

Winter 12-15-2017

# Fast Monte Carlo Simulations for Quality Assurance in Radiation Therapy

Yuhe Wang

*Washington University in St. Louis*

Follow this and additional works at: [https://openscholarship.wustl.edu/art\\_sci\\_etds](https://openscholarship.wustl.edu/art_sci_etds)



Part of the [Physics Commons](#)

---

## Recommended Citation

Wang, Yuhe, "Fast Monte Carlo Simulations for Quality Assurance in Radiation Therapy" (2017). *Arts & Sciences Electronic Theses and Dissertations*. 1205.

[https://openscholarship.wustl.edu/art\\_sci\\_etds/1205](https://openscholarship.wustl.edu/art_sci_etds/1205)

This Dissertation is brought to you for free and open access by the Arts & Sciences at Washington University Open Scholarship. It has been accepted for inclusion in Arts & Sciences Electronic Theses and Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact [digital@wumail.wustl.edu](mailto:digital@wumail.wustl.edu).

WASHINGTON UNIVERSITY IN ST. LOUIS

Department of Physics

Dissertation Examination Committee:

Zohar Nussinov, Chair

Harold Li, Co-Chair

Anders Carlsson

Li Yang

Deshan Yang

Fast Monte Carlo Simulations for Quality Assurance in Radiation Therapy

by

Yuhe Wang

A dissertation presented to  
The Graduate School  
of Washington University in  
partial fulfillment of the  
requirements for the degree  
of Doctor of Philosophy

December 2017  
St. Louis, Missouri

© 2017, Yuhe Wang

# Table of Contents

List of Figures .....	v
List of Tables .....	viii
Acknowledgments.....	ix
Abstract of the Dissertation .....	xii
Chapter 1: Introduction.....	1
1.1 History of Monte Carlo simulation .....	2
1.2 Workflow of radiation therapy.....	4
1.3 MRgRT: MRIdian platform.....	6
1.4 Ways to improve efficiency .....	7
1.5 Overview of the dissertation .....	8
Chapter 2: Monte Carlo basics .....	10
2.1 Random-sampling methods .....	10
2.1.1 Pseudo random number generator.....	10
2.1.2 Continuous variable: inverse-transform sampling .....	13
2.1.3 Continuous variable: rejection sampling.....	14
2.1.4 Discrete variable: reverse-transform sampling .....	15
2.1.5 Discrete variable: Walker’s aliasing method .....	15
2.2 Radiation transport model.....	17
2.2.1 Scattering model .....	17
2.2.2 Random tracks.....	18
2.3 Photon interactions.....	21
2.3.1 Rayleigh scattering.....	22
2.3.2 Photoelectric absorption.....	23
2.3.3 Compton scattering .....	24
2.3.4 Pair production .....	25
2.4 Electron/positron interactions .....	26
2.4.1 Continuous energy loss .....	27
2.4.2 Multiple Scattering.....	29
2.4.3 Random hinge .....	30
Chapter 3: GPU acceleration .....	32



3.1	GPU architecture vs CPU architecture .....	32
3.2	Introduction of CUDA .....	34
3.2.1	CUDA Kernel .....	34
3.2.2	Thread Hierarchy .....	35
3.2.3	Memory Hierarchy .....	36
3.2.4	Difficulties of GPU programming .....	39
3.3	Prototype: PENELOPE .....	40
3.4	PENELOPE in C++: cPENELOPE .....	41
3.4.1	Rewrite PENELOPE in C++ .....	41
3.4.2	User-friendly features.....	42
3.4.3	Transport in heterogeneous phantom .....	43
3.4.4	Transport in magnetic field .....	44
3.4.5	Validate cPENELOPE .....	45
3.5	cPENELOPE to gPENELOPE.....	46
3.5.1	GPU workflow .....	46
3.5.2	MRIdian head model.....	49
3.5.3	Performance benchmark.....	49
3.6	Validate gPENELOPE .....	50
3.6.1	3D dose comparisons .....	50
3.6.2	Comparisons in single thread .....	51
3.6.3	Comparisons in multithreads.....	53
3.7	Application 1: validate MRIdian head model .....	53
3.7.1	Depth dose.....	54
3.7.2	Off-axis profile.....	54
3.7.3	Output factor .....	55
3.7.3	AAPM TG-119 .....	56
3.8	Application 2: Magnetic effect on MRIdian .....	59
3.9	Application 3: validate MRIdian's treatment plans .....	61
3.10	Discussion and conclusion .....	63
Chapter 4:	Transport simplification & variance reduction .....	66
4.1	Introduction of online ART .....	66
4.2	Transport simplification.....	68

4.3	gDPM with variance reduction .....	71
4.4	GPU implementation: gDPMvr .....	72
4.5	Accuracy and performance benchmarks .....	75
4.5.1	Phantoms .....	75
4.5.2	Homogeneous water phantom.....	77
4.5.3	Water-lung-water phantom .....	79
4.5.4	Water-lung-tumor-water phantom .....	81
4.5.4	Clinical patients.....	85
4.6	Discussion and conclusion .....	88
Chapter 5:	DVH constraint .....	92
5.1	Introduction of DVH .....	92
5.2	Gamma passing rate vs DVH consistency .....	94
5.3	Focus on the important region.....	95
5.4	Accuracy and performance.....	97
5.5	Discussion and conclusion .....	100
Chapter 6:	Geometry system .....	101
6.1	Regularly shaped model .....	102
6.2	Arbitrary triangle-mesh model .....	106
6.3	Arbitrary tetrahedron-mesh model .....	108
Chapter 7:	Graphic user interface .....	110
7.1	DoseViewer .....	110
7.2	TetViewer.....	111
Chapter 8:	Conclusion and outlook .....	113
8.1	Summary of the results .....	113
8.2	Outlook .....	114
Bibliography	.....	116

