam (b), (d), and (f). The cone beam is 20 degrees in both the xy and yz planes but the fant beam is different. The cross section of the cone beam is cicular (e), but the fan beam cross section is rectangular (f).

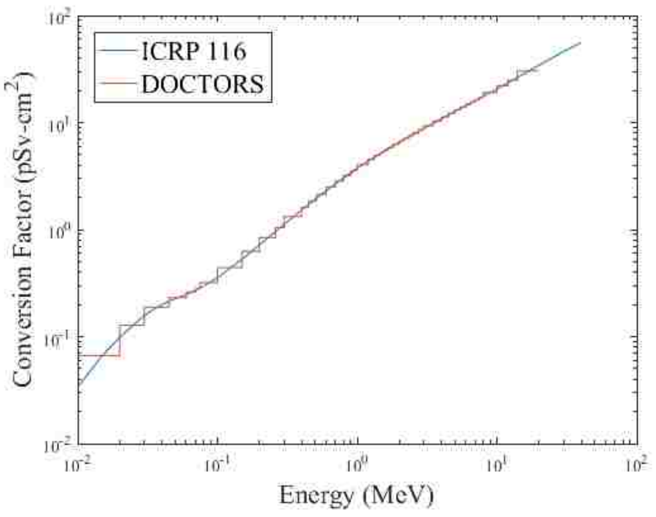


Figure 3.14. Fluence-to-dose conversion factors from ICRP 116 for photons and the group averaged conversion factors for the 47-group data library in DOCTORS.

3.4. DOSE COMPUTATION

Dose can be computed in either one of two ways. The first way is to compute the fluence distribution and multiply by a fluence-to-dose conversion factor from ICRP 116 [ICRP, 2010]. The second way to compute dose is to compute the energy deposition in each cell. In both methods, transport of secondary particles is neglected and all energy deposition is assumed to be local.

3.4.1. Fluence-to-dose Conversion. The publications of the ICRP include fluence-to-dose conversion factors for a reference phantom from various particle types and patient orientations. For CT systems, the rotationally symmetric (ROT) orientation is the most appropriate since the beam rotates about the patient. Figure 3.14 shows the energy dependent fluence-to-dose conversions for the ROT orientation. ICRP 116 distributes the conversion factors as point-wise data which must be converted to group data. This is performed using linear interpolation between the ICRP 116 data points.

This methodology is very simplistic and can be computed quickly with no extra (memory) overhead. When DOCTORS finishes running, the scalar flux is multiplied by the conversion factor and summed over all energy groups:

$$D_i = \sum_{g=0}^{G-1} \varphi_i^g H^g \tag{3.60}$$

where H^g is the conversion factor for the g^{th} group.

3.4.2. Energy Deposition. The alternative way to compute the dose is directly from the local energy deposition. The total deposition, $\mathcal{E}_{T,i}$ for voxel i is

$$\mathcal{E}_{T,i} = \mathcal{E}_{a,i} + \mathcal{E}_{s,i} \tag{3.61}$$

where $\mathcal{E}_{a,i}$ is the energy deposition from absorption and $\mathcal{E}_{s,i}$ is the energy deposition from scatter. The absorption deposition is

$$\mathcal{E}_{a,i} = \sum_{g=0}^{G-1} \Sigma_{a,i}^g \varphi_i^g V_i \bar{E}^g F_1 \tag{3.62}$$

where $\bar{E}$ is the average energy (in MeV) of the g^{th} group and F_1 is a unit conversion factor that converts MeV to Joules. The average group energy is assumed to be the average between the two bounding energies of the group. Equation 3.62 assumes total local energy deposition in the voxel of interest. The energy deposition from scatter is

$$\mathcal{E}_{s,i} = \sum_{g=0}^{G-1} \sum_{h=g+1}^{G-1} \left[\Sigma_{s,i}^{g,h} \varphi_i^g V_i (\bar{E}^g - \bar{E}^h) F_1 \right] \tag{3.63}$$

making the total deposition

$$\mathcal{E}_{T,i} = V_i F_1 \sum_{g=0}^{G-1} \varphi_i^g \left[\Sigma_{a,i}^g \bar{E}^g + \sum_{h=g+1}^{G-1} \left\{ \Sigma_{s,i}^{g,h} (\bar{E}^g - \bar{E}^h) \right\} \right] \tag{3.64}$$

The mass of the voxel, M_i is then computed

$$M_i = \rho_i V_i F_2 \tag{3.65}$$

where ρ_i is the density in g/cm^3 and F_2 is a conversion factor to convert grams to kilograms. The absorbed dose (in Gy) to the voxel is then

$$D_i = \frac{\mathcal{E}_{T,i}}{M_i} = \frac{F_1}{\rho_i F_2} \sum_{g=0}^{G-1} \varphi_i^g \left[\Sigma_{a,i}^g \bar{E}^g + \sum_{h=g+1}^{G-1} \left\{ \Sigma_{s,i}^{g,h} (\bar{E}^g - \bar{E}^h) \right\} \right]. \tag{3.66}$$

4. IMPLEMENTATION

This section covers the details of the implementation of the different components of the DOCTORS code base. The code is available from a GitHub Git repository. At the time of this writing, the repository is located at `https://github.com/eNorris/thesis` and is only available subject to special request and license agreement. An executable version can also be made available upon request. An executable-only public release is planned for the future.

The code is implemented in C++ and requires a C0x11 compliant compiler. The user interface is written using the Qt5 framework. GPU computation is performed via CUDA which must be compiled seperatly by Nvidia's proprietary compiler, `nvcc`.

Unless noted otherwise, all code listings are formatted as C++ code though some parts of the code may be omitted. Omissions include typical "boiler plate" code that accompanies all code such as the `#include` statements and any `using namespace` commands. Additionally, the `main()` function and others are omitted, showing only the code relevant to the discussion.

4.1. VECTOR FLATTENING

Data is stored in large, multi-dimensional arrays. For example, the CT voxel phantom can be intuitively stored as a 3D array of floating precision values which can be easily indexed. However, DOCTORS is written such that it is extensible to GPUs. GPUs are not optimized for data stored in a multi-dimensional fashion, but rather for data stored as a single 1-dimensional array. To emulate the 3-dimensional storage arrangement of the data, indexing arithmetic is used.

On the CPU, data is stored as C++ `std::vector`<T> objects where the `T` template parameter can be either `float` or `double` depending on the precision (32 bits or 64 bits respectively) needed. The storage space available to a `std::vector`<T> object is effectively unlimited, bounded only by the host CPU's physical RAM. However, other effects such as memory fragmentation can limit the practical size of a single, continuous `std::vector`<T> object significantly. To emulate indexing of a $N_x \times N_y$ matrix, a single `std::vector`<T> is created and resized to $N_x \times N_y$ elements. The global index, i, in the flattened array is then

$$i = i_x N_y + i_y. \tag{4.1}$$

This pattern extends to multi-dimensional matrices. For example, a $N_x \times N_y \times N_z$ matrix would be globally indexed as

$$i = i_x N_y N_z + i_y N_z + i_z. \tag{4.2}$$

Figure 4.1 gives an example of the spatial indexing scheme using an $8 \times 8 \times 8$ mesh. The indicated voxel has i_x, i_y, and i_z indices of 1, 5, and 6 respectively. Therefore, the index value of the highlighted voxel is $1 \times 8 \times 8 + 5 \times 8 + 6 = 110$. This flattening pattern continues to higher dimensional phase space to encompas energy and direction.

A variable that is dependent on x, y, z, E, and Ω would be written as $\phi(x, y, z, E, \Omega)$ would be discretized to $\phi(i_x, i_y, i_z, G, i_a)$. Rather than storing ϕ as a 5-dimensional matrix, ϕ is stored as a 1-dimensional vector of the same number of elements.

4.2. CROSS SECTION PARSING

AMPX cross sections are stored in a binary format. These files are *not* distributed with DOCTORS but must be obtained from SCALE6.2 or another code. Figure 4.2 shows the data format of the overall file. The binary file is split into a sequence of records. The

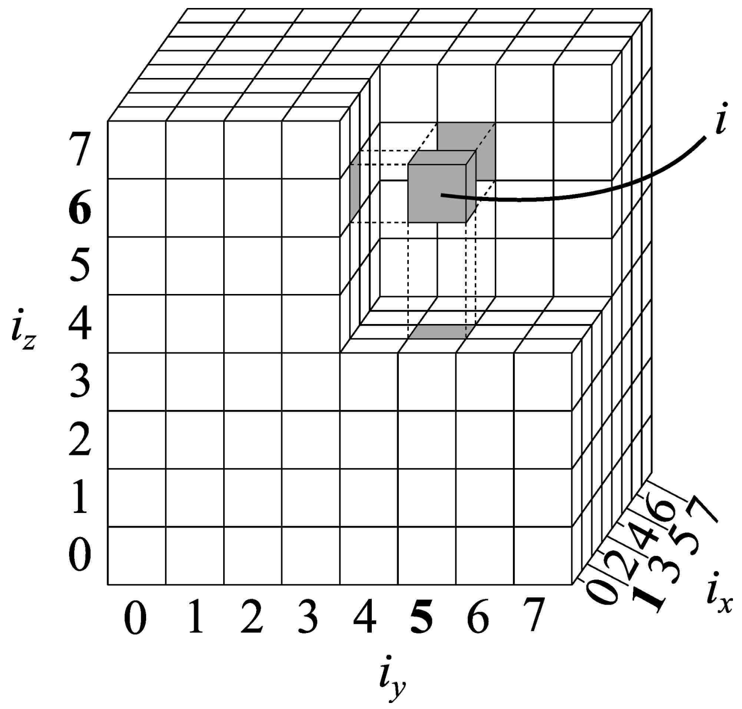


Figure 4.1. Conversion from 3D indexing to 1D "flattened" indexing.

header section of the file has overall information about the file including the number of nuclides stored, their energy group structure, and a comment string describing the file. Following the header section is a list of directory records. Each directory has information about a specific nuclide including the cross section reaction types available, temperatures for which the cross sections were evaluated, the number of Legendre expansion coefficients stored in the scatter matrix, and other data. Following the directories, more records follow that describe the energy structure used in the file. There is one such record for each particle type. The libraries used in the current version of DOCTORS have both neutron and photon cross section data stored.

The file header section, directories, and energy group boundaries describe the structural information necessary to read the following nuclide records. Nuclides are listed in the same order their directories are found after the header setion. Each nuclide contains numerous records.

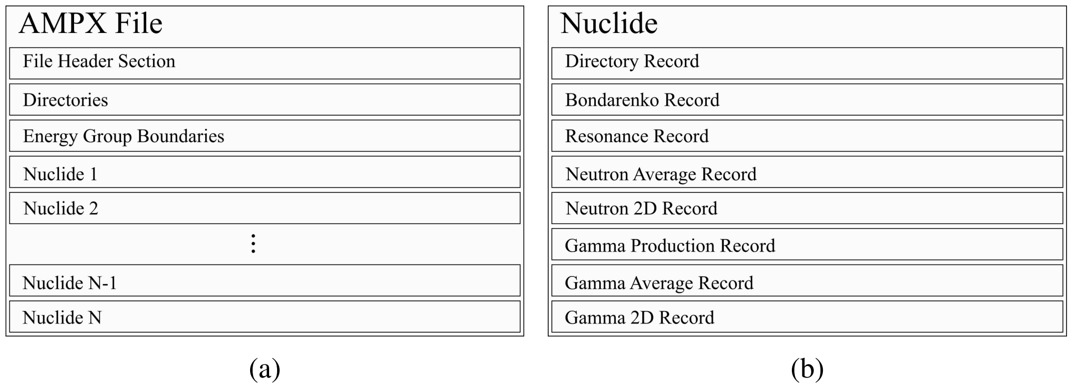


Figure 4.2. The format of the AMPX file. (a) The file contains a header, a list of directory records, energy group information and a list of nuclide entries. (b) Each nuclide entry contains multiple records; the two of concern in this work are the last two containing gamma data.

Each record contains a string of data bytes between a four byte header and four byte footer. The header and footer are identical and, when interpreted as a signed four byte integer, give the size in bytes of the record. An example record is shown in Figure 4.3.

The data is stored in Big Endian format which must be converted to Little Endian for most Intel and AMD processors. The endianess rearrangement is shown in Figure 4.4. Each record is one of 12 unique types; each type of record is formatted and interpreted differently. For example, Type 1 contains general information about the data file such as the number of nuclides, number of energy groups, and a brief description. Type 2 records contains a list of floating-point numbers representing the energy group boundaries. Figure 4.3 shows a Type 1 record. In the case of the header bytes given in Figure 4.3 and 4.4, the bytes correspond to the integer 440 which is the length in bytes of the record (not including the header or footer). Each byte of data is sequentially read, reordered, and converted to an apprpriately typed variable. Table 4.1 shows the converstion for the record given in Figure 4.3.

The header always reports the number of 8-bit bytes required to store the data. However, some data types, such as `char`, are only one byte so each word represents multiple (4 in the case of `char`) distinct characters.

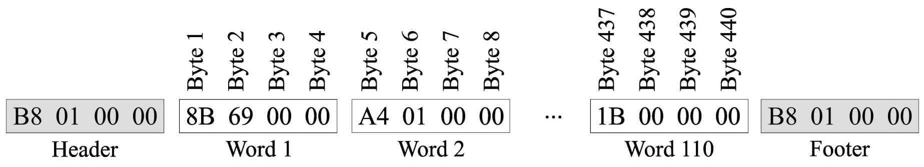


Figure 4.3. An example record parsed. The header and footer are identical and equal to the number of bytes in the record.

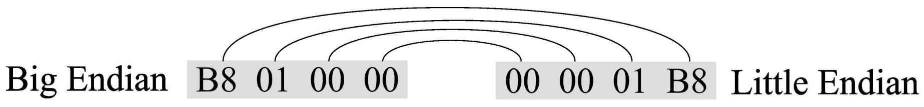


Figure 4.4. The byte reordering from Big Endian to Little Endian. Individual bits within a byte are not rearranged, but the ordering of the four bytes withing a 32-bit word is reversed.

The first entry in each nuclide entry is the directory record. The directory contains general information regarding the nuclide and information necessary to parse the proceeding records. Following the directory, Bondarenko data, resonance parameters, neutron data, and gamma production which are not of interest in photon only problems are read.

The penultimate record contains the average cross section data. This data is averaged over all energies and directions using Eq. 3.10. The only data used in this section in DOCTORS is the total cross section values necessary for implementing the fully discretized form of the LBE given in Eq. 3.15.

The 2D data is stored in a special format optimized for scatter matrix data. The format concists of a sequence of "magic numbers" interpersed within a list of data points. A magic number is a nine digit number IIIJJJKKK. The first three digits, III, are the group number of the highest energy group to scatter to the energy of interest. The next three digits, JJJ, are the group number of the lowest energy to scatter. The final three digits, KKK, are the group number of the sink energy to which particles are scattering. Given a magic number `magic`, the three values can be computed in Listing 1. Note that the group numbers are indexed from 1 to G instead of 0 to $G-1$ as required by DOCTORS. This is corrected by subtracting 1 while doing the indexing arithmetic.

Table 4.1. Byte Reordering

Word	Big Endian	Little Endian	Interpreted	Notes
0	B8 01 00 00	00 00 01 B8	440	Header
1	8B 69 00 00	00 00 69 8B	27019	ID
2	A4 01 00 00	00 00 01 A4	420	Number of nuclides
...	—	—	—	—
11	70 75 6F 43	43 6F 75 70	"Coup"	First four `char` of the title
12	20 64 65 6C	6C 65 64 20	"led "	Second four `char` of the title
...	—	—	—	—
110	20 20 20 20	20 20 20 20	" "	4 spaces ending the title
111	B8 01 00 00	00 00 01 B8	440	Footer

```
magic = READ_NEXT_BINARY_INT()
KKK = magic % 1000
JJJ = (magic % 1000000 - KKK) /1000
III = (magic - JJJ - KKK) / 1000000

src = JJJ
while src >= III
        data = READ_NEXT_BINARY_FLOAT()
        xs[(src - 1)*G + KKK - 1] = data
        src = src - 1
```

Listing 1. Computation of the source and sink groups from the magic number and the subsequent data parsing.

Figure 4.5 shows the microscopic cross section data pulled from the 200-neutron/47-gamma group data file currently used by DOCTORS for hydrogen and oxygen. The data is compared to reference data pulled directly from the ENDF/B-VII.1 photoatomic (MT=501) data library [Cullen et al., 1997] accessible through the Sigma database [NNDC, 2011].

4.3. GENERATION OF MATERIAL CROSS SECTION DATA

Once the data is parsed, weighted combinations of elemental data is used to create material cross sections. Cross section data for photons always uses the naturally ocurring since photo-atomic reactions are not sensitive to the nuclear differences between isotopes.

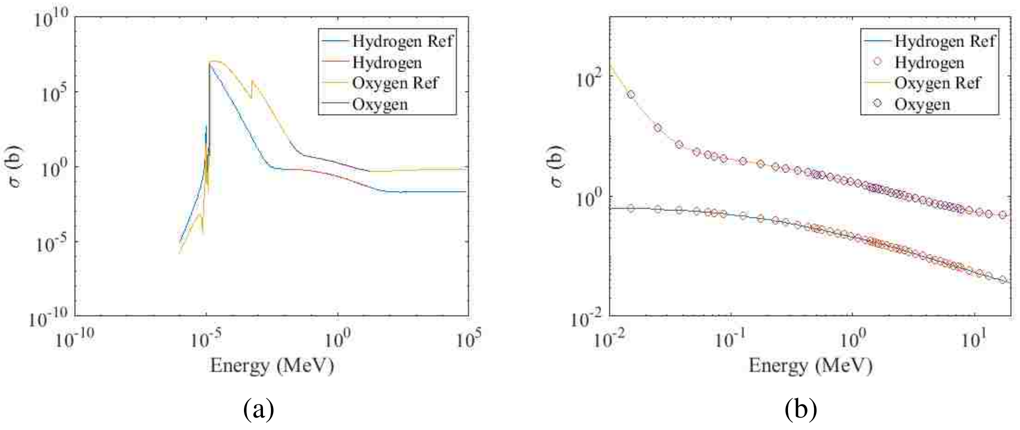


Figure 4.5. The microscopic cross section in bars for hydrogen and oxygen. Both subfigures show identical data, (a) shows the entire data range available in the reference data library and (b) shows only the data range applicable to DOCTORS.

The compositions of N materials are assumed to be given as weight fractions, w_i subject to

$$\sum_{i=1}^{N} w_i = 1 \tag{4.3}$$

and is converted to an atom fraction, a_i:

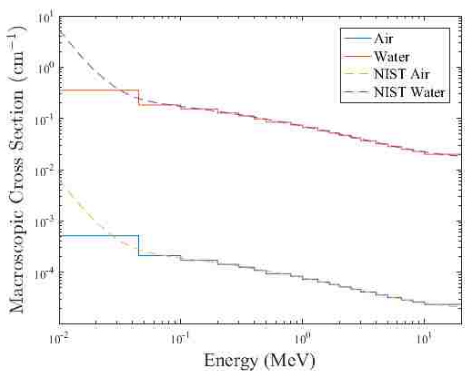
$$a_i = \frac{w_i}{M m_i} \tag{4.4}$$

where

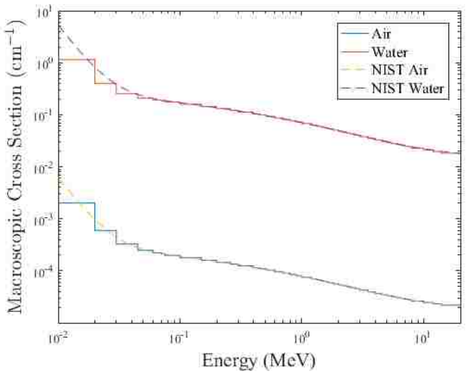
$$M = \sum_{i=1}^{N} \frac{w_i}{m_i} \tag{4.5}$$

and m_i is the element's atomic weight.

In the solution employed by DOCTORS, only the total and scattering cross sections are required. The AMPX data files however, support arbitrary reaction types and have up to a dozen or more reactions for photo-atomic reactions alone for elemental datasets. The



(a) 19 group data



(b) 47 group data.

Figure 4.6. Comparison of the group-averaged DOCTORS cross section data to continuous NIST data.

reaction types are identified by their MT designation. The total, inelastic, and elastic scatter cross sections are MT 501, 502, and 504 respectively. The scatter cross section used by DOCTORS is the sum of the elastic and inelastic cross sections.

The cross section of compound materials are computed as a weighted summation of their components. For example, the cross section of water for a particular reaction is

$$\sigma_{H_2O} = \frac{2\sigma_H + \sigma_O}{3}. \tag{4.6}$$

Two materials were used to validate that the material generation algorithm implemented is correct. Water and air were both generated and compared to emperical data obtained from NIST for their macroscopic cross sections shown in Figure 4.6. The computed group-wise data and the continuous NIST evaluations agree very well.

4.4. QT5 FRAMEWORK

The Qt5 framework was used for implementation of the user interface. Qt5 enables asynchronous calls through its signal/slot mechanism. Signals and slots are special functions that have additional processing performed by the meta-object-compiler (MOC). A signal can be emitted which will execute all connected slots. Signals and slots are connected manually by the user except special ones automatically generated by Qt5.

Qt5 prvides a user interface for building user interfaces. Components such as buttons, drop boxes, radio buttons, etc. are made available to the user. Through extensive use of polymorphism, Qt5 simplifies the addition of graphical interactive elements called widgets. All widgets inherit from the base `QWidget` class which inherits from `QObject`. Any class that utilizes the signal/slot mechanism must extend the `QOjbect` class. As an example, a button can be created in the Qt5 user interface and named `button1`. This object will be accessible as `ui->button1`. When the user clicks on the button, the signal `button1.clicked()` will automatically be emitted. The user can connect this signal to any slot with an identical number arguments of the same type. The `button1.clicked()` signal can be connected to the slot `doStuff1()` but not `doStuff2(int)` since the arguments are not compatible.

Listing 2 gives a C++ snippet that has a long function that will block the user interface. A corresponding sequence diagram is given in Figure 4.7. When the user interacts with the GUI, the writer object begins executing the `doWrite()` function. Until this function completes, the UI thread will be busy and unable to handle additional user interaction or updates. This results in the GUI becoming unresponsive and potentially issuing a warning to the user from the operating system.

To remedy the blocked code, asynchronous calls are made as shown in Figure 4.8. The `doWrite()` function now executes on a seperate thread while code in the GUI continues execution. This prevents the GUI from becoming unresponsive. The code in Listing 2 is updated to use the Qt signal/slot mechanism whose code is given in Listing 3.

```
class MainWindow : public QMainWindow
{
        // Constructor
        MainWindow(QObject *parent);

        // Other parts of the class

protected slots:
        void doSomethingLong();
}

void MainWindow::doSomethingLong()
{
        // Execute a long piece of code
}


MainWindow::MainWindow(QObject *parent)
{
        // Initializations

        // When a button named button1 which was created in the Qt5
        //   creator interface is clicked, the clicked() signal is
        //   automatically emitted which executes the doSomethingLong()
        //   function
        connect(ui->button1, clicked(), this,
                doSomethingLong());
}
```

Listing 2. A long function causes the user interface to block.

4.5. GRAPHICAL USER INTERFACE

The data display plots that show materials and other geometry as well as the flux output use an implementation of the built-in `QGraphicsView` object to render a matrix of `QGraphicsRectItem`s on the screen. The data to be drawn on the screen is populated and some metrics such as the dataset minimum and maximum are computed. The data value at each point is evaluated to determine which bin it belongs to.

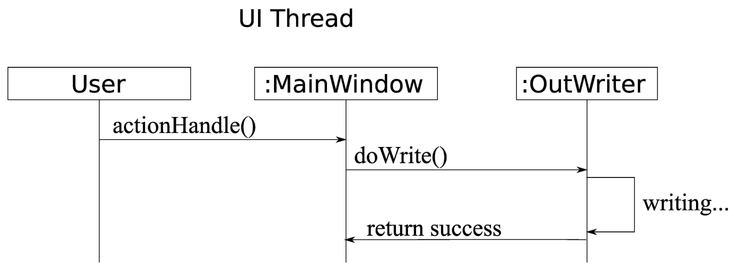


Figure 4.7. Sequence diagram for the synchronous call.

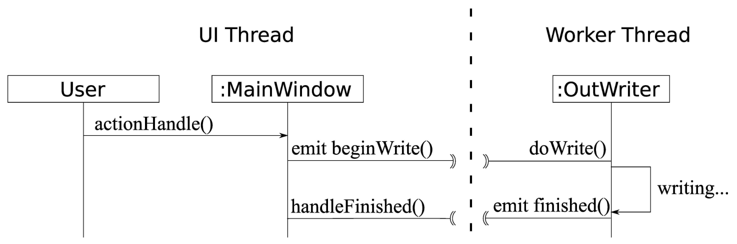
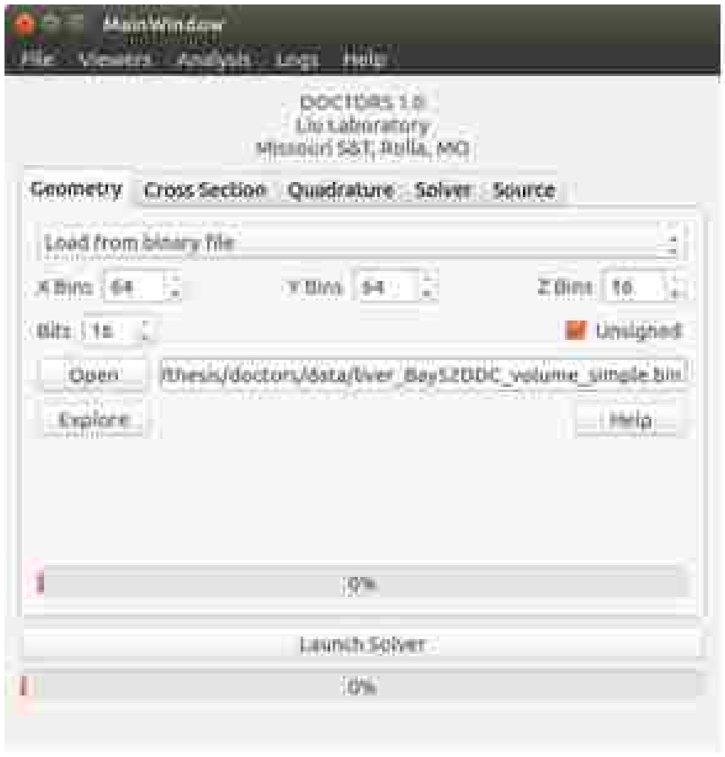


Figure 4.8. Sequence diagram for the asynchronous call.

The first tab is the geometry input, shown in Figure 4.9. In the current version of DOCTORS, the only supported format is unsigned 16-bit binary files whose size is known *a priori*. When the user clicks on the "Open" button, the user is prompted with a dialog for selection of a binary file. The file is read in as a list of $N_x \times N_y \times N_z$ CT numbers. As soon as the data is read in and some checks are performed, the user is prompted for selection of a CT-number-to-material conversion. Currently, two conversions are included in DOCTORS. The first is a water/air only conversion and the second is a more complex conversion to a series of dosimetrically equivalent materials representative of a human patient. Once the user specifies a conversion, the geometry explorer becomes available which allows the user to visualize the geometry file to verify that it was read and interpreted correctly.

Some screenshots of the geometry explorer are shown in Figure 4.10. The geometry viewer can show slices through the voxel phantom along all three major planes. The depth of the slice viewed is changed by moving the scroll bar in the center of the viewer up or down. As the scroll bar moves, the number at its bottom is updated accordingly to indicate



Load from binary file

X Bins 64 Y Bins 64 Z Bins 16

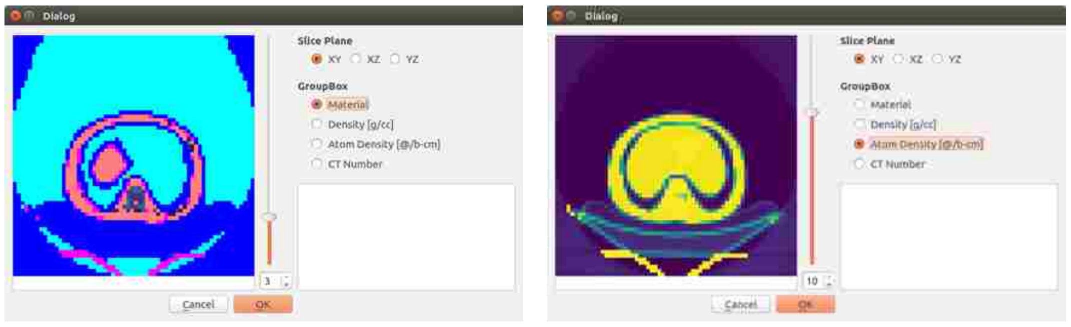
Bits 16 Unsigned

/thesis/doctors/data/liver_BayS2DDC_volume_single.bin

Figure 4.9. The geometry selection tab. The user must first specify the dimensionality of the voxel phantom and then load it from a binary file.

the depth (in voxels) of the slice. In addition to the material, the geometry viewer can render the physical density (in g/cm^3) and the atom density (in atom/b-cm) as well as the raw CT number used to generate the material and density.

The next three tabs identify the cross section dataset to use for the material generation, the quadrature, and the solution anisotropy treatment used. All three tabs are relatively strightforward and contain few widgets. The final tab, shown in Figure 4.11 defines the source. The user is presented with a number of different source geometries to choose from. Each type is described more fully in Section 3.3. Clicking on the "Energy Distribution" button will reveal the popup shown in Figure 4.12. This popup cannot be fully populated until *after* the cross section is loaded in the "Cross Section" tab since the cross section data file defines the energy group structure.