# List of Figures

Figure 1.1 Simplified workflow of modern radiation therapy.....	5
Figure 1.2 Devices to measure dose: (left) ion-chamber, point dose; (middle) radiographic film, 2D dose; (right) arch-check, surface dose in 3D.....	6
Figure 2.1 Illustration of rejection sampling.....	14
Figure 2.2 Graphical representation of the inverse transform method (top) and Walker's aliasing method (bottom) for random sampling from a discrete distribution. In this example, the random variable can take the values $i = 1, 2, 3$ and 4 with relative probabilities.....	16
Figure 2.3 (left) The model for analyzing the probability of being scattered. (right) Angular deflection in single-scattering events.....	19
Figure 2.4 Four types of photon interactions with matter.....	22
Figure 2.5 Various notations for inner atomic electron shells (left) and allowed radiative transitions (right) to these shells .....	23
Figure 2.6 Four types of electron interactions with matter .....	26
Figure 2.7 (left) Real curved electron path vs simulated electron path as a straight line. The accumulated scattering angle is applied at the end of this CH step. (right) The scattering angle is applied at a random point along the path, i.e. through random hinge method.....	30
Figure 3.1 (left) The architecture of GPU (right) The architecture of CPU .....	33
Figure 3.2 Floating-Point Operations per Second for the CPU and GPU .....	33
Figure 3.3 Memory architecture corresponding to thread architecture in CUDA .....	36
Figure 3.4 The sample of “C”-style configuration script.....	42
Figure 3.5 The setup with cone beam and cubic water phantom to test the output of cPENELOPE.....	46
Figure 3.6 Workflow of gPENELOPE including (a) Initialization (b) Generate photons (c) Copy photons to GPUs (d) GPU kernel calls: RA, CO, PH and PP are short for Rayleigh, Compton, photoelectric and pair production while EL, IN, BR, SI, AN are short for elastic, inelastic, bremsstrahlung, shell ionization and annihilation respectively. The “kernel-by-kernel” calls in sequence can help reducing instruction divergence. (e) Clean up. ....	47

Figure 3.7 Dose distributions for a lung case calculated by C++ PENELOPE (left) and gPENELOPE (middle). The two distributions are identical within 1%. The frequency distribution of z-scores in comparison to a standard normal distribution (right). .....	52
Figure 3.8 Percentage-depth-dose comparisons between gPENELOPE and ionization chamber measurement (spline interpolated) for field sizes 4.2×4.2, 10.5×10.5 and 27.3×27.3 cm <sup>2</sup> at 100 cm SSD (a). Off-axis profile comparisons between gPENELOPE and radiochromic film measurement for field sizes 2.1×2.1, 4.2×4.2, 10.5×10.5, and 21.0×21.0 cm <sup>2</sup> at depths of 5 (b), 10 (c) and 15 cm (d) at 100 cm SSD. ....	55
Figure 3.9 Isodose and profile comparison between gPENELOPE and radiographic film measurement: (a) AP-PA, (b) Prostate, (c) C-shape, (d) Multi-target, where solid lines represent gPENELOPE and dashed lines represents film measurement. (e) Profile of C-shape along the right-to-left central axis, (f) Profile of Multi-target along the inferior-to-superior central axis. Note that the circled points are located in the high gradient region. ....	59
Figure 3.10 Water-lung-water phantom, <sup>60</sup> Co beam and magnetic field configuration. Voxel size is 1×1×1 mm <sup>3</sup> . (b) Central axis depth dose profiles. Larger magnetic field results in larger dose distortion. (c) Dose distributions for configuration in (a) at indicated magnetic field strengths. ....	60
Figure 4.1 (a) CSDA range of an electron vs. kinetic energy in water. (b) Dose distribution calculated for a clinical IMRT plan. The iso-dose lines are shown relative to the maximum dose. The area enclosed by the red lines indicate the area of energy deposition by photons with energies greater than 743 KeV. ....	70
Figure 4.2 Workflow of gDPMvr including 5 modules: (a) Initialization, (b) Generate photons, (c) Copy photons to GPUs, (d) GPU kernel, and (e) Clean up. ....	74
Figure 4.3 Phantoms used to evaluate accuracy and performance of gDPM, gDPMvr, and gPENELOPE. (a) 30.3×30.3×30.3 cm <sup>3</sup> uniform water phantom, (b) water-lung-water phantom where lung's height is 9.9 cm, (c) water-lung-tumor-water phantom where tumor size is 2.1×2.1×2.1 cm <sup>3</sup> , and (d) patient. Voxel sizes are all set to 3×3×3 mm <sup>3</sup> . .....	76
Figure 4.4 Percentage depth dose (upper row) and off-axis profiles (lower row, 5 cm depth) for a homogeneous water phantom. FS: field size. ....	78
Figure 4.5 z-score histograms among gPEN, gDPM and gDPMvr for a homogeneous water uniform phantom. FS: field size. ....	78
Figure 4.6 Percentage depth dose (upper row) and off-axis profiles (middle row: 5 cm depth and inside the water, lower row: 15 cm depth and inside the lung) for a water-lung-water phantom. FS: field size. ....	80

Figure 4.7 z-score histograms among gPEN, gDPM and gDPMvr for a water-lung-water phantom.....	81
Figure 4.8 Percentage depth dose and off-axis profile for a water-lung-water phantom. First row: percentage depth dose. Second to fourth rows: off-axis profiles at depth of 5, 12 and 15 cm, i.e. in the water, the lung, the tumor respectively. FS: field size. ....	83
Figure 4.9 z-score histograms among gPEN, gDPM and gDPMvr for a water-lung-tumor-water phantom.....	84
Figure 4.10 Z-score histograms for 7 IMRT plans .....	87
Figure 5.1 Cumulative DVHs from a radiotherapy plan.....	93
Figure 5.2 2D illustration of the reverse free walk. The yellow pixel belongs to the important region while the rest belong to the unimportant region.....	96
Figure 5.3 (left) dose distribution calculated by gDPMvr. (right) dose distribution calculated by gDVH with PTV as the important regions.....	98
Figure 5.4 (left) comparison of line dose profiles. (right) comparison of DVHs (PTV).....	98
Figure 5.5 (left) gamma distribution of gDPMvr vs KMC. (right) gamma distribution of gDVH vs KMC. ....	99
Figure 6.1 (left) Non-planar reduced quadric surfaces and their indices in PENELOPE. (right) A skeleton constructed by PENELOPE's quadratic surfaces.....	102
Figure 6.2 (left) Model of A18 ion-chamber. (right) Model of Farmer ion-chamber. ....	103
Figure 6.3 (left) Dolphin modeled by triangular mesh. (right) demonstration of spatial octree division of triangles.....	107
Figure 6.4 Human body modeled by tetrahedron mesh. It provides natural containers for spatial dose counting, and can be easily deformed to simulate the motion of patients' organs.....	108
Figure 7.1 GUI of DoseViewer that enables DICOM phantom creation/conversion, dose calculation, multiple views, 3D observation, dose profiling and gamma analysis (GPU accelerated), etc. in a compact size. The smaller window, as an example, shows an arbitrary dose profile comparison between gPENELOPE and KMC (engine of MRIdian's treatment planning system) in a lung case.....	111
Figure 7.2 GUI of TetViewer that provides 3D display of tetrahedron meshes and radiation dose. ....	112