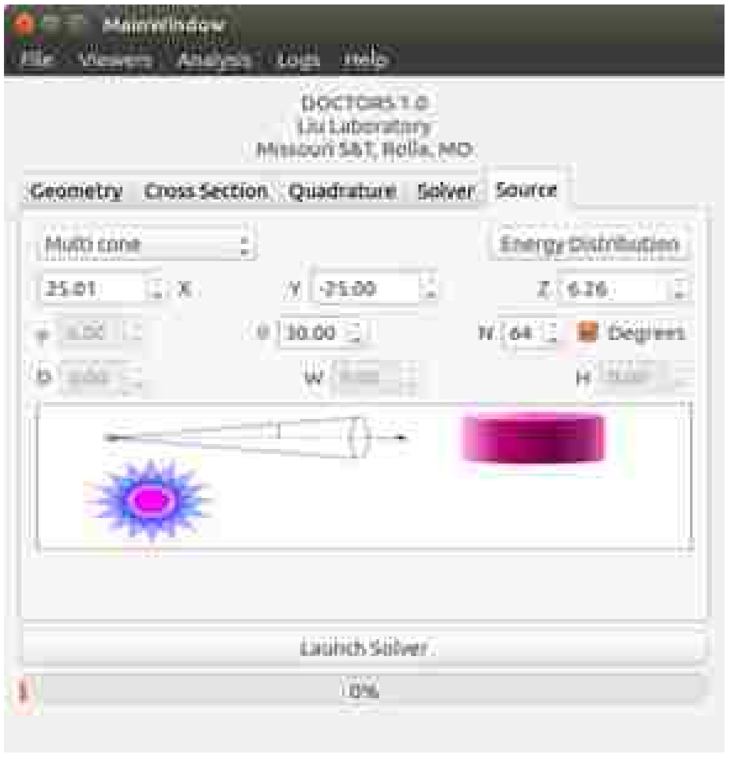
(a) (b)

Figure 4.10. The geometry explorer. (a) The material viewer which shows which material each voxel was converted to. (b) The atom density plot.

The energy distribution dialog, shown in Figure 4.12, can plot the *x* axis on either a linear or logarithmic scale. The user can click and drag to "paint" a spectrum to generate quick, qualitative spectra suiting the user's needs. Alternatively, the user can select from any of the preset spectra in the dropdown menu. Currently, only the uniform beam and diagnostic pulse distributions described later are present.

Once all necessary data is loaded, the "Launch" button becomes active and will remain so until the user creates a conflicting set of input that would prevent the solver from being able to run. When the user clicks on the launch button, the raytracer begins running asynchronously and the output display shown in Figure 4.13 opens automatically. When the raytracer completes, the output dialog is updated with the flux information and the solver begins executing asynchronously. As each iteration completes, the output dialog is progressively updated so that the user can monitor the evolution of the solution.

The output dialog has controls very similar to the geometry explorer. The user can freely choose the plane to slice through the solution and can plot the solution on either a linear or logarithmic scale. The "Distribution" radio buttons allows the user to choose whether the uncollided, collided, or total flux is shown. The "Level Local Scale" checkbox allows the user to control whether the current slice colormap is scaled to the contents of the viewed slice or the contents of the entire voxel phantom. Checking the "Level Local



Multi cone
25.01 X
Y -25.00
Z 6.26
θ 30.00
N 64
Degrees

Figure 4.11. The source tab. The user must select a geometry type and enter all applicable values as well as define the source energy spectrum.

Scale" box will ensure that the viewed data is normalized to include only the viewed data. Unchecking the box will scale the viewed data based on the range of the entire voxel phantom.

In the current version of DOCTORS, dosimetric output is only written to an output file and cannot be visualized graphically. The output format is designed to be human readable and easily read by any major programming language. Listing 4 gives the code to read the file format where the `prod()` algorithm is given in Listing 5. Note that `LIST` may be any appropriately typed list object such as a `std::vector` and a `LIST2D` is a list of lists or vector of vectors.

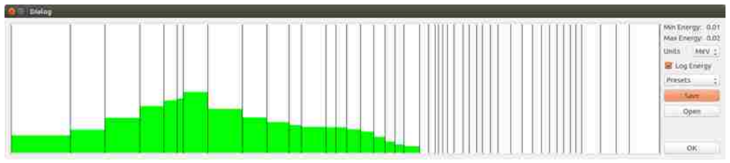


Figure 4.12. The spectrum energy distribution. The spectrum can be displayed on either a linear or logarithmic scale on the *x* axis. The user can either click and drag on the spectrum to "paint" or select a preset.

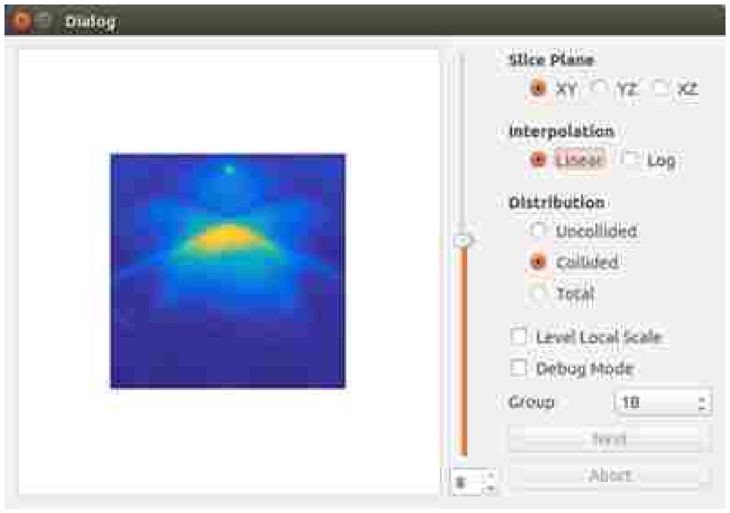


Figure 4.13. The output dialog. The controls are similar to those found in the geometry viewer.

4.6. CUDA

CUDA code is compiled with the Nvidia complier `nvcc`. Qt5 uses the `gcc` compiler and its MOC generator for meta code. In order to connect CUDA code to the Qt MOC, the CUDA code is compiled by `nvcc` to produce an object (`.o`) file. Qt5 then compiles all other files into corresponding object files. The linker then automactically picks up all object files generated by `nvcc`. The final result is an executable that has a Qt5 generated user interface that can communicate with an Nvidia graphics card through the CUDA language.

In order to accelerate the discrete ordinate solution on a GPU, the concurrent tasks at any given moment must be known. In the single-threaded version of the code, the solver sweeps through the voxels one by one in a pre-determined fashion based on the direction. At the very beginning, the input flux to a single voxel is known from its boundary conditions. However, once that voxel's flux is computed, all three of its outgoing flux values enable three voxels to be computed independently of each other. After those three, six can be computed. Each layer of voxels whose flux can be computed independently is called a "subsweep." Figure 4.14 illustrates some subsweeps through a cubic mesh.

In order to parallelize the sweep through the mesh, the global index of each voxel that can be computed in the S subsweep must be known. When those indices are known, each voxel in subsweep S can be solved independently by a CUDA kernel. This task is greatly simplified by noticing that the x, y, and z indices of all voxels in the fourth subsweep, shown in Figure 4.14 all sum to 4 as shown in Table 4.2.

Table 4.2. Subsweep Indices

i	ix	iy	iz	ix+iy+iz
0	4	0	0	4
1	3	1	0	4
2	3	0	1	4
3	2	2	0	4
4	2	1	1	4
5	2	0	2	4
6	1	3	0	4
7	1	2	1	4
8	1	1	2	4
9	1	0	3	4
10	0	4	0	4
11	0	3	1	4
12	0	2	2	4
13	0	1	3	4
14	0	0	4	4

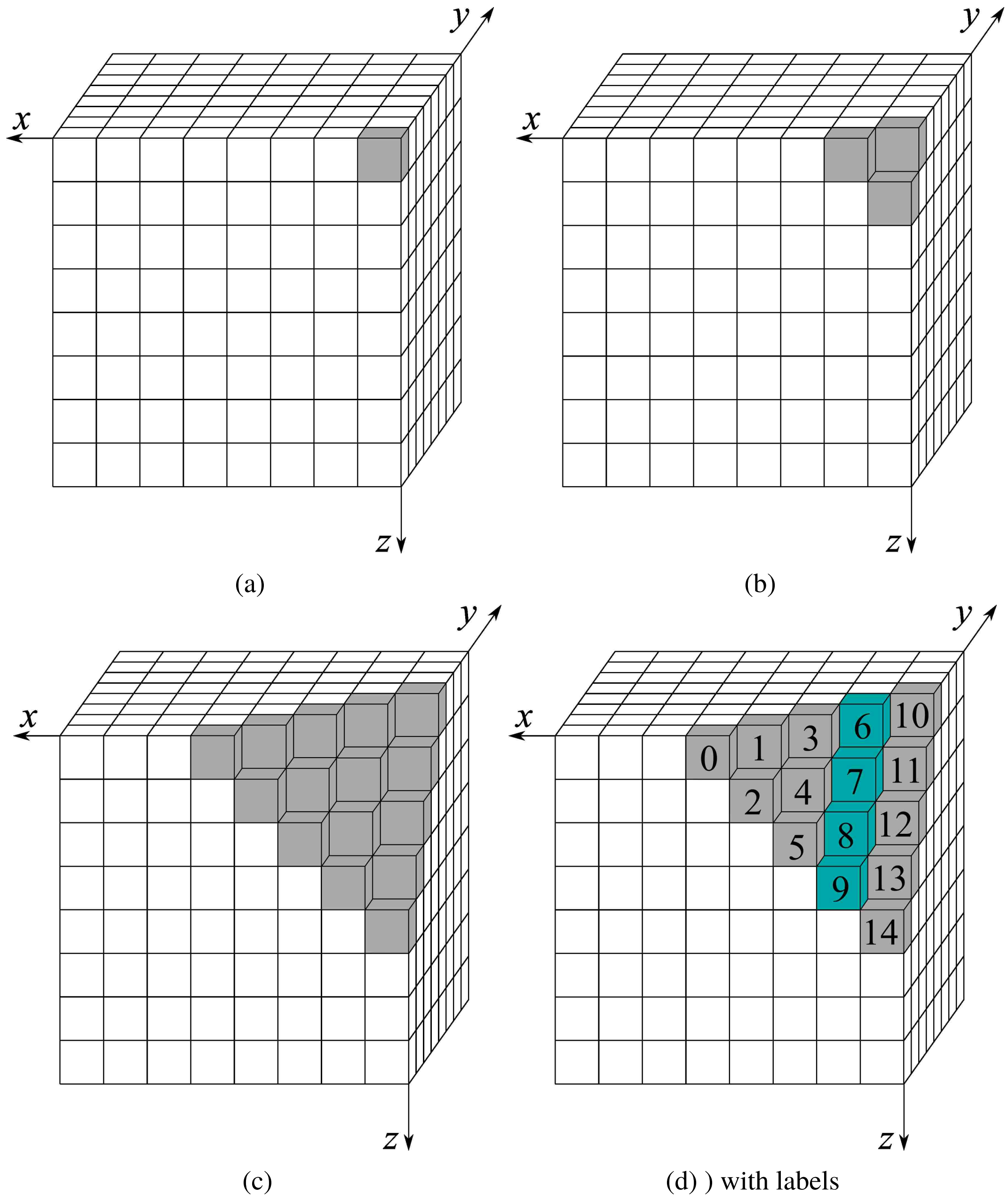


Figure 4.14. The progression of subsweeps throughout a sweep. Each subsweep must complete before those after it. Each voxel within a subsweep can be solved in parallel with all others in its subsweep. (a) Subsweep 0 (S = 0) (b) Subsweep 1 (S = 1) (c) Subsweep 6 (S = 6) (d)Subsweep 6 (S = 6) with labels.

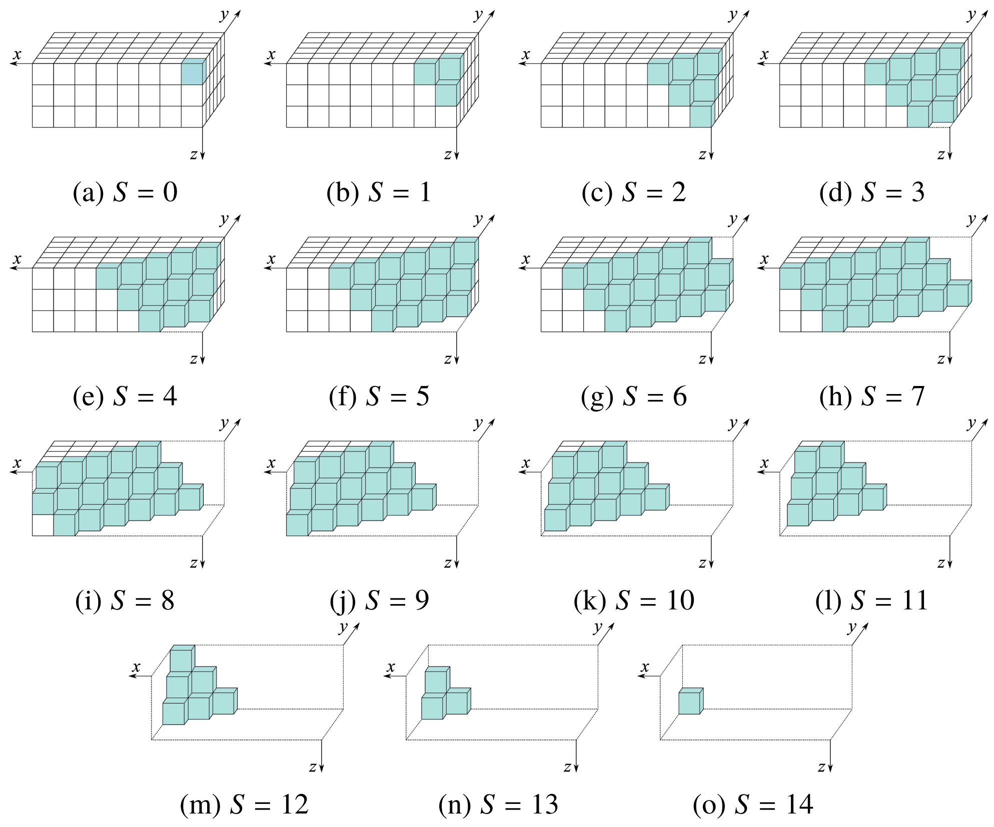


Figure 4.15. The generalized subsweep.

The special case of the cube shown in Figure 4.14 can be extended to a more general case illustrated in Figure 4.15. Notice that the aforementioned intuition that $i_x + i_y + i_z = S$ is not necessarily true. The final subsweep ($S = 14$) contains a single voxel even though many combinations of three integers will sum to 14. The additional constraint is

$$\begin{aligned} i_x &< N_x \\ i_y &< N_y \\ i_z &< N_z. \end{aligned} \tag{4.7}$$

The constraints listed in Eq. 4.7 can be used to compute the total number of parallel tasks in any subsweep. The number of parallel tasks, P, that can be done on subsweep S of an $N_x \times N_y \times N_z$ mesh is given by

$$P = C_S - L_x - L_y - L_z + G_{xy} + G_{yz} + G_{xz} \tag{4.8}$$

where C_S, L_x, L_y, L_z, G_{xy}, G_{yz}, and G_{xz} are defined by Equations 4.9-4.21.

$$C_S = \frac{(S+1)(S+2)}{2} \tag{4.9}$$

$$L_x = \frac{d_x(d_x+1)}{2} \tag{4.10}$$

$$L_y = \frac{d_y(d_y+1)}{2} \tag{4.11}$$

$$L_z = \frac{d_z(d_z+1)}{2} \tag{4.12}$$

$$G_{xy} = \frac{d_{xy}(d_{xy}+1)}{2} \tag{4.13}$$

$$G_{yz} = \frac{d_{yz}(d_{yz}+1)}{2} \tag{4.14}$$

$$G_{xz} = \frac{d_{xz}(d_{xz}+1)}{2} \tag{4.15}$$

$$d_x = \max(S+1-N_x, 0) \tag{4.16}$$

$$d_y = \max(S+1-N_y, 0) \tag{4.17}$$

$$d_z = \max(S+1-N_z, 0) \tag{4.18}$$

$$d_{xy} = \max(S+1-N_x-Ny, 0) \tag{4.19}$$

$$d_{yz} = \max(S+1-N_y-Nz, 0) \tag{4.20}$$

$$d_{xz} = \max(S+1-N_x-Nz, 0) \tag{4.21}$$

Each voxel in a subsweep can be computed in parallel. Mathematically, the i^{th} subsweep from all directions can be computed in parallel. However, in practice, this results in a race condition on the GPU hardware. As such, parallelization in angular space is not implemented in the current version of DOCTORS.