# List of Tables

Table 3.1 Particle status differences of $10^7$ steps between gPENELOPE and C++ PENELOPE .....	52
Table 3.2 Output factor comparison for square and rectangular fields.....	56
Table 3.3 TG-119 point dose comparisons: gPENELOPE vs. ionization chamber (IC) .....	57
Table 3.4 Gamma passing rates (2%/2 mm and 10% threshold) and z-scores distributions comparing gPENELOPE and MRIdian's KMC .....	62
Table 4.1 Performance benchmarks in a homogeneous water phantom.....	79
Table 4.2 Performance benchmarks in a water-lung-water phantom .....	81
Table 4.3 Performance benchmarks in a water-lung-tumor-water phantom .....	84
Table 4.4 Gamma passing rates and standard deviations of relative differences for 15 clinical IMRT plans .....	85
Table 4.5 Performance benchmarks in real patient phantoms .....	88
Table 5.1 Performance comparison between gDPMvr and gDVH.....	100

# Acknowledgments

Life is full of surprises, so is my Ph.D. study in Washington University in St. Louis. Planned to dedicate my time to condensed matter researches in Prof Zohar Nussinov's group, I actually end up studying Monte Carlo simulations for radiation therapy supervised by Prof Harold Li in the medical school. I'm honored that they're the co-chairs of my dissertation defense committee and both taught me a lot of research skills. I'm also the luckiest man to meet and fall in love with my wife in the medical school during my Ph.D. study. I'd appreciate WUSTL so much for offering me the opportunity to pursue a Ph.D. degree in physics, make bunch of good friends, accomplish meaningful projects, and meet my true love here.

Before joining Prof Harold's group, I had been working with Prof Zohar on "coagulation process of escaping bubbles and parallel simulation" for one year. He is an extremely nice person who would explain ideas and offer suggestions patiently and politely. Though this project did end up in a published paper as I moved to Harold's group due to funding issue, I learned plenty of research skills that benefit me a lot in the long term, such as CUDA programming, fitting analysis, scientific writing, etc. He is the person that introduced me to the world of scientific research, and I appreciate so much for his continuous support throughout my whole Ph.D. study. In addition, I would thank Bo Sun, graduate student in his group, for offering me many research advices and referring me to a job position in his company.

I would pay Prof Harold Li my highest respect for his forward-thinking guidance and generous help to my Ph.D. researches. He is a person always full of passion for innovative technologies, so he keeps reading and forwarding me the latest papers, and wants me to know the tendency of hot researches as well. He's a brilliant person that often pops up innovative ideas, and eager to share with me even though it's off-work time. He's also a person emphasizing

details and accumulation, so he requires me to keep records of everything related to research for later reference. Now I would thank his requirement as these records helped me finish this dissertation very quickly. He's a nice person never pushing me for a project deadline. It's his trust that pushing me working hard and productively.

Every member in Harold's group makes my Ph.D. life brighter. It's my honor to spend quality time in the office and lab with you guys. I own special thanks to Thomas Mazur, who offered me a lot of help in several cooperated projects, and made dedicated effects to correcting the language errors in my papers. I'm also grateful that he put me as co-author in two of his impactful papers.

I would thank Prof Deshan Yang for offering many technical advices for my researches. His help often saves me plenty of time. I'd also thank his students Shi Liu for sharing his MATLAB codes, and Yabo Fu for insightful career path advices.

Moreover, I would like to thank Professors Anders Carlsson, and Li Yang for sitting in my committee and for their insightful comments on my dissertation.

Finally, I would like to devote all my thanks to my family and friends who always support me for pursuing the Ph.D. degree in physics in WUSTL. I own special thanks to my wife, Jingxia Meng, for meeting me in such a beautiful place and becoming the biggest surprise in my Ph.D. life.

Yuhe Wang

*Washington University in St. Louis*

*December 2017*



Dedicated to my family.

## ABSTRACT OF THE DISSERTATION

Fast Monte Carlo Simulations for Quality Assurance in Radiation Therapy

by

Yuhe Wang

Doctor of Philosophy in Physics

Washington University in St. Louis, 2017

Dr. Zohar Nussinov, Associate Professor of Physics, Chair

Dr. Harold Li, Associate Professor of Radiation Oncology, Co-Chair

Monte Carlo (MC) simulation is generally considered to be the most accurate method for dose calculation in radiation therapy. However, it suffers from the low simulation efficiency (hours to days) and complex configuration, which impede its applications in clinical studies. The recent rise of MRI-guided radiation platform (e.g. ViewRay's MRIdian system) brings urgent need of fast MC algorithms because the introduced strong magnetic field may cause big errors to other algorithms. My dissertation focuses on resolving the conflict between accuracy and efficiency of MC simulations through 4 different approaches: (1) GPU parallel computation, (2) Transport mechanism simplification, (3) Variance reduction, (4) DVH constraint. Accordingly, we took several steps to thoroughly study the performance and accuracy influence of these methods. As a result, three Monte Carlo simulation packages named gPENELOPE, gDPMvr and gDVH were developed for subtle balance between performance and accuracy in different application scenarios. For example, the most accurate gPENELOPE is usually used as "golden standard" for radiation meter model, while the fastest gDVH is usually used for quick in-patient dose calculation, which significantly reduces the calculation time from 5 hours to 1.2 minutes (250

times faster) with only 1% error introduced. In addition, a cross-platform GUI integrating simulation kernels and 3D visualization was developed to make the toolkit more user-friendly. After the fast MC infrastructure was established, we successfully applied it to four radiotherapy scenarios: (1) Validate the vendor provided Co60 radiation head model by comparing the dose calculated by gPENELOPE to experiment data; (2) Quantitatively study the effect of magnetic field to dose distribution and proposed a strategy to improve treatment planning efficiency; (3) Evaluate the accuracy of the build-in MC algorithm of MRIdian's treatment planning system. (4) Perform quick quality assurance (QA) for the "online adaptive radiation therapy" that doesn't permit enough time to perform experiment QA. Many other time-sensitive applications (e.g. motion dose accumulation) will also benefit a lot from our fast MC infrastructure.