4.7. MCNP6 GENERATION

DOCTORS has the capability to generate MCNP6 input files from the CT mesh data and source specification provided by the user. The input file is procedurally generated using the information provided to DOCTORS. If insufficient information is provided, an error message is generated and no output file is made. Note that the output is always named "mcnp_out.inp" so it will *override* existing files. To save a file, it must be manually renamed to avoid accidental deletion!

The geometry is the largest section of the MCNP6 input file. The dimensions of the phantom mesh are used to generate planar surfaces at the appropriate locations. Those surfaces are then arranged into cells defining each voxel. The density information is pulled directly from the internally generated data built during the CT-number to material conversion.

The source is the most difficult component to generate. Currently, all source types except fan beams can be generated automatically. The difficulty in generating fan beams comes from an intrinsic limitation of MCNP6. MCNP6 can only model cone beams through a biasing mechanism; it cannot model other kinds of beams. In practice, this is rarely a major limitation since physical collimators can be added to a input model to generate the desired beam shape. However, since DOCTORS already has a geometry mesh, addition of collimators is very difficult since the collimator becomes "smeared" across voxels.

4.8. HARDWARE

For this work, a computer with an Intel i7-5960X 8 core (16 hyperthreads) processor with a base clock speed of 3.5 GHz and an Nvidia Titan Z graphics card was used. Currently, if the problem requires more memory than is available on the GPU, the problem can still be solved, but much more time will be required to to copy overhead between the CPU and GPU. If the problem requires more memory than either the GPU or CPU can provide, an error is thrown and the simulation is not run.

Only Nvidia GPUs are recognized by DOCTORS since it relies on the CUDA interface which is a Nvidia proprietary product. There are no limitations on the CPU except that it be Little-Endian based (nearly all major CPUs meet this requirment). Big-Endian hardware would fail to parse the data files correctly.

```
class MainWindow : public QWindow
{
        // Member variables
        QThread workerThread;
        Worker *worker;

        // Constructor
        MainWindow(QObject *parent);

protected slots:
        handleResult();
}

MainWindow::MainWindow(QObject *parent)
{
        // Initializations
        worker = new Worker;

        worker.moveToThread(&workerThread);

        // Set up the thread connections
        connect(&workerThread, finished(), worker, deleteLater());
        connect(this, begin(), worker, doSomethingLong());
        connect(worker, done(), this, handleResult());
}

class Worker : public QObject
{
private signals:
        void done();

public slots:
        void soSomethingLong();
}

Worker::doSomethingLong()
{
        // Do stuff...
        emit done();
}
```

Listing 3. Signals and slots enable a long function to be called without blocking the user interface.

```
// Declare variables
int N;
LIST S;
LIST2D dims;
LIST data;

// Read the number of dimensions
READ_INT(N);

// Read the size of each dimension
for(int i = 0; i < N; i++)
        READ_INT(S[i]);

// Read the axis labels
for(int i = 0; i < N; i++)
        for(int j = 0; j < S[i]; j++)
                READ_FLOAT(dims[i][j]);

// Read the data
int dataElem = prod(S);
for(i = 0; i < dataElem; i++)
        READ_FLOAT(data[i]);
```

Listing 4. Algorithm to read the output format used by DOCTORS.

```
int prod(LIST p)
{
        int product = 1;
        for(int i = 0; i < p.SIZE(); i++)
                product = product * p[i];
        return product;
}
```

Listing 5. The `prod()` algorithm

5. RESULTS

This section summarizes the computational results obtained from DOCTORS and the validation to MCNP6. The MCNP6 reference compared to is automatically generated by DOCTORS to guarantee that the geometry and source parameters are identical. This process is described in detail in Section 4.7.

5.1. PREPROCESSING

Two computational phantoms were used for verification of the DOCTORS code. The first phantom is a $256 \times 256 \times 64$ voxel mesh of a 35 cm water phantom. The water phantom is in an acrylic casing and the entire mesh measures $50 \times 50 \times 12.5$ cm. Each voxel is then a cube with sides of 1.95 mm. The mesh has a total of 4194304 voxels. The second phantom is of the same dimensions and resolution, but of a realistic human phantom. The mesh emulates a CT scan of a patient's midsection where the liver would be located.

Hounsfield units range from -1000 for air to 0 for water and higher for more radiopaque materials. However, data is stored as an 16-bit `unsigned int` in the CT voxel phantoms. Therefore, the data has 1024 (2^{10}) added to it to avoid rolling negative values over. The first step is to reinterpret the `unsigned int` values as `int` values and subtract 1024.

5.1.1. Artifact Removal. At the periphery of the computational mesh, artifacts can appear as a byproduct of the reconstruction algorithm used. Figure 5.1 shows the artifacts in question in a xy slice at the lowest ($z = 0$) level in the phantom. Two hundred sixty one (261) artifacts were detected in the phantom. To remove these artifacts, the CT number of any voxels whose CT number was greater than or equal to 65500 was set to 0. Figure 5.2 shows the same cross sectional plot after the artifacts are removed.

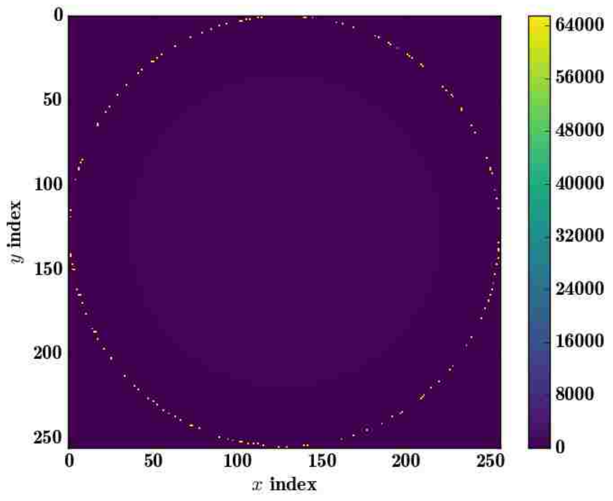


Figure 5.1. The unmodified phantom. An xy slice at z index = 0. Scale is the raw CT number before correction to CT number (by subtracting 1024).

The water phantom in Figure 5.2 can be broken into four regions: the water phantom, the container, the air, and the corner artifacts. The water phantom is the centermost region of the slice.

5.1.2. Geometry Simplification. The original phantom data for both the water phantom and the liver phantom are $256 \times 256 \times 64$ meshes which result in 4,194,304 total voxels. Though MCNP6 is capable of running such a mesh, the overhead of loading the mesh into memory and initiating the Monte Carlo solver can take on the order of hours. Therefore, the benchmarks were run using a simplified geometry of $64 \times 64 \times 16$ which has only 65,536 voxels. The smaller files can run to completion of 10^9 particles within two hours. The water phantom original and simplified geometries are shown in Figures 5.3 and 5.4 respectively. A more detailed comparison of the xy slices is shown in Figure 5.5.

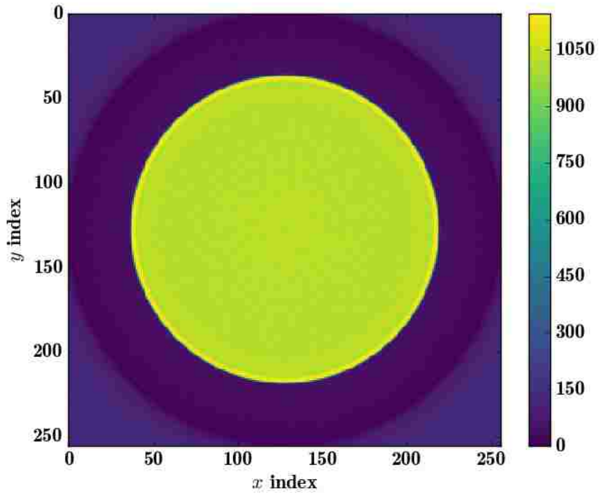


Figure 5.2. The phantom after artifacts are removed. An xy slice at z index = 0. Scale is the raw CT number before correction to CT number (by subtracting 1024).

5.2. CONE-BEAM WATER BENCHMARK

The first benchmark, is a 8 degree cone targeting a water phantom. The water phantom is a 35 cm diameter water cylinder that stands 12.5 inches tall. The water is encased in a 0.5 cm thick plastic container. Each voxel in the mesh is assigned one of the materials given in Table 5.1. Note that the CT range column is of the raw data before 1024

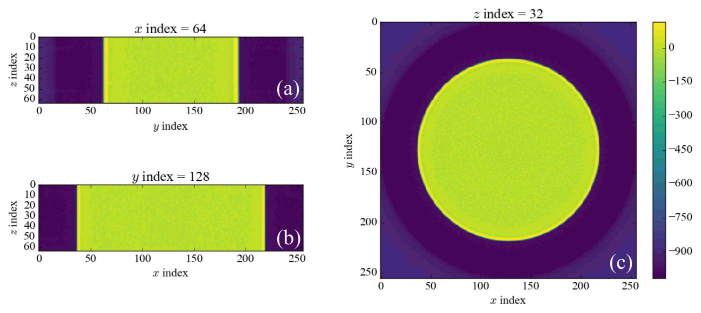


Figure 5.3. CT number data for the 256 × 256 × 64 water phantom.

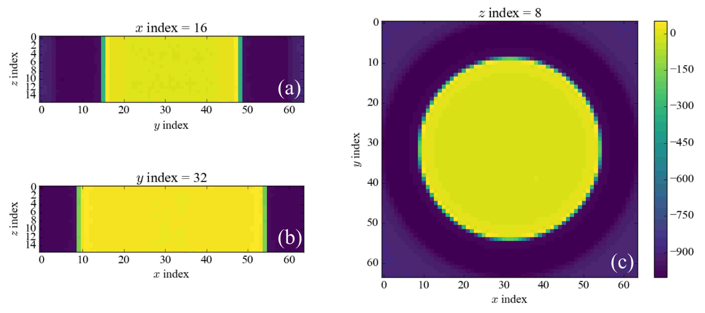
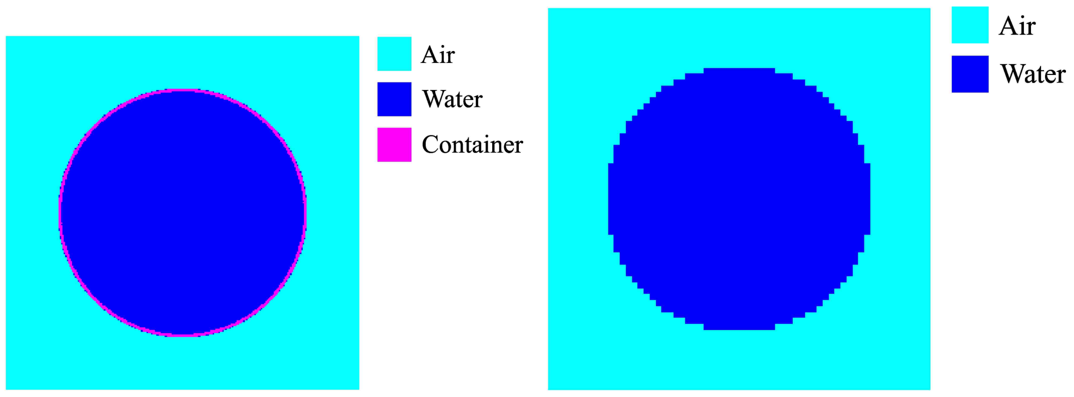


Figure 5.4. CT number data for the 64 × 64 × 16 water phantom.



(a) (b)

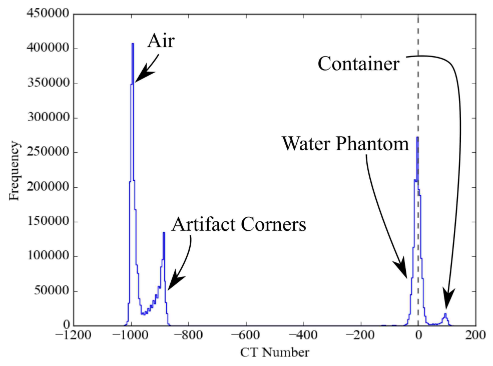
Figure 5.5. The phantom materials derived from CT numbers before (a) and after (b) geometry simplification.

is subtracted. Once the artifacts are removed, the histogram is plotted in Figure 5.6. The logarithmic scaled histogram shows the low frequency voxels between -800 and -100 the are difficult to identify as water or air. These voxels are around the periphery of the container.

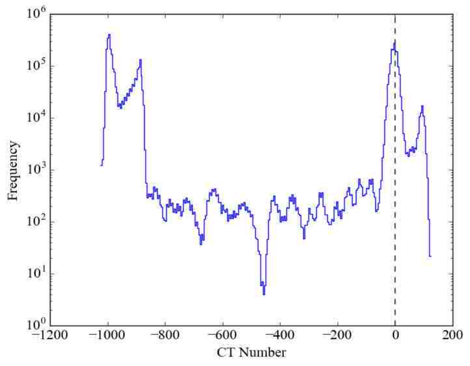
One of the materials listed in Table 5.1 is the "Artifact" material. These voxels apppear in the corners of the reconstruction and are artifacts caused by the reconstruction algorithm. The corners are not sampled by each projection as the CT system rotates about the phantom. During reconstruction, this results in those regions being asigned proportionally higher CT numbers.

Table 5.1. Water Regions

Region	CT Range	Voxels	Fraction (%)
Water	$0 \geq x > -67$	1581199	37.70
Container	$-67 \geq x > -600$	81502	1.94
Air	$-600 \geq x > -1080$	1650372	39.35
Artifact	$-1080 \geq x > -65535$	881231	21.01



(a)



(b)

Figure 5.6. The histogram of the CT numbers in the water phantom on a (a) linear scale and (b) logarithmic scale.

Due to the meshing used in CT phantoms, the spatial domain of the problem is identical for both the deterministic and Monte Carlo problems. A key difference between though is the group structure used in the group-averaged deterministic cross sections. Figure 5.7 shows the continuous energy macroscopic cross section for both air and water versus the discretized data used by DOCTORS. Unfortunately, the datasets used in DOCTORS originate from SCALE6.2, a light water reactor analysis code. Therefore, their upper photon energy bound is 20 MeV which is far higher than necessary for diagnostic imaging. This results in the region of interesting being discretized very roughly. In the 19-group discretization, the entire diagnostic domain is covered by only three energy groups.