# Chapter 1: Introduction

The term “Monte Carlo method” has been well known since the 1940s when scientists working on the nuclear-weapon project began to use it to simulate how particles transport and interact in such a beyond-experiment scenario. Nowadays, Monte Carlo method has been greatly advanced since then and widely used to solve complex problems involving multiple independent variables where conventional “deterministic method” would cost formidable memory and computational time. Its application to radiation therapy was first introduced in the 1950s [1, 2], but it has not been as widely used as its competitor, “convolution-superposition method” [3], even though Monte Carlo method is well acknowledged as the most accurate approach [4, 5] for radiation dose calculation. The key obstacle turns out to be the low computational efficiency of conventional Monte Carlo packages. The recent rise of MRI-guided radiation platform (e.g. ViewRay’s MRIdian system [6]) brings urgent need of fast Monte Carlo algorithms because the introduced strong magnetic field may cause significant errors to other algorithms. Therefore, a large percent of this dissertation will focus on improving the efficiency of Monte Carlo dose calculation via four approaches, i.e. (1) GPU parallel computation, (2) Transport simplification, (3) Variance reduction, and (4) DVH-constraint. The accelerated calculation will not only benefit the existing applications, but also enable new researches in time-sensitive scenarios (e.g. motional dose accumulation).

This chapter contains a general introduction to the history of Monte Carlo method and its application to the quality assurance (QA) procedure in radiation therapy. In addition, it covers some technical features of our target platform, MRIdian system by ViewRay, and explains the

motivations of the above four accelerating methods. Finally, it provides a structure overview of this dissertation so readers know what to expect in the rest chapters.

## **1.1 History of Monte Carlo simulation**

The idea of using randomness to solve deterministic problems can be traced back at least to the eighteenth century, to Georges Louis LeClerc, Comte de Buffon (1707-1788), an influential French scientist [7]. He created a famous study method called “Buffon’s needle”, which uses repeated needle tosses onto a lined background to estimate  $\pi$ . He proved the probability that each time a needle would intersect a line was  $2/\pi$ . He tested his theory by throwing baguettes over his shoulder on to a tile floor. The needle or the baguette had to be the same length as the distance between the lines.

In the 19th and early 20th centuries, simulation was increasingly used as an experimental means of confirming theory, analyzing data, or supplementing intuition in mathematical statistics. However, it is significantly different from typical modern Monte Carlo simulations. The early simulations dealt with previously understood deterministic problems. Modern simulation inverts the process, treating deterministic problems by first finding a probabilistic analog and solving the problem probabilistically.

This form of simulation was first developed and used systematically during the Manhattan Project(1940s), the American World War II effort to develop nuclear weapons. It was applied to investigate radiation shielding and the distance that neutrons would likely travel through various materials. It was named after the Monte Carlo Casino in Monaco due to its probabilistic nature.

Monte Carlo simulation is now a widely-used scientific tool for problems that are analytically intractable and for which experimentation is too time-consuming, costly, or

impractical. Researchers explore complex systems, examine quantities that are hidden in experiments, and easily repeat or modify experiments. However, Monte Carlo simulation also has its own disadvantages: it can require huge computing resources; it doesn't give exact solutions; results are only as good as the model and inputs used; and simulation software, like any software, is prone to bugs.

The first attempt to employing Monte Carlo methods in radiation dose calculation dates to 1950s when Robert R. Wilson published his method using a “spinning wheel of chance” [1, 2]. Although apparently quite tedious, Wilson’s method was still an improvement over the analytic methods of the time—particularly in studying the average behavior and fluctuations about the average. The first use of an electronic computer in simulating high-energy cascades by Monte Carlo methods was reported by Butcher and Messel [8] in 1960. For the last 60 years, several outstanding Monte Carlo packages (e.g. EGSnrc [9], MCNP [10], GEANT4 [11], PENELOPE [12], etc.) for photon-electron transport were developed and tested with cooperation of many research institutions.

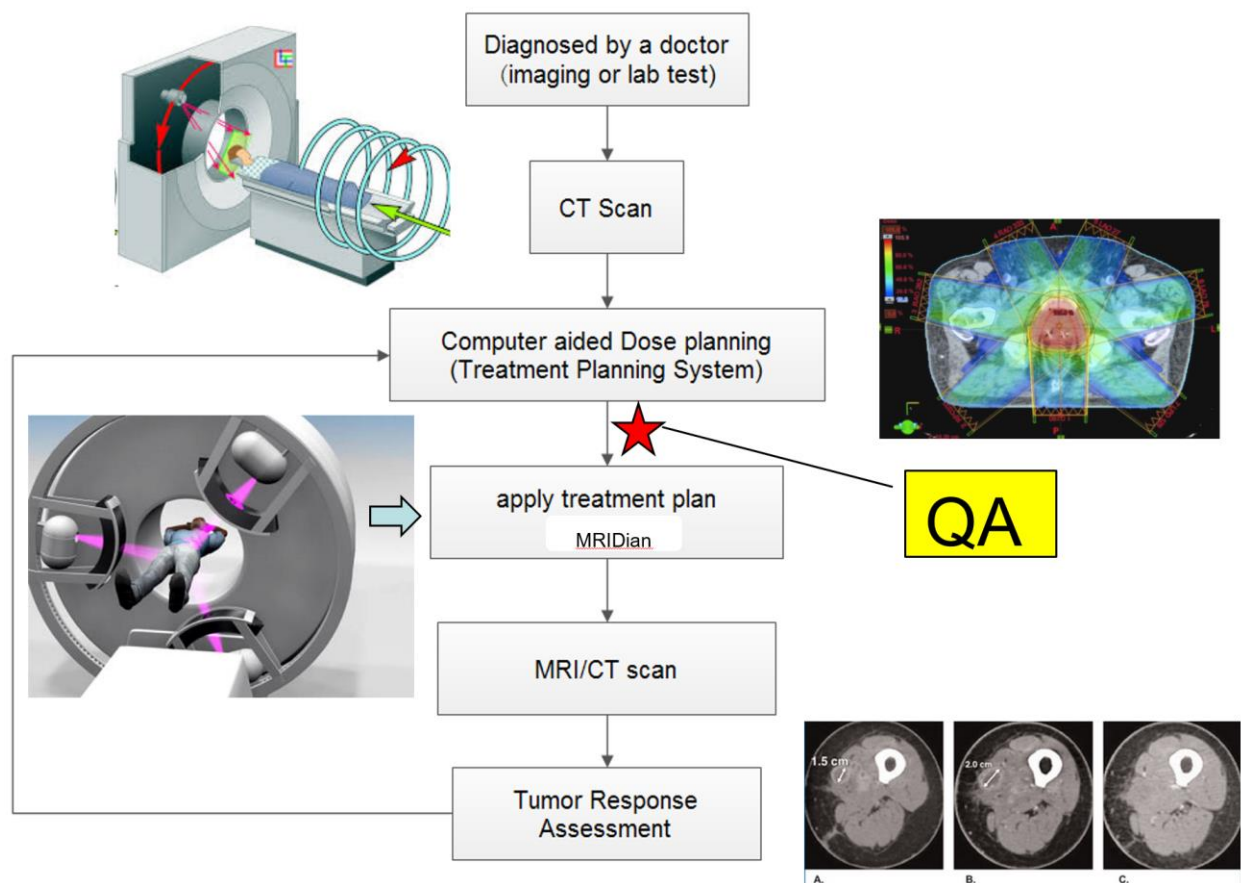
The theory of Monte Carlo transport became very mature in 1990s and hasn’t changed much since then. The simulation efficiency heavily relies on the improvement of modern CPU’s speed. People often use large CPU clusters to achieve desired simulation speed. The CPU clusters, however, are very not easily affordable in the clinical environment. In 2010s, the appearance of general-purpose programming language on GPU greatly lowered the computational cost of parallel Monte Carlo simulation, and enabled more clinical applications. Nowadays, clinically feasible Monte Carlo simulations are drawing more attentions since the recent rise of MRI guided radiation therapy introduces a strong magnetic field which may cause significant errors to other algorithms.

## 1.2 Workflow of radiation therapy

**Figure 1.1** shows the simplified workflow of current radiation therapy. When a patient is diagnosed with cancer tumor and is recommended for radiation therapy, the therapist will first take a Computed Tomography (CT) scan of the patient to generate 3-dimensional density information called “phantom”. The phantom is then dissected and marked at different regions. The therapist may also take MRI scan of the patient to assist the dissection as MRI has much better tissue contrast. These information is imported into the so called “Treatment Planning System” (TPS) to generate an appropriate treatment plan that optimizes the radiation dose distribution within the tumor by properly configuring the angles and shapes of multiple photon/electron beam. Before the treatment plan can be delivered to patient, we perform a procedure called “Quality Assurance” (QA) to ensure the plan works as we expect through a set of radiation experiments. Otherwise, any error could cause severe damage to the patients’ normal tissue. After QA, the therapist puts the patient into a treatment machine to deliver the radiation according to the plan. This machine is usually a conventional “linear accelerator” (LINAC) or a novel MRI-guided Co60 radiation machine called MRIdian (produced by ViewRay). After certain period, the therapist will take another MRI/CT scan to evaluate the tumor response to the treatment. These new images are imported into TPS to adjust plan for next treatment fraction. We repeat this workflow loop to fight against the cancer.

The experimental QA procedure, however, has several limitations. First, the measurement devices are not ideal for QA purpose. **Figure 1.2** shows the three typical devices to measure dose distributions in point, 2D and 3D. The ion-chamber has high accuracy but its volume is usually bigger than a voxel in TPS, and its output shows slightly dependence on the magnetic fields’ orientation. The radiographic film has very high spatial resolution, but the Ag+ material inside

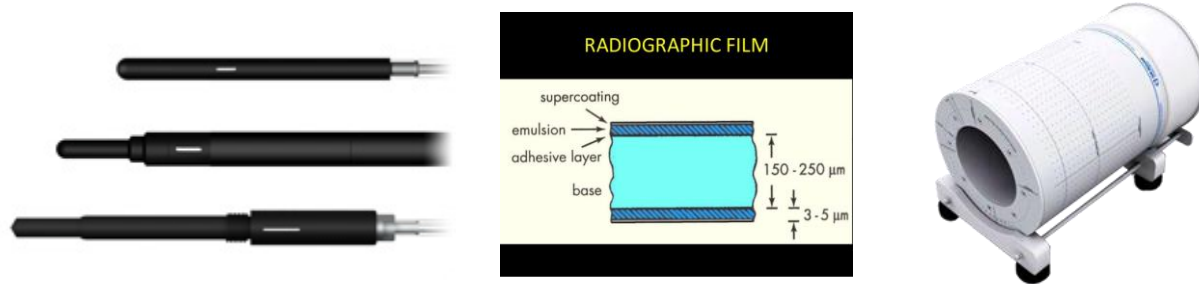
will cause low-energy dose overresponse problem. The arch-check is made of diodes array which is configured at low spatial resolution, and the nature of diodes will result in severe orientation dependence and low accuracy. Second, the phantom for radiation experiment cannot be patient specific as building a specific dummy for each patient is impractical. Currently, we use unified phantoms radiated by patient specific treatment plans. The measurements are then compared to the results given by TPS. Third, the measurements cannot cover the all the points. Even the arch-arch is limited to a 2D surface curved in 3D space. Fourth, performing experiments takes much longer time than running a Monte Carlo simulation.



**Figure 1.1** Simplified workflow of modern radiation therapy.



Nowadays, online adaptive radiation therapy (ART) is being popularized in clinical practice as it takes tumor's response into consideration but doesn't cost much time. It acquires patient's new phantom, adjust treatment plan, perform QA, and deliver radiation dose while keeping the patient onboard since the second treatment fraction. The whole preparation time before delivery has a narrow window of about 26 minutes. The QA procedure via experiment cannot be completed in such a brief time, so it requires a fast Monte Carlo simulation to finish the task.



**Figure 1.2** Devices to measure dose: (left) ion-chamber, point dose; (middle) radiographic film, 2D dose; (right) arch-check, surface dose in 3D.

### 1.3 MRgRT: MRIdian platform

The MRIdian by ViewRay [6] is an innovative platform that combines the real-time MR imaging and radiation delivery. As shown in figure 1.1, the MRIdian system integrates a 0.35 T whole-body MR imaging system into an RT delivery system consisting of a rotating gantry with three Co60 heads spaced  $120^\circ$  apart that can provide a maximum combined dose rate of 550 cGy/min at the isocenter. Its MR imaging system can generate 5 frames of planar MR image per second, and provides near real time delivery guidance. Once the imaging system detects the tumor is far from the beam center due to respiration or gastrointestinal motility, it will inform the delivery system to shut down the beam temporarily and wait until the tumor moves back to the beam center.

This mechanism greatly improves the delivery error and reduces the damage to patients' normal tissue. However, it also increases the treatment time as the tumor could be off the beam center for a large portion of total time if a strict “gating” criteria is applied. Therefore, ViewRay is developing next generation of MRIdian by replacing the Co60 head by compact LINAC that can provide much higher radiation flux rate.

## **1.4 Ways to improve efficiency**

Monte Carlo radiation transport simulation is generally considered to be the most accurate method for dose calculation in radiation therapy [4, 5]. Well-known Monte Carlo packages such as MCNP [10] Geant4 [11] EGS4/EGSnrc [9, 13] and PENELOPE [12, 14] have been demonstrated to agree excellently with experimental data under a wide range of conditions. For example, EGSnrc was shown to pass the Fano cavity test at the 0.1% level [5]. Here, we categorize these platforms as “accuracy-oriented”. While these packages are highly accurate, they typically require long simulation time to finish a sufficient number of histories in order to achieve adequate statistical uncertainty.