5.2.1. Uniform Beam. This benchmark uses a "uniform" beam in which source particles are generated equiprobably in all groups. Therefore, the energy distribution $S_0(E)$, is not uniform, but rather $S_0(E)/\Delta E$ is. The goal of this benchmark is to determine

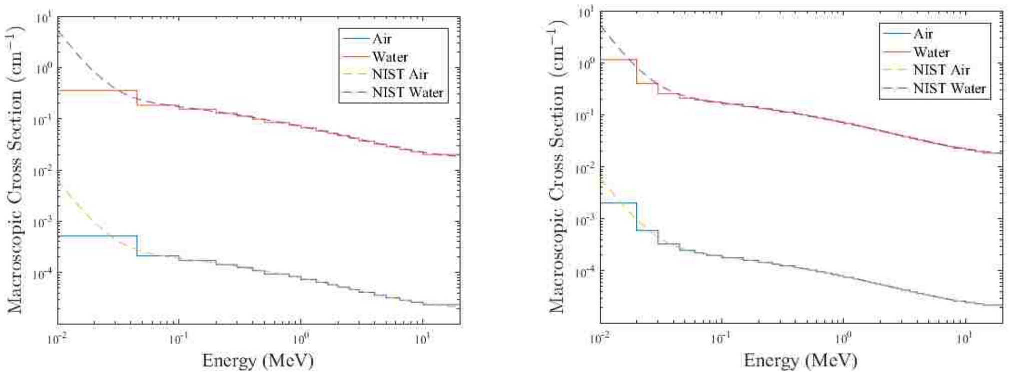
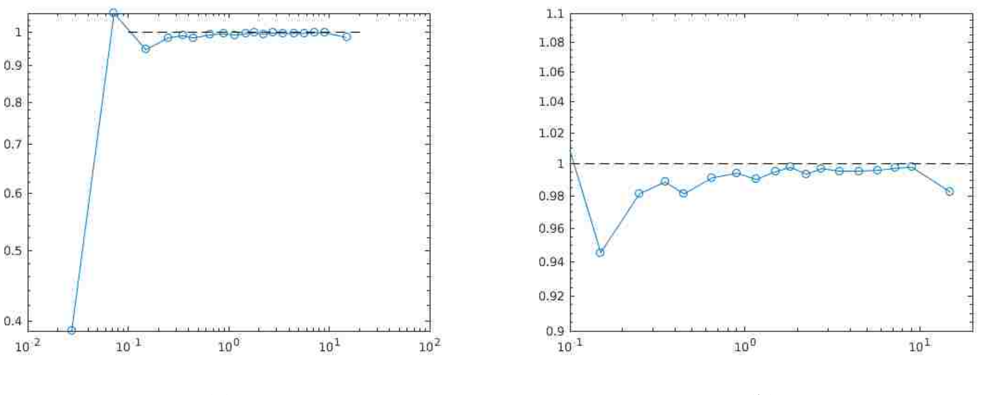


Figure 5.7. Comparison of the group cross section data used by DOCTORS to the continuous cross section data used by MCNP6 for (a) 19 groups and (b) 47 groups. The continuous data is provided by NIST [Hubbell and Seltzer, 1996].

the impact the energy discretization has on the results. The 19-group data set is very coarse in the diagnostic energy range; this benchmark is designed to determine whether or not this data set can yield meaningful results.

Figure 5.8 shows the results of the comparison of the group averaged uncollided flux values between MCNP6 and DOCTORS for the 19-group approximation. The plot shows the uncollided flux averaged over the center 8 voxels in the phantom. Only the uncollided flux is used in this benchmark because it is not dependent on the additional solver parameters such as quadrature and anisotropy treatment but instead has an exact solution. The agreement is within about 2% for energy groups above 300 keV but falls off rapidly in the diagnostic energy range.

To explore the reason for the discrepancy in the diagnostic range, the finely meshed uncollided flux in energy space is examined in Figure 5.9. The uncollided flux at lower energy drops off rapicly at the low energy side of each group in the diagnostic regime. This is caused by the significant downscatter that is not accurately modeled with such wide energy groups. To address this, the 47-group data file was also run. Similar results are plotted in Figure 5.10.



(a) (b)

Figure 5.8. Ratio of the MCNP6 uncollided flux to the DOCTORS uncollided flux for each group. (a) The entire range of data. (b) The same data scaled to show the range of interest.

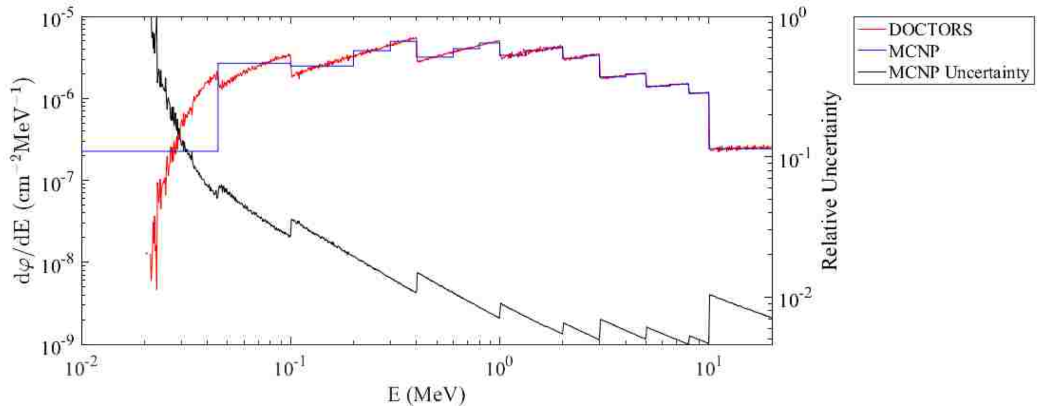


Figure 5.9. The 19 group center flux comparison for a uniform beam.

As an additional verification that DOCTORS is computing correct results, analytical data points were added. The analytical points are computed

$$\left(\frac{d\varphi}{dE}\right)^g = \frac{e^{-(\mu_a^g x_a + \mu_w^g x_w)}}{4\pi G(x_a + x_w)^2(\Delta E)^g} \tag{5.1}$$

where μ_a^g and μ_w^g are the attenuation coefficients for air and water repectively for the g group, x_a and x_w are the pathlength through air and water respectively, and ΔE is the width of the g group in MeV. Dividing by the number of groups maintains consistency with the computational codes which uniformly distribute particles across all groups.

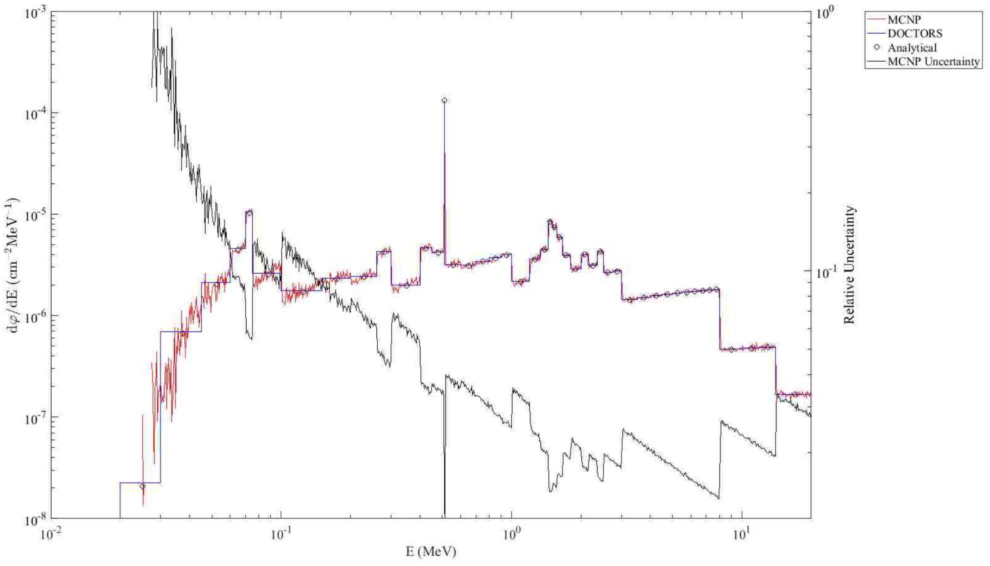
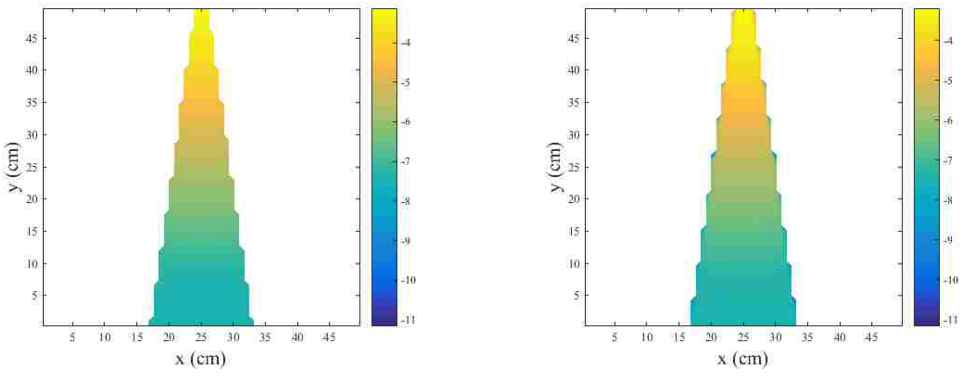


Figure 5.10. The 47 group center flux comparison for a uniform beam.

The results shown in Figure 5.10 agree with the MCNP6 uncollided flux distribution much more accurately than for the 19-group approximation. Therefore, all results after this use the 47-group approximation exclusively. Ideally, a more refined dataset with *only* photon data in the diagnostic range would be used, but currently, no such data set is available in DOCTORS.

5.2.2. Diagnostic Pulse. Once the 19 and 47 group datasets were compared using the uniform energy distribution, a second benchmark was created. This benchmark uses a 1-group pulse of photons in group 41 (70-75 keV). Using a single group pulse is not representative of a diagnostic x-ray beam, but it makes interpretation of the results simpler. Since all uncollided flux is in a single group, the downscatter and within-group inscatter can be differentiated explicitly which would not be otherwise possible. This facilitates debugging of code and validation against MCNP6.



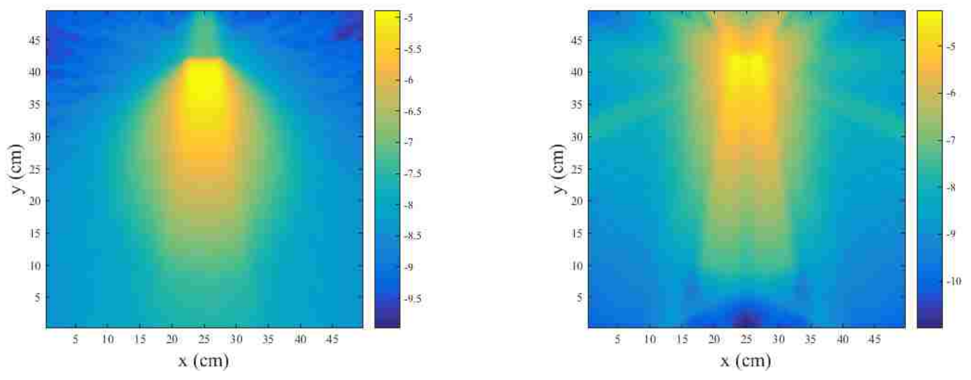
(a) (b)

Figure 5.11. Comparison of the uncollided flux in (a) DOCTORS and (b) MCNP6.

Figure 5.11 shows the uncollided flux distribution for both MCNP6 and DOCTORS. Qualitatively, the distributions appear to agree very well except for the artifacts around the edges of the beam in the MCNP6 results.

Figure 5.12 shows a comparison of the MCNP6 collided flux and the DOCTORS collided flux. The DOCTORS flux exhibits noticable ray effect artifacts but good qualitative agreement otherwise with respect to the shape of the flux distribution. A vertical lineout of Figure 5.12 is shown in Figure 5.13. The lack of anisotropy results in more particles streaming out to the side of the beam and backscattering that expected. This causes a buildup when the beam enters the phantom and a rapid decrease as more particles scatter than predicted by MCNP6. Figure 5.14 shows lineouts for the in-group scatter with the uncollided flux (group 41) and the purely downscattered flux in group 42. Figure 5.15 also shows lineouts for the in-group scatter with the uncollided flux (group 41) and the purely downscattered flux in group 42 but with respect to different quadratures instead of anisotropy treaments.

In MCNP6, the effective dose was computed using a FMESH tally with associated DE and DF cards to weight the flux. The most accurate tally to compute the absorbed dose with is a F6 tally, however, those cannot be spatially distributed via a FMESH tally.



(a) (b)

Figure 5.12. The in-group collided flux in group 41 (70-75 keV)

Therefore, a F4 tally was used with a multiplier designed to emulate a F6 tally as closely as possible. The comparison is shown in Figure 5.16. The F6 and F4 tallies agreed within 2% at the center of the phantom, which validates that the spatially distributed FMESH with a FM multiplier is a good approximation of the energy depositied in a voxel.

The effetive dose and equivalent doese in DOCTORS are computed using Eq. 3.60 and 3.66 respectively. The effective dose and absorbed doses are given in Figure 5.17.

5.3. 16 CONE-BEAM WATER BENCHMARK

This benchmark is designed to test the multi-cone beam projection. Sixteen cone beams are simultaneously used to generate the data. Each cone beam fires the 1-group diagnostic pulse energy spectra described in Section 5.2. Overall, the raytracer shows excellent agreement but the collided flux has more error than expected.

The uncollided flux, which should be nearly identical between MCNP6 and DOCTORS is shown in Figure 5.18. The relative error between the two and a vertical lineout through the center are shown in Figure 5.19.

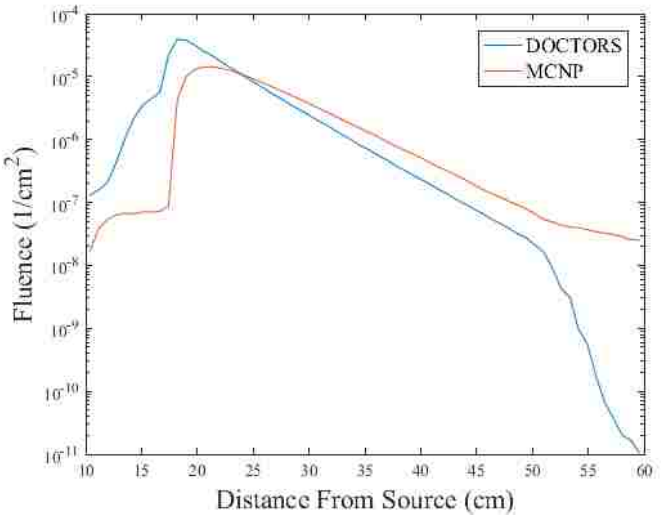
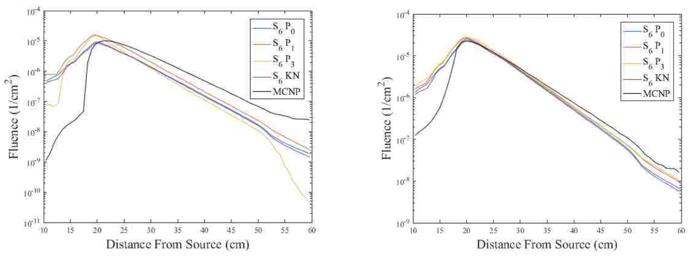


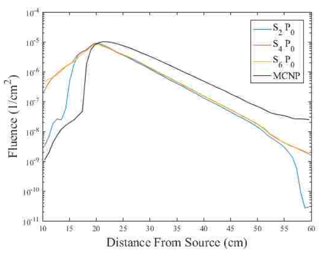
Figure 5.13. Comparison of the in-group scatter in group 41 using an isotropic assumption.