Three general approaches have been considered for accelerating Monte Carlo calculations: (1) simplifying particle transport mechanisms, thus reducing the necessary time for each particle history, (2) using variance reduction techniques such as particle splitting, Russian roulette, and interaction forcing to reduce the total history number required to achieve a given uncertainty and (3) enhancing the computational capability by parallelizing the simulation on multiple CPU or GPU threads [15]. Packages like VMC [16-18] and DPM [19] applied approaches (1) and (2) to achieve clinically desired speeds, but sacrificed generality and absolute accuracy by dropping simulation of positrons and using simpler cross-section profiles, among other simplifications. gDPM [20, 21] further utilized approach (3) (i.e. GPU parallelism) to obtain higher efficiency

compared to the original DPM, while GPUMCD [22] performed similar simplification to DPM and was directly oriented to GPU implementation. GMC [23] was developed based on Geant4 but it results in larger discrepancy from Geant4 than expected (2%/2 mm gamma passing rate is 91.74% for IMRT plans). Accuracy was possibly compromised for GMC by the fact that Geant4 uses a lot of virtual functions and class inheritances that make implementing a faithful adaptation from C++ to CUDA difficult. Here, we categorize these implementations as “efficiency-oriented”.

Another approach may be applied to further accelerate the QA procedure. We can perform detailed simulation in the important target regions, and only do rough estimation in the rest voxels. As the important regions used to calculate “Dose Volume Histograms” (DVHs) only take a small proportion of the whole volume, the simulation time can be significantly reduced. We named this patient QA specific method as “DVH constraint”.

## **1.5 Overview of the dissertation**

Chapter 2 covers some basic knowledge of Monte Carlo simulations regarding to both photon and electron transport. Chapter 3 discusses how to build the GPU accelerated Monte Carlo dose calculation package gPENELOPE. It starts with the introduction of the GPU architecture and CUDA programming language. It then introduces the optimized workflow for GPU implantation and the method to validate gPENELOPE. Finally, it discusses three applications of gPENELOPE in radiation therapy. Chapter 4 further applied transport simplification and variance reduction to build a faster code gDPMvr. It includes thorough benchmarks to study the performance and accuracy influence of these methods. Chapter 5 introduces "DVH constraint" to further accelerate the dose calculation for DVH-oriented QA. Chapter 6 provides three powerful geometry modules to handle practical applications. Chapter 7 gives a brief glimpse of two

graphic user interfaces aiming to lower the level of difficulty for average users. Chapter 8, in the end, summarize the results of we have achieved and discusses potential developments of fast Monte Carlo simulation in the future.

# Chapter 2: Monte Carlo basics

This chapter gives a basic introduction of Monte Carlo simulation. We will discuss random-sampling methods, transport models, and various photon and electron interactions.

## 2.1 Random-sampling methods

The most essential part of Monte Carlo simulation is to obtain a large number of random variables obeying given probability distribution functions (PDF). This process is called “random-sampling”. For the sampling of continuous variables, we usually start from a group variables linearly distributed in the interval  $(0,1)$ . Therefore, a good “pseudorandom number generator” (PRNG) is worth some discussion. Then Inverse-transform method and rejection method are introduced to perform the sampling of continuous variables. For discrete distributions, we will introduce “summation method” for simple cases and “Walkers’ aliasing method” for optimized performance.

### 2.1.1 Pseudo random number generator

The model of a pseudo random number generator is that given an initial value, we can get a large sequence of numbers that behaves like random numbers. The sequence is allowed to repeat after a large period. Usually the larger period, the better statistical properties. Although the sequences that are closer to truly random can be generated using hardware random number generators, pseudorandom number generators are more practical with respect to their speed in number generation and their reproducibility.

#### Linear congruential generator

The easiest and fastest PRNG among the applicable ones is so-called linear congruential generator, which is defined by the following recurrence relation:

$$X_n = (aX_{n-1} + c) \bmod m, \quad \xi_n = X_n/m \quad (2.1)$$

where  $\{X\}$  is the integer sequence of pseudo random values,  $a$ ,  $c$  and  $m$  are carefully chosen integer constant, and  $\{\xi\}$  is the desired random number sequence uniformly distributed in  $(0,1)$ . One good instance of the constant  $a$ ,  $c$  and  $m$  would be  $7^5$ ,  $0$ , and  $2^{31} - 1$ . It's exactly the famous 16807 ( $= 7^5$ ) PRNG [24], which has a recurrence period of the order of  $10^9$ . However, the value is far too short considering that modern Monte Carlo simulations of radiation therapy may require as many as  $10^9$  tracks to reach desired uncertainty. A more sophisticated congruential generator was proposed by L'Ecuyer [25] (1988) as follow:

```
class PRNG {
    int32_t iseed1, iseed2;

    void init(int32_t s1, int32_t s2) {
        iseed1 = s1;
        iseed2 = s2;
    }
    double operator () () {
        int32_t i1 = iseed1/53668;
        iseed1 = 40014*(iseed1 - i1*53668) - i1*12211;
        if (iseed1 < 0) iseed1 += 21474835663;
        int32_t i2 = iseed2/52774;
        iseed2 = 40692*(iseed2 - i2*52774) - i2*3791;
        if (iseed2 < 0) iseed2 += 2147483399;

        int32_t iz = iseed1 - iseed2;
        if (iz < 1) iz += 2147483562;
        return iz/2.147483563e9;
    }
};
```

(2.2)

This C++ implementation requires two numbers to initiate the PRNG, and its sequence period is of the order of  $10^{18}$ , which is virtually inexhaustible in a simulation costing  $10^9$  tracks. The

famous Monte Carlo package PENELOPE chose this method to generate random number sequences.

### **Mersenne Twister**

Though the last version linear congruential generator has a very easy implementation and a long  $10^{18}$  period, its statistical property is not as good as the famous Mersenne twister [26] method, which even has a huge  $2^{19937} - 1$  recurrence period. For more precise simulation (e.g. the radiation source model), Mersenne twister generator seems to be a better choice. The C++ 11 standard library has included a template of Mersenne twister generator.