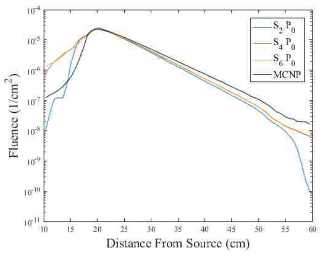
(a) (b)

Figure 5.14. Lineout showing the effect of anisotropy. (a) Group 41 (70-75 keV). (b) Group 42 (60-70 keV).

The collided flux is shown in Figure 5.20. The MCNP6 collided flux is much more uniform across the patient. The uniformity is seen in both the side-by-side comparison of the flux distributions shown in Figure 5.20 and the vertical lineout shown in Figure 5.21. DOCTORS overestimates the flux at the periphery and underestimates the flux at the center of the patient. The trend is effectively an overestimation of the attenuation through the patient by DOCTORS.



(a)



(b)

Figure 5.15. Lineout showing the effect of quadrature. (a) Group 41 (70-75 keV). (b) Group 42 (60-70 keV).

Knowing that the flux profiles vary, the dose profiles can also be compared; a similar discrepancy appears in the dose profile generated from both dose computation methodologies used shown in Figure 5.22. Qualitatively, the dose profiles in both MCNP6 and DOCTORS are very similar, though the DOCTORS dose is overestimated by roughly a factor of two. This is believed to indicate that the scatter distribution at lower energies is being overestimated. This overestimation is in turn caused by the angular treatment of the transport.

The angular transport is affected by two parameteres: the quadrature and the anisotropy treatment. Since the anisotropy treatment was explored in the previous benchmark and no significant changes were found be increasing the P_N expansion or changing the anisotropy methodology altogether, the problem likely lies with the quadrature. Higher quadratures may result in significantly improved flux distributions since lower quadratures will cause particles to preferentially leave the beam if insufficient angular directions are within the beam. This then motivates the final benchmark problem.

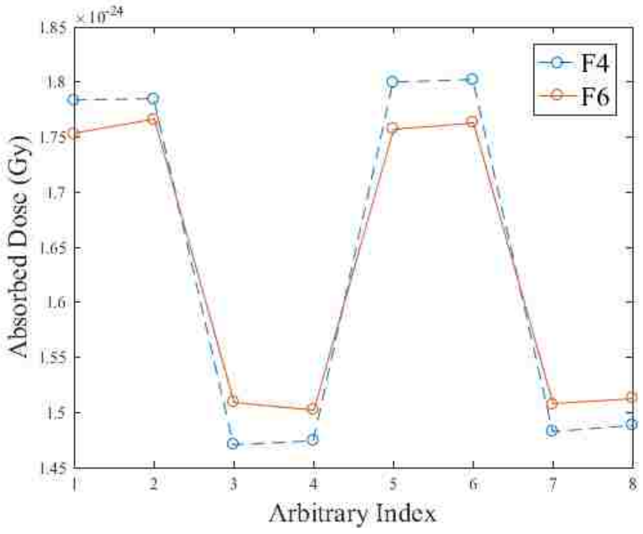
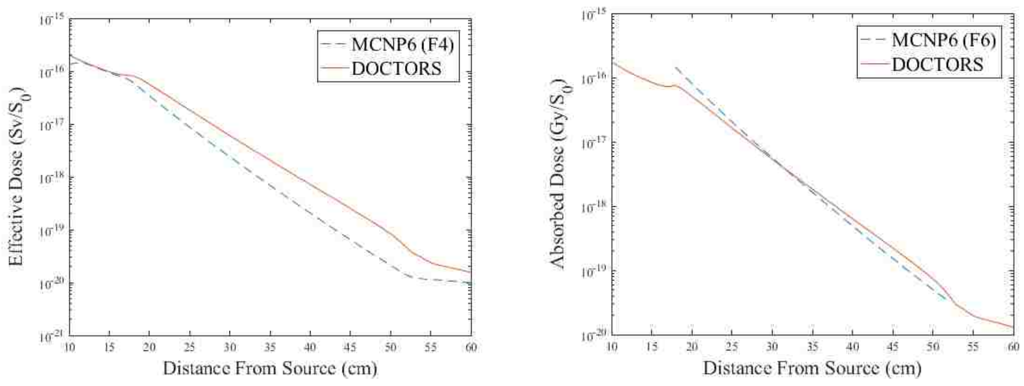


Figure 5.16. Comparison of the absorbed dose computed with F4 and F6 tallies in MCNP6. All eight data points were located at the center of the phantom and the two computation methods were within 2% of each other.

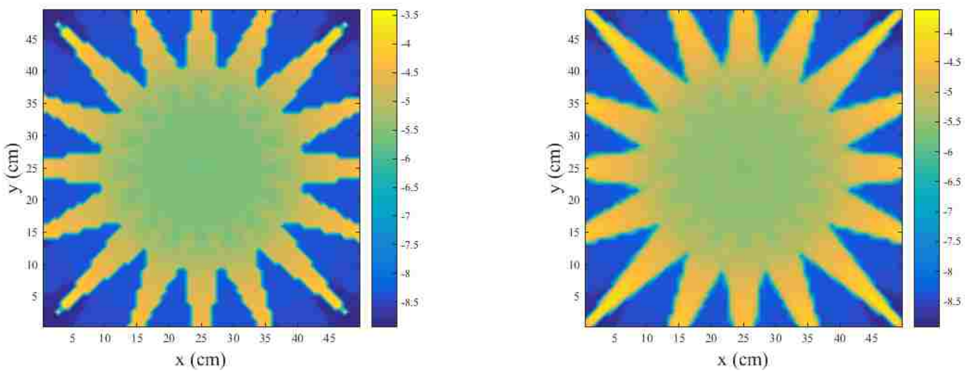
5.4. 64 CONE-BEAM PHANTOM BENCHMARK

The final benchmark problem uses 64 cone beams surrounding a full patient phantom. Since the triangular artifacts were still seen in the 16-beam water phantom, the number of beams used in this benchmark was increased to 64 cone beam projections. Also, each beam was moved further from the isocenter of the phantom. The total diameter between the detector and x-ray tube is 1 meter, thus the coordinate of the first projection is at y = -25 cm. The cone beam was broadened to 30 degrees to more closely match the CT data. The energy distribution was not changed. This benchmark problem is much more realistic and provides a more distributed uncollided source which is expected to help mitigate some of the problems identified in the water benchmarks.



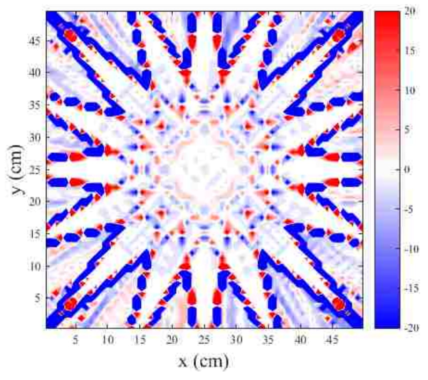
(a) (b)

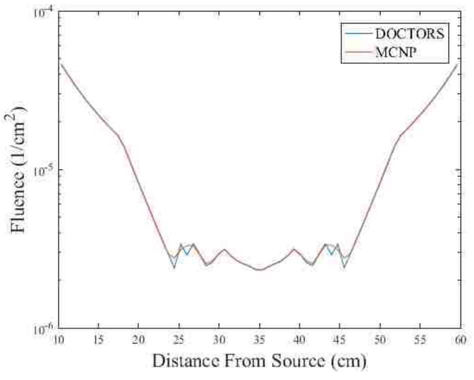
Figure 5.17. The computed dose in the phantom. (a) The effective dose computed using ICRP 116 fluence-to-dose conversion factors. (b) The absorbed dose using energy deposition.



(a) (b)

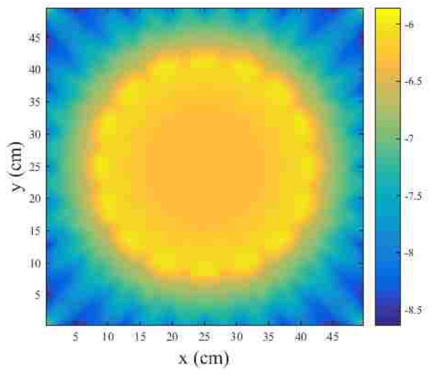
Figure 5.18. Comparison of the uncollided flux in (a) DOCTORS and (b) MCNP6 for the 16 cone-beam water benchmark problem.


(a)



(b)

Figure 5.19. Comparison of the uncollided flux in (a) DOCTORS and (b) MCNP6 for the 16 cone-beam water benchmark problem.


(a)

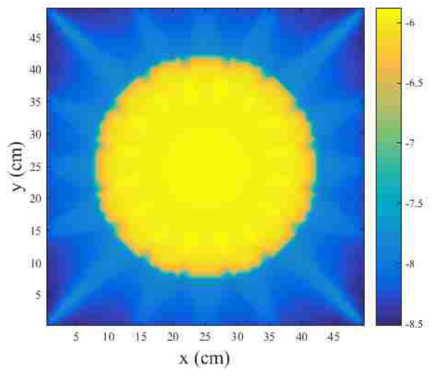

(b)

Figure 5.20. Comparison of the collided flux in (a) DOCTORS and (b) MCNP6 for the 16 cone-beam water benchmark problem.

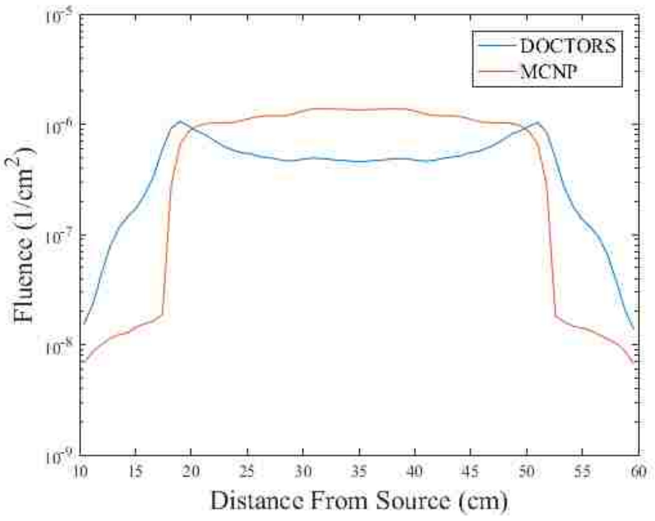
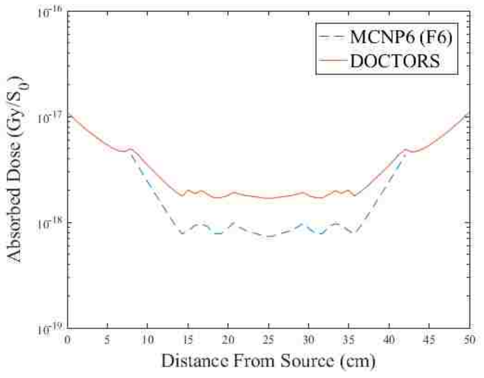


Figure 5.21. Vertical lineout of the collided flux for the 16 cone-beam water benchmark problem.

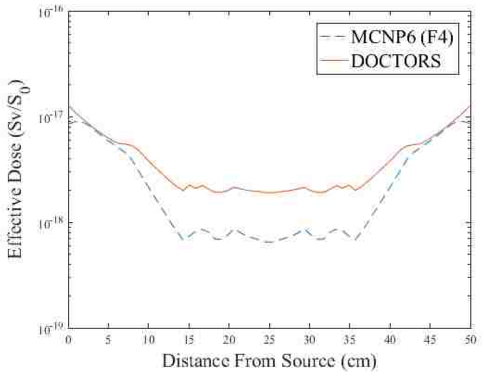
In some phantom models, a table the patient lies on is present. The table poses a challenge since it is a highly attenuating material relative to the patient which reduces the available flux. Further, the table is not mentioned in the literature which only consider the phantom. The CT number to material conversion developed by Ottosson and Behrens [2011] which is implemented in this work does not have an equivalent for the table region.

All voxels whose CT number is above the highest bone value given by Ottosson and Behrens [2011] are assumed to be part of the table. The table is assumed to be aluminum with a density of 2.70 g/cm^3. Figure 5.23 shows the materials before and after geometry simplification. Figure 5.23 shows the geometry of the full phantom before and after geometry simplification.

As with the previous benchmark problem, the raytracer provides very good agreement (well within 1-2% for most voxels) between MCNP6 and DOCTORS as shown in Figure 5.24. Because the "corner artifacts" cannot be removed in the full phantom model, those artifacts show up in the flux profile. Since those regions are only air, the artifacts'

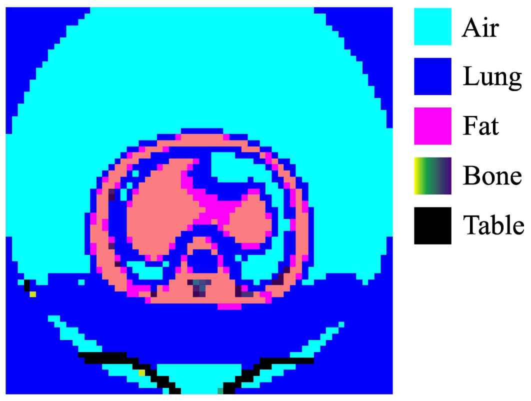


(a)

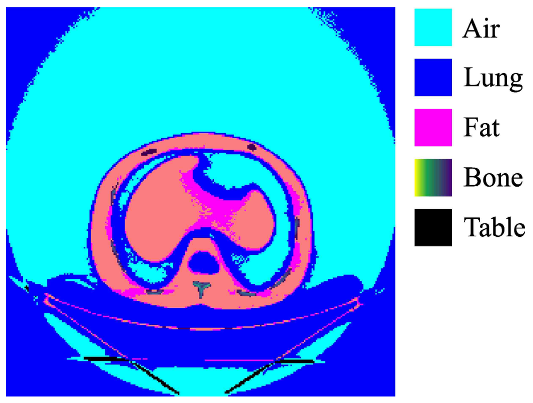


(b)

Figure 5.22. Comparison of the dose to the water phantom using (a) the energy deposition model and (b) the ICRP fluence-to-dose conversion model.

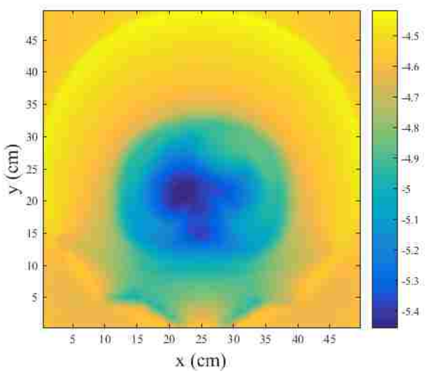


(a)



(b)

Figure 5.23. The phantom materials derived from the full phantom CT number data before (a) and after (b) geometry simplification.