### **PRNG on GPU: cuRand**

As GPU acceleration will be utilized throughout the four optimization stages in this dissertation, it's necessary to have a close look at the random number library provided by NVIDIA's CUDA development toolkit. We need to wrap two functions i.e. **curand\_init()** and **curand\_uniform()** and maintain a status struct **curandState** in C++ as shown in code block (1.2). The function **curand\_init()** guarantee that PRNGs initialized with different seeds will be uncorrelated and **curand\_uniform()** implemented a XORWOW generator [27] optimized for GPU. Compared to the implementation in code block (1.2), GRNG is a little slower due to larger status struct (means longer memory access latency), but has better statistical properties. We will adapt this method when we move to the second optimization stage – transport simplification – to compensate possible accuracy losses.

```

class GRNG {
public:
    __device__ void init(int seed) {
        curand_init(1234, seed, 0, &state);
    }
    __device__ __forceinline__ ZFloat operator () () {
#ifdef USE_SINGLE_PRECISION
        return curand_uniform(&state);
#else
        return curand_uniform_double(&state);
#endif
    }
private:
    curandState state;
};

```

(2.3)

### 2.1.2 Continuous variable: inverse-transform sampling

Given a continuous random variable  $x$ , the cumulative distribution function of  $x$  is defined as

$$\Phi(x) = \int_{x_{min}}^x p(x') dx' \quad (2.4)$$

which is a non-decreasing function of  $x$ , and then it has an inverse function  $\Phi^{-1}(\xi)$ . The transform  $\xi = \Phi(x)$  maps  $x$  to a new random variable  $\xi \in (0,1)$ . The PDF of  $\xi$ ,  $p_\xi(\xi)$ , must hold the relationship with  $p(x)$  due to the conservation of probability:

$$p_\xi(\xi) d\xi = p(x) dx \quad (2.5)$$

Therefore, we'll have

$$p_\xi(\xi) = p(x) / \left( \frac{d\xi}{dx} \right) = p(x) / \left( \frac{d\Phi(x)}{dx} \right) = p(x) / p(x) = 1 \quad (2.6)$$

It's clear that the new variable  $\xi$  distributes uniformly between 0 and 1. On the contrary, if we are given a set of uniformly distributed random number  $\xi \in (0,1)$  (via PRNG introduced in



section **2.1.1**), the random number  $x = \Phi^{-1}(\xi)$  will obey the desired PDF  $p(x)$ , and accomplish the sampling procedure.

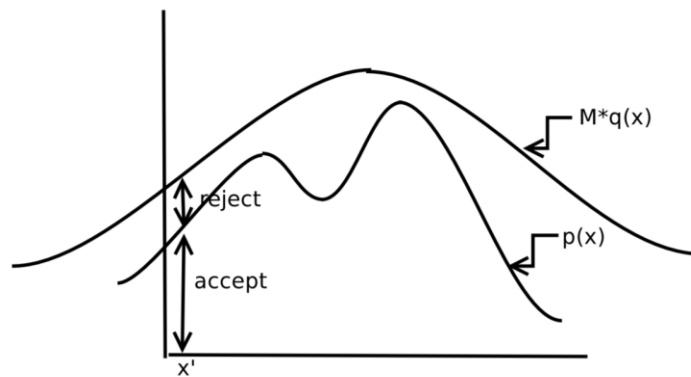
### 2.1.3 Continuous variable: rejection sampling

The inverse-transform sampling introduced in section **2.1.2** is simple and efficient. However, it's only applicable when  $p(x)$  can be integrated in analytical form and  $\Phi^{-1}(\xi)$  is easy to resolve.

Most practical PDFs cannot fulfil the above requirement so another more general method called rejection sampling is created to address the drawback.

Suppose we have an analytically non-integral PDF  $p(x)$  as shown in **Figure 2.1**. The key point is to find another analytically integral PDF  $q(x)$  and an appropriate number  $M$  so that  $M \cdot q(x) > p(x)$  validates for all  $x$ . The sampling of each random variable  $x$  will work as follow:

- (1) Using inverse-transform method to sample one random number  $x \sim p(x)$
- (2) Sample one uniformly distributed random variable  $\xi \sim U(0,1)$
- (3) If  $\xi < \frac{p(x)}{M \cdot q(x)}$ , accept  $x$ ; else reject  $x$ , and go to step (1).



**Figure 2.1** Illustration of rejection sampling

To ensure the sampling process has a high efficiency, we need to choose  $M$  and  $q(x)$  properly so that the area covered by  $M \cdot q(x)$  is as close as possible to the area covered by  $p(x)$ .

### 2.1.4 Discrete variable: reverse-transform sampling

We can treat discrete distribution as the general continuous distribution multiple Dirac delta functions as equation (1.7).

$$p(x) = \sum_{i=1}^N p_i \delta(x - i) \quad (2.7)$$

The corresponding accumulative distribution function is

$$\Phi(x) = \sum_{i=1}^{[x]} p_i \quad (2.8)$$

where  $[x]$  means the integer part of  $x$ . The inverse function of (1.8) leads to the following sampling formula:

$$x = j \text{ if } \Phi_{i-1} < \xi < \Phi_i \quad (2.9)$$

Where  $\Phi_i = \sum_{i=1}^j p_i$  means the discrete accumulated probability. If the number  $N$  of the  $x$  values is large, we can use binary search to reduce the searching complexity from  $O(N)$  to  $\log(N)$ . In real Monte Carlo simulations, however, the discrete sampling happens so frequently that we'll turn to Walker's aliasing method introduced in next section.

### 2.1.5 Discrete variable: Walker's aliasing method

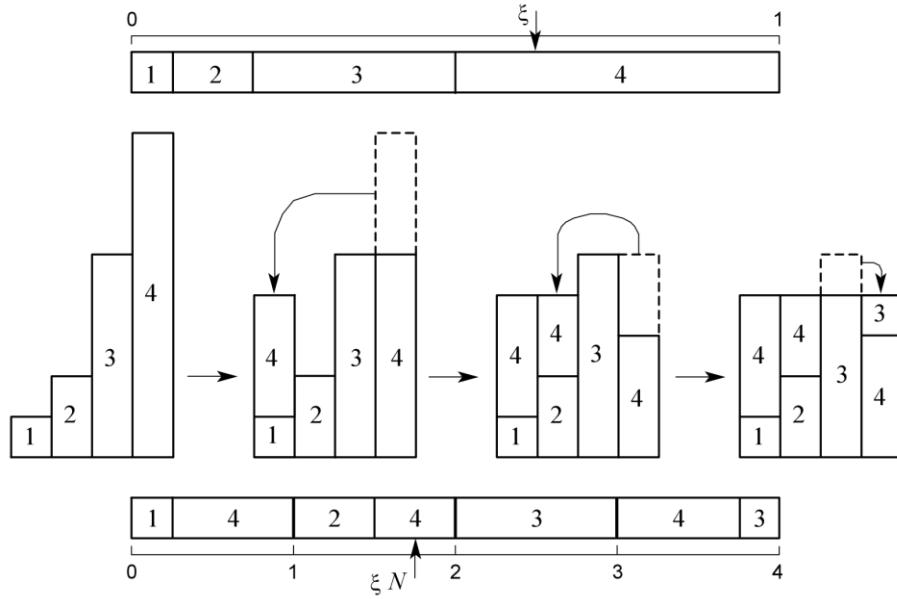
The idea underlying Walker's method [28] can be easily understood by graphical arguments [29].