(a)

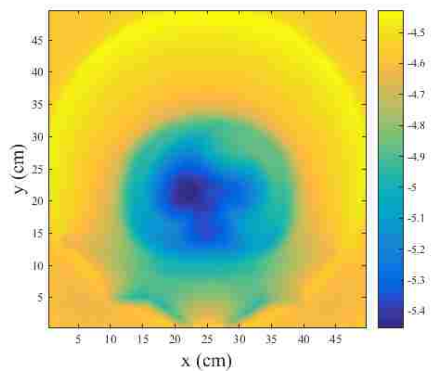
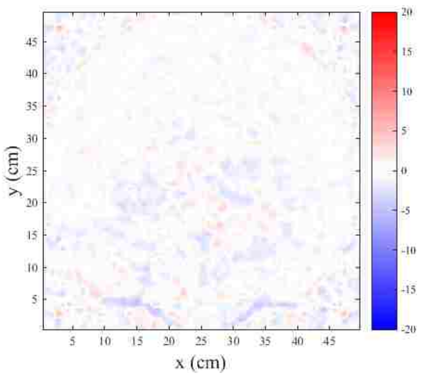
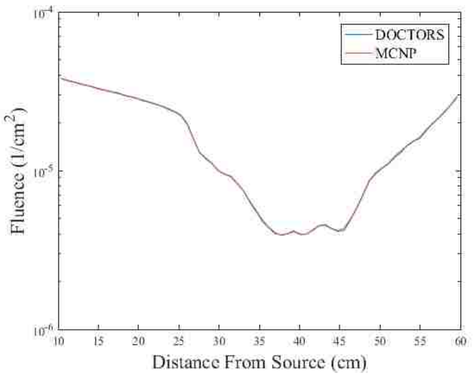

(b)

Figure 5.24. Comparison of the uncollided flux in (a) DOCTORS and (b) MCNP6 for the 64 cone-beam phantom benchmark problem.


(a)



(b)

Figure 5.25. The difference between the MCNP6 and DOCTORS uncollided distribution. (a) The 2D relative difference. (b) A vertical lineout down the center.

impact on the overall results are expected to be negligible. Otherwise, the uncollided has no apparent artifacts such as the triangle artifacts in the previous benchmark problem. Figure 5.25 shows that the profile between the two are effectively identical.

Similar to the water phantom benchmarks, DOCTORS underestimates the dose on the interior of the patient as shown in Figures 5.26 and 5.27. Again, this may be due to either the anisotropy treatment or the quadrature, further investigation will refeal the culprit.

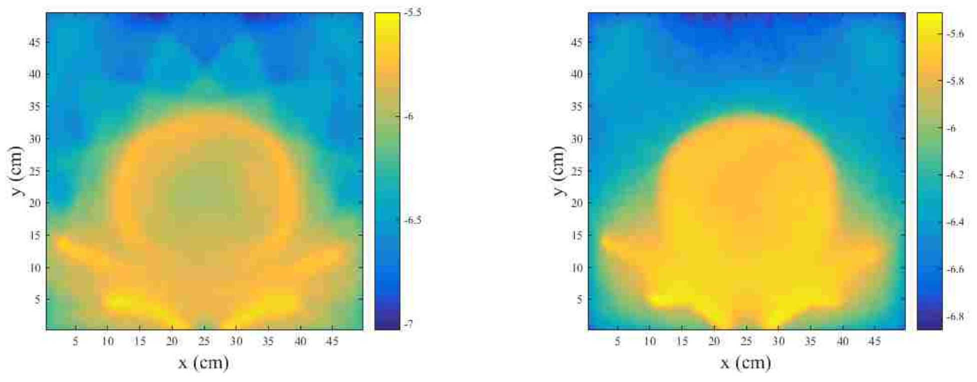


Figure 5.26. Comparison of the uncollided flux in (a) DOCTORS and (b) MCNP6 for the 64 cone-beam phantom benchmark problem.

5.5. GPU ACCELERATION

Table 5.2 summarizes the runtime required for the CPU-only version of DOCTORS and Table 5.3 summarizes the analogous runtime results for the GPU accelerated version. The speedup of the GPU over the CPU is given in Table 5.4. From those tables, a number of conclusions can be drawn.

The first, and most obvious, conclusion is that the speedup for large problems is much greater than the speedup for small problems. The GPU failed to accelerate the $64 \times 64 \times 16$ problems by more than a factor of a few which hardly warrants any acceleration at all, instead further CPU optimization would likely benefit more. Larger problems showed significant speedups though due to the reduced proportion of time spent in overhead operations. The most time consuming part of the GPU computation is launching a new kernel. Each kernel is itself very simple and can execute rapidly so small problems that have kernels with fewer parallel tasks do not perform as well.

The second, and more surprising, conclusion is that the speedup varied very little for single and double precision floating arithmetic. In fact, in most cases, the double precision outperformed the single precision! This result is unexpected in general, but even more so

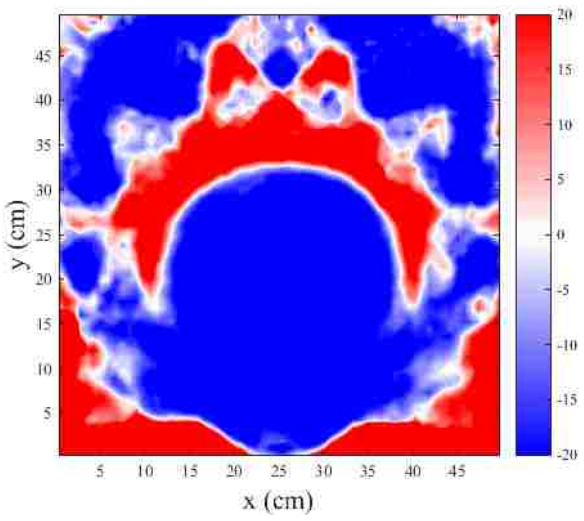


Figure 5.27. Comparison of the collided flux for the 64 cone-beam phantom benchmark problem.

on a GPU where single precision is vastly superior. The higher precision allows the double precision computations to converge more rapidly reducing the total number of iteraions necessary. Since a large portion of the overall time spent is in overhead operations, the time savings of using single precision are actually outweighed by the time savings of reducing the convergence iterations.

Table 5.2. CPU Runtime (msec)

Data Type	Method	$64 \times 64 \times 16$	$256 \times 256 \times 64$
`float`	Raytracer	820	125,000
	Solver	15,500	2,700,000
`double`	Raytracer	930	180,000
	Solver	14,300	2,400,000

Table 5.3. GPU Runtime (msec)

Data Type	Method	$64 \times 64 \times 16$	$256 \times 256 \times 64$
`float`	Raytracer	360	10,300
	Solver	2,420	71,600
`double`	Raytracer	363	13,500
	Solver	4,500	61,400

Table 5.4. Speedup

Data Type	Method	$64 \times 64 \times 16$	$256 \times 256 \times 64$
`float`	Raytracer	2.3	12.2
	Solver	6.4	37.6
`double`	Raytracer	2.6	13.5
	Solver	3.2	39.1

6. CONCLUSIONS

This section summarizes the conclusions drawn from the results given in Section 5 and suggests further improvements to DOCTORS.

6.1. ACCURACY

The raytracer is very fast compared to the discrete ordinate solution which computes the collided flux. The uncollided flux quantifies the source distribution from the medical system. This uniquely enables discrete ordinate solutions to rapidly characterize complex systems. Multiple cone beams and fan beams are possible as well as many other, more complex beam shapes.

Qualitatively, the most of the trends identified by DOCTORS agreed very well with MCNP6. The largest discrepancy was in the attenuation of particles through the patient region. However, since the raytracer is very accurate with respect to MCNP6, the difference is likely with the discrete ordinate methodology rather that cross section related. The key driver of this discrepancy is believed to be due to poor treatment of the angular transport either by way of the quadrature or anisotropy treatment.

This indicates that DOCTORS is better suited for problems that use a broad, distributed source as opposed to a narrow, directionally biased source. Such directional sources, require much higher quadratures to accurately characterize the beam due to ray effect.

6.2. APPLICABILITY TO CLINICAL SETTINGS

With further development, DOCTORS is well positioned to become a code of significant clincal impact. Though it is still lacking in medical diagnostic quality accuracy, it is easily extensible and capable of characterization of complex beams making it ideal for some types of clinical dosimetry.

Great effort was put into producing a code that is simple to use yet capable of producing powerful results quickly. However, usage in a clinical setting would likely require further refinement of the GUI to make the code more robust and intuitive. Additionally, the direct output of DOCTORS may not necessarily be of clinical importance since radiologists and technitians are more interested in compliance with regulations and ensuring patient safety. Therefore, a bulk report of overall numbers would likely be of more benfit than the entire spatial flux and dose distribution in a clinical setting.

6.3. GPU SPEEDUP

The GPU algorithm was implemented in CUDA and is very straightforward. At each subsweep, the global index of all voxels that can be computed in parallel are determined and a kernel is launched that executes those voxels. Unsurprisingly, this algorithms scales much better on larger problems where more time is spent in the computational execution of the kernal as opposed to the overhead operations of launching the kernel and memory transfer.

The GPU acceleration algorithm was found to speed the DOCTORS code up by a factor of 40x for large problems and only a factor of 3-6x for smaller problems. The improved accuracy of double precision arithmetic was found to outweigh the speed improvement from faster calculations.

6.4. FUTURE WORK

A number of simple modifications to DOCTORS could be made that would greatly increase the code's usability and robustness. One of these modifications, generation of more refined group structures, is not a change to DOCTORS *per se* but rather a change in the input cross section data. The other changes are additions to DOCTORS that would add new capabilities to the software. In addition to specific modifications, some additional, broader future goals can also be identified.

6.4.1. Anisotropy Treatment. A more sophisticated anisotropy treatment would be appropriate since the collided flux was found to have issues regarding its behavior. Alternatively, adding a spherical harmonics solver may help the anisotropy treatment as well, or at least reduce its memory footprint and runtime. This would require moving the solution from discrete angle space to flux moment space which can greatly complicate the debugging. However, once the correctness of the overall algorithm is shown, this step should be relatively simple.

6.4.2. Group Structure. The cross section data currently used by DOCTORS are taken from SCALE6.2. While these cross sections have been found to be sufficient to produce flux distributions in medical CT imaging, a more refined group structure designed for medical applications would be worth investigation. Also, the data distribtuion from SCALE6.2 contains potentially export controlled information since it also includes nuclear reactor materials. A medical purposed cross section library would alleviate this problem and allow the code to be freely released with a dataset. In the current version of DOCTORS, the user is required to obtian the cross section data files independently. This work can be done using either NJOY or the newly released AMPX code. Either code has the capability to collapse an ENDF formatted data file into a group structure of the user's choosing in the format DOCTORS reads.

6.4.3. Therapy Extension. An advantage of the raytracer is that it is very fast. Therefore, it can characterize many beams and integrate them temporally easily. This would allow DOCTORS to model more complex beam shapes and scan protocols accurately. This would be particularly useful for medical treatment systems that use a multileaf collimator.

Some treatment systems employ a multileaf collimator to continuously shape the beam, resulting in a large dose deposition only at the area of interest. DOCTORS can likely be extended to high energy therapeutic beams for clinical treatment. However, higher energy photons will scatter more anisotropicly. This will require careful analysis of the angular treatment used.

6.4.4. Partial Acceptance Criteria. In the current version of doctors, the raytracer is very accurate with respect to MCNP6 in predicting the uncollided flux, except for along the periphery of the beam. Voxels are currently either completely inside the beam or completely outside of it as determined by its isocenter. A more sophisticated technique whereby a voxel cah be partially accepted in the beam would remove these artifacts.

6.4.5. Organ Identification. Currently, DOCTORS cannot automatically identify specific organs, thus it is only able to compute the equivalent dose via dose deposition. If specific organs could be identified, tissue specific weighting factors could be applied resulting in the effective dose to each organ which would be of greater clinical significance. However, such identification currently requires additional input about the scan protocol and extensive knowledge about the human anatomy not currently integrated into DOCTORS.

APPENDIX

LEGENDRE POLYNOMIALS AND SPHERICAL HARMONICS

This appendix gives a brief summary of flux moments and the Legendre spherical harmonics needed to compute them.

FLUX MOMENTS

The flux moments are an alternate representation of the angular flux within a system. The rationale behind usage of flux moments is that they map the discrete flux (ψ) from a function of angle to a function of P_N expansion. This reduces the amount of memory required for computation. Storing ψ directly requires storing $G \times N_V \times N_a$ values where G is the number of groups, N_V is the number of voxels, and N_a is the number of angles. Storing the flux moments, however, requires storing only $G \times N_V \times (N^2 + 2N + 1)$ values where N is the P_N expansion value. For example, a problem requiring S_6 needs $6 \times 8 = 48$ directions, but a problem requiring P_6 needs 49 expansion coefficients which would seem approximately eqivalent, but typical problems require far more discrete directions than expansion coefficients. Complex problems can easily require S_{16} or higher but expansions above P_5 are rarely encountered.

LEGENDRE POLYNOMIALS

The Legendre polynomials are solutions to:

$$\begin{aligned} P_0(\mu) &= 1 \\ P_l(\mu) &= \frac{1}{2^l l!}\frac{d^l}{d\mu^l}(\mu^2 - 1)^l, \quad l \in \mathbb{N} \end{aligned} \tag{A.1}$$

where $\mathbb{N}$ is the set of natural numbers (1, 2, 3...). Table 7.1 gives the solution of the first few Legendre polynomials. The Legendre polynomials are orthogonal on the domain $[-1, 1]$ in that

$$\frac{1}{2}\int_{-1}^{1} P_l(\mu)P_{l'}(\mu)d\mu = \begin{cases} \frac{1}{2l+1}, & l = l' \\ 0, & \text{otherwise} \end{cases} \tag{A.2}$$

holds.

Table 7.1. Legendre Polynomials

P_l	$P_l(\mu)$
P_0	1
P_1	μ
P_2	$\frac{1}{2}(3\mu^2 - 1)$
P_3	$\frac{1}{2}(5\mu^3 - 3\mu)$
P_4	$\frac{1}{8}(35\mu^4 - 30x^2 + 3)$
P_5	$\frac{1}{8}(63\mu^5 - 70\mu^3 + 15\mu)$
P_6	$\frac{1}{16}(231\mu^6 - 315^4 + 105^2 - 5)$
P_7	$\frac{1}{16}(429\mu^7 + 693\mu^5 - 315^3 + 35^3\mu)$

The orthogonality of the Legendre polynomials allows an infinite series to exactly represent any function on the domain $[-1, 1]$ as

$$f(\mu) = \sum_{l=0}^{\infty} C_l P_l(\mu) \tag{A.3}$$

with apprpriately selected constants, C_l and is used to approximate functions arbitrarily

$$f(\mu) \approx \sum_{l=0}^{N} C_l P_l(\mu). \tag{A.4}$$

The Legendre polynomials are extended into the associated Legendre polynomials which have an additional orthogonality.

ASSOCIATED LEGENDRE POLYNOMIALS

The associated Legendre polynomials are defined as the solution to

$$\begin{aligned} P_l^m(\mu) &= (-1)^m (1-\mu^2)^{m/2} \frac{d^m}{d\mu^m} P_l(\mu) \\ P_l^0(\mu) &= P_l(\mu) \end{aligned} \tag{A.5}$$

and exhibit the following orthogonality on $[-1, 1]$:

$$\frac{1}{2}\int_{-1}^{1} P_l^m(\mu) P(\mu) d\mu = \begin{cases} \frac{1}{2l+1}\frac{(l+m)!}{(l-m)!}, & l = l' \text{ and } m = m' \\ 0, & \text{otherwise.} \end{cases} \tag{A.6}$$

The spherical harmonics take advantage of the double orthogonality to approximate 2D distributions.

SPHERICAL HARMONICS

The spherical harmonics are defined as:

$$Y_{lm}(\hat{\Omega}) = \sqrt{\frac{(2l+1)(l-m)!}{(l+m)!}} P_l^m(\mu) e^{e\tau m} \tag{A.7}$$

where μ is the cosine of the angle between $\hat{\Omega}$ and the x-axis and τ is the rotation about the x-axis with respect to the y-axis of $\hat{\Omega}$ projected onto the yz plane which can be compuated as:

$$\tau = \frac{\eta}{\sqrt{\eta^2 + \xi^2}}. \tag{A.8}$$

The spherical harmonics have the orghogonality

$$\int Y_{lm}(\hat{\Omega}) Y^*_{l'm'}(\hat{\Omega}) d\hat{\Omega} = \begin{cases} 1, & l = l' \text{ and } m = m' \\ 0, & \text{otherwise} \end{cases} \tag{A.9}$$

where Y_{lm}^* is the complex conjugate of Y_{lm}. Using the addition theorem which states

$$P_l(\hat{\Omega} \cdot \hat{\Omega}') = \frac{1}{2l+1} \sum_{m=-l}^{l} Y_{lm}^*(\hat{\Omega}') Y_{lm}(\hat{\Omega}) \tag{A.10}$$

gives

$$P_l(\hat{\Omega} \cdot \hat{\Omega}') = P_l(\mu) P_l(\mu') + 2 \sum_{m=1}^{l} \frac{(l-m)!}{(l+m)!} P_l^m(\mu) P_l^m(\mu') cos(m[\tau - \tau']). \tag{A.11}$$

REFERENCES

J. H. Hubbell and S. M. Seltzer. Tables of X-Ray Mass Attenuation Coefficients and Mass Energy-Absorption Coefficients from 1 keV to 20 MeV for Elements Z = 1 to 92 and 48 Additional Substances of Dosimetric Interest. Technical Report NISTIR 5632, Radiation Physics Division, PML, NIST, Gaithersburg, MD, 1996.

G. N. Hounsfield. Computerized transverse axial scanning (tomography): Part I. Description of system. *British Journal of Radiology*, 46:1016–1022, 1973.

D. J. Brenner and E. J. Hall. Computed tomography-An increasing source of radiation exposure. *New England Journal of Medicine*, 357:2277–2284, 2007.

A. J. Einstein, M. J. Henzlova, and S. Rajagopalan. Estimating risk of cancer associated with radiation exposure from 64-slice computed tomography coronary angiography. *JAMA*, 298:317–323, 2007a.

Richard R. Monson, James E. Cleaver, Herbert L. Abrams, Eula Bingham, Patricia A. Buffler, Elisabeth Cardis, Roger Cox, Scott Davis, William C. Dewey, and Ethel S. Gilbert. Biological effects of ionizing radiation (BEIR) report VII: Health risks from exposure to low levels of ionizing radiation. Technical report, National Research Council, Washington, D.C., 2004.

A. J. Einstein, K. W. Moser, R. C. Thompson, M. D. Cerqueira, and M. J. Henzlova. Radiation dose to patients from cardiac diagnostic imaging. *Circulation*, 116:1290–1305, 2007b.

C. H. McCollough, A. N. Primak, N. Braun, J. Kofler, L. Yu, and J. Christner. Radiation dose to patients from cardiac diagnostic imaging. *Radiologic Clinics of North America*, 47:27–40, 2009.

Lifeng Yu, Xin Liu, Shuai Leng, James M. Kofler, Juan C. Ramirez-Giraldo, Mingliang Qu, Jodie Christner, Joel G. Fletcher, and Cynthia H. McCollough. Radiation dose reduction in computed tomography: techniques and future perspective. *Imaging Medicine*, 1(1): 65–84, 2009.

Tinsu Pan, Ting-Yim Lee, Eike Rietzel, and George T. Y. Chen. 4D-CT imaging of a volume influenced by respiratory motion on multi-slice CT. *Medical Physics*, 31(2):333–340, 2004.

Anthony Brinton Wolbarst. *Physics of Radiology*. Medical Physics Publishing, 2 edition, 2005.

Walter Huda and Fred A. Mettler. Volume CT Dose Index and Dose-Length Product Displayed during CT: What Good Are They? *Radiology*, 258(1):236–242, 2011.

K. Lau, J. Sippel, A. Westerink, K. Dobeli, and J. Younger. Comparison of Radiation dose Between Computer Tomography Coronary Angiography (CTCA) and Invasive Coronary Angiograpy (ICA) in Severely Obese Patients. *Hear, Lung, and Circulation*, 25(2): S221–S222, 2016.

W. Ralph Nelson and Clive Field. Comparison of EGS5 simulations with experiment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometer, Detectors and Associated Equipment*, 572(3):1083–1093, 2007.

Francesc Salvat. The PENELOPE code system. Specific features and recent improvements. *Annals of Nuclear Energy*, 82:98–109, 2015.

J. Perl, J. Shin, J. Schümann, B. Faddegon, and H. Paganetti. TOPAS: An innovative proton Monte Carlo platform for resarch and clinical applications. *Medical Physics*, 39(11): 6818–6837, 2012.

S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, F. Behner, L. Bellagamba, J. Boudreau, L. Broglia, A. Brunengo, and H. Burkhardt. GEANT4 - a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometer, Detectors and Associated Equipment*, 506(3):250–303, 2003.

T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. J. Cox, J. Durkee, J. Elson, M. Fensin, R. A. Forster, J. Hendricks, H. G. Hughes, R. Johns, B. Kiedrowski, R. Martz, S. Mashnik, G. McKinney, D. Pelowitz, R. Prael, J. Sweazy, L. Waters, T. Wilcox, and T. Zukaitis. Features of MCNP6. *Annals of Nuclear Energy*, 87(2):772–783, 2016.

Xun Jia, Jan Schümann, Harald Paganetti, and Steve B. Jiang. GPU-based fast Monte Carlo dose calculation for proton therapy. *Physics in Medicine and Biology*, 57(23):7783–7797, 2012.

J. R. Askew. A characteristics formulation of the neutron transport equation in complicated geometries. Technical Report AEEW-M-1108, United Kingdom Atomic Energy Authority, Reactor Group, Winfrith, United Kingdom, 1972.

M. Hursin, B. Collins, Y. Xu, and T. Downar. The Development and Implementation of a One-Dimensional S_n Method in the 2D-1D Integral Transport Solution. *Nuclear Science and Engineering*, 176:186–200, 2014.

M. L. Williams, D. Ilas, E. Sajo, D. B. Jones, and K. E. Watkins. Deterministic photon transport calculations in general geometry for external beam radiation therapy. *Medical Physics*, 30(12):3183–3195, 2003.

Adam J. Hoffman and John C. Lee. A time-dependent neutron transport method of characteristics formulation with time derivative propagation. *Journal of Computational Physics*, 307:696–714, 2016.

ICRP. Conversion Coefficients for Radiological Protection Quantities for External Radiation Exposures. ICRP Publication 116. *Annals of the ICRP*, 40(2-5), 2010.

Russell M. Mersereau and Alan V. Oppenheim. Digital Reconstruction of Multidemensional Signals from Their Projections. *Proceedings of the IEEE*, 62(10):1319–1338, 1974.

Guy Besson. CT image reconstruction from fan-parallel data. *Medical Physics*, 26(3): 415–426, 1999.

H. Turbell. *Cone-beam reconstruction using filtered backprojection*. PhD thesis, Linköping University, Sweden, 2001.

François Pontana, Alain Duhamel, Julien Pagniez, Thomas Flohr, Jean-Baptiste Faivre, Anne-Lise Hachulla, and Jacques Remy. Chest computed tomography using iterative reconstruction vs filtered back projection (Part 2): image quality of low-dose CT examinations in 80 patients. *European Radiololgy*, 21(3):636–643, 2011.

John R. Lamarsh and Anthony J. Baratta. *Introdution to Nuclear Engineering*. Prentice Hall, 3 edition, 2001.

F. C. P. du Plesis, C. A. Willemse, M. G. Lötter, and L. Goedhals. The indirect use of CT numbers to establish material properties needed for Monte Carlo calculation of dose distributions in patients. *Medical Physcis*, 25(7):1195–1201, July 1998.

Cheng B. Saw, Alphonse Loper, Krishna Komanduri, Tony Combine, Saiful Huq, and Carol Scicutella. Determination of CT-to-density Conversion Relationship for Image-based Treatment Planning Systems. *Midical Dosimetry*, 30(3):145–148, 2005.

Uwe Schneider, Eros Pedroni, and Antony Lomax. The calibration of CT Hounsfield units for radiotherapy treatment planning. *Physics in Medicine and Biology*, 41:111–124, 1996.

J. A. Halbleib, R. P. Kensek, G. D. Valdez, S. M. Seltzer, and M. J. Berger. ITS: the integrated TIGER series of electron/photon transport codes-Version 3.0. *IEEE Transactions on Nuclear Science*, 39(4):1025–1030, 1992.

Wilfried Schneider, Thomas Bortfeld, and Wolfgang Schlegel. Correlation between CT numbers and tissue parameters needed for Monte Carlo simulations of clinical dose distributions. *Physics in Medicine and Biology*, 45:459–478, 2000.

Hyung Dong Kim, Byung Yong Kim, Eng Chan Kim, Sang Mo Yun, Jeong Ku Kang, and Sung Kyu Kim. Comparison of Dose Distributions for Hounsfield Number Conversion Methods in GEANT4. *Journal of the Korean Physical Society*, 64(12):1912–1918, June 2014.

Barbara Vanderstraeten, Pil Wai Chin, Michael Fix, Antonio Leal, Grisel Mora, Nick Reynaert, Joao Seco, Martin Soukup, Emiliano Spezi, Wilfried De Neve, and Hubert Thierens. Conversion of CT numbers into tissue parameters for Monte Carlo dose calculations: a multi-centre study. *Physics in Medicine and Biology*, 52:539–562, 2007.

Rickard O. Ottosson and Claus F. Behrens. CTC-ask: a new algorithm for conversion of CT numbers to tissue parameters for Monte Carlo dose calculations applying DICOM RS knowledge. *Physics in Medicine and Biology*, 56(22):N263–N274, 2011.

S. Chandrasekahar. *Radiative Transfer*. 1950.

E. E. Lewis and W. F. Miller, Jr. *Computational Methods of Neutron Transport*. American Nuclear Society, 1993.

B G. Carlson. Solution of the Transport Equation by S_n Approximations. Technical Report LA-1981, Los Alamost National Laboratory, Los Alamos, NM, 1953.

K. D. Lanthrop and B. G. Carlson. Discrete Ordinates Angular Quadrature of the Neutron Transport Equation. Technical Report LA-3186, Los Alamost National Laboratory, Los Alamos, NM, 1965.

W. A. Rhoades and D. B. Simpson. The TORT Three-Dimensional Discrete Ordinates Neutron/Photon Transport Code. Technical Report ORNL/TM-13221, Oak Ridge National Laboratory, Oak Ridge, Tennessee, October 1997.

Mark David DeHart. *A Discrete Ordinates Approximation to the Neutron Transport Equation Applied to Generalized Geometries*. PhD thesis, Texas A&M University, 1993.

Todd A. Wareing, Jim E. Morel, and Donald K. Parsons. A First Collision Source Method for ATTILA, An Unstructured Tetrahedral Mesh Discrete Ordinates Code. Technical Report LA-UR-98-1373, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1998.

William J. Walters. *Development of the Adaptive Collision Source Method for Discrete Ordinates Radiation Transport*. PhD thesis, Virginia Polytechnic Instituute and State University, 2015.

Cory D. Ahrens. Lagrange Discrete Ordinates: a new angular discretization for the three dimensional linear Boltzmann equation. *Nuclear Science and Engineering*, 180:273–285, 2015.

Ahmad M. Ibrahim, Paul P. H. Wilson, Mohammed E. Sawan, Scott W. Mosher, Douglas E. Peplow, John C. Wagner, Thomas M. Evans, and Robert E. Grove. Automatic Mesh Adaptivity for Hybrid Monte Carlo/Deterministic Neutronics Modeling of Difficult Shielding Problems. *Nuclear Science and Engineering*, 181:48–59, 2015.

B. Lee. Space-angle-energy multigrid methods for S_n discretizations of the multi-energetic Boltzmann equation. *Numerical Linear Algebra with Applications*, 19:773–795, 2012.

Dmitry Efremenko, Adrian Doicu, Diego Loyola, and Thomas Trautmann. Acceleration techniques for the discrete ordinate method. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 114:73–81, 2013.

Thomas M. Evans, Alissa S. Stafford, Rachel N. Slaybaugh, and Kevin T. Clarno. DENOVO: A New Three-Dimensional Parallel Discrete Ordinates Code in SCALE. *Nuclear Technology*, 171:171–200, 2010.

Tom Evans, Greg Davidson, Josh Jarrell, Steven Hamilton, Seth Johnson, and Tara Pandya. Exnihilo Transport Methods Manual. Technical report, Radiation Transport Group, Reactor and Nuclear Systems Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 2006.

Zheng Ying, Zhang Bin, Chen Meng-Teng, Zhang Liang, Chen Yi-Xue, Yin Wen, and Liang Tian-Jiao. Application of the Fist Collision Source Method to CSNS Target Station Shielding Calculation. *Chinese Physics C*, 40, 2015.

J. E. Bresenham. Algorithm from Computer Control of a Digital Plotter. *IBM System Journal*, 4(1):25–30, 1965.

Y. K. Liu, B. Žalik, and H. Yang. An Integer One-Pass Algorithm for Voxel Traversal. *Computer Graphics*, 23:167–172, 2004.

J. C. Cleary and G. Wyvill. Analysis of an Algorithm for Fast Ray Tracing using Uniform Space Subdivision. *The Visual Computer*, 4(2):65–83, 1988.

John Amanatides and Andrew Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *Eurographics ’87*, pages 3–10, 1987.

Lijun He, Shuang Liu, Jian Yun, and Yongkui Liu. A Line Generation Algorithm over 3D Body-centered Cubic Lattice. *Journal of Multimedia*, 8(1):40–47, February 2013.

IEEE. IEEE Standard for Floating-Point Arithmetic. Technical report, IEEE Computer Society, New York, NY, 2008.

J. K. Shultis and R. E. Faw and. *Fundamentals of Nuclear Science and Engineering*. 1950.

Kirk A. Mathews. On the Propagation of Rays in Discrete Ordinates. *Nuclear Science and Engineering*, 132:155–180, 1999.

John Tencer. Ray Effect Mitigation Through Reference Frame Rotation. *Journal of Heat Transfer*, 138(11):112701–1–112701–11, 2016.

D. E. Cullen, J. H. Hubbell, and L. D. Kissel. EPDL97: the Evaluated Photon Data Library, 97 Version. Technical report, Lawrence Livermore National Laboratory, Livermore, CA, 1997.

NNDC. sigma: Evaluated Nuclear Data File (ENDF) Retrieval & Plotting. Technical report, National Nuclear Data Center, Brookhaven National Laboratory, Upton, NY, 2011.

VITA

Edward Norris graduated from Missouri University of Science and Technology in 2013 with a B.S. in both nuclear engineering and computer science. In July 2017, he received his PhD degree in Nuclear Engineering from Missouri University of Science and Technology. His PhD work was funded by a fellowship through the NRC. Edward was awarded the College of Engineering and Computing PhD Scholar award in 2017 for his academic achievements.

Previously he participated in four summer internships at Sandia National Laboratories. While there, he served in various capacities ranging from cyber defense to radiation detection and neutron scatter simulation. He was largely involved with numerical simulations of radiation transport using Geant4 and MCNP6 as well as data collection.

Edward was also an active member of the Nuclear Science Design Team at Missouri University of Science and Technology. While on the team he solved radiation transport problems to facilitate the dosimetry of an inertial electrostatic confintement fusion device being constructed.