Let us represent the discrete PDF as a histogram constructed with  $N$  bars of width  $1/N$  and heights  $Np_i$ . The histogram then can be cut off at convenient heights and the resulting pieces can be arranged in such a way that each vertical line crosses at most two different pieces (**Figure 2.2**). This struct can be setup by selecting the lowest and the highest bars in the histogram, say the  $l$ th and  $j$ th, respectively, and by cutting the highest bar off to complete the lowest one., which

subsequently is kept unaltered. We label the added piece with the “alias” value  $K_l = j$ , and introduce the cutoff value  $F_l$  defined as the height of the lower piece in the  $l$ th bar. This lower piece keeps the label 1. Iteration of this process eventually leads to the complete square after  $N-1$  steps. Note that the probability  $p_i$  can be reconstructed from alias and cutoff values:

$$Np_i = F_i + \sum_{j \neq i} (1 - F_j) \delta(i, K_j) \quad (2.10)$$

Walker’s sampling method works as follows: We sample two independent random numbers  $\xi_1$  and  $\xi_2$ , and define the random point  $(\xi_1, \xi_2)$ , which is uniformly distributed in the square. If  $(\xi_1, \xi_2)$  lies over a piece labelled with the index  $i$ , we take  $x = i$  as the selected value.



**Figure 2.2** Graphical representation of the inverse transform method (top) and Walker's aliasing method (bottom) for random sampling from a discrete distribution. In this example, the random variable can take the values  $i = 1, 2, 3$  and  $4$  with relative probabilities

The sampling process can be optimized to required only one random variable: (1) Generate a random number  $\xi$  and set  $R = \xi N + 1$ , (2) Set  $i = [R]$  and  $r = R - I$ , (3) If  $r > F_i$ , deliver  $x = K_i$ , (4) Else deliver  $x = i$ . This method only needs one random number and one comparison at the cost of doubling the storage compared to the reverse transform method. The sampling performance is quite satisfactory. However, the calculation of alias and cutoff values is fairly involved and this limits its application to distributions that remain constant during the simulation.

## 2.2 Radiation transport model

In this section, we describe the model for Monte Carol simulation of radiation transport.

### 2.2.1 Scattering model

Let us limit our considerations to homogeneous random scattering media, where the molecules are distributed at the random with uniform density. The number of molecules per unit volume can be easily derived as

$$N = N_A \frac{\rho}{A_M} \quad (2.11)$$

where  $N_A$  is Avogadro's number and  $\rho$  is the mass density of the material. In each interaction, the particle may lose energy  $W$  and change its direction of momentum (described by polar scattering angle  $\theta$  and azimuthal scattering angle  $\phi$ ). For simplicity, we assume that the particle interacts with the medium through two independent mechanisms "A" and "B". The scattering model is then completely specified by the molecular differential cross section (DCS)

$$\frac{d^2\sigma_{A,B}}{dWd\Omega}(E; W, \theta) \quad (2.12)$$

where  $d\Omega$  is a solid angle element in the direction  $(\theta, \phi)$ . Note that the DCSs explicitly depend on the particle energy  $E$ . Since the molecules in the medium are oriented at random, the DCS is usually independent of the azimuthal angle. The total cross sections are

$$\sigma_{A,B}(E) = \int_0^E dW \int_0^\pi 2\pi \sin \theta d\theta \frac{d^2\sigma_{A,B}}{dW d\Omega}(E; W, \theta). \quad (2.13)$$

And the PDFs of the energy loss and the polar scattering angle are normalized to be

$$p_{A,B}(E; W, \theta) = \frac{2\pi \sin \theta}{\sigma_{A,B}(E)} \frac{d^2\sigma_{A,B}}{dW d\Omega}(E; W, \theta) \quad (2.14)$$

The azimuthal scattering angle is uniformly distributed in the interval  $(0, 2\pi)$ , i.e.  $p(\phi) = \frac{1}{2\pi}$ .

The total cross section for all kinds of interaction is

$$\sigma_T(E) = \sigma_A(E) + \sigma_B(E) \quad (2.15)$$

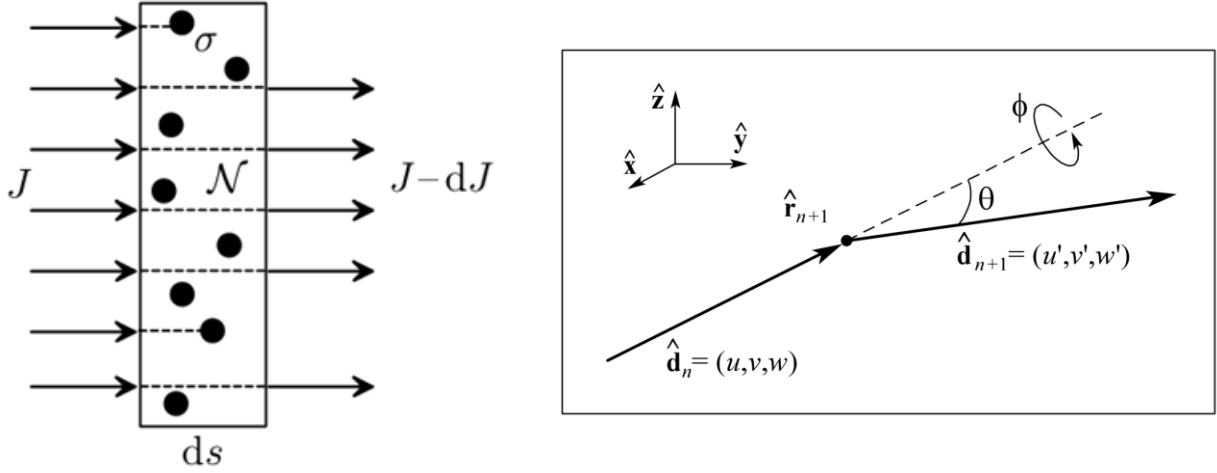
The probability of occurring that kind of interaction is given as

$$p_{A,B}(E) = \sigma_{A,B}(E)/\sigma_T(E) \quad (2.16)$$

We can use inverse-transform method to sample which kind of interaction to happen.

### 2.2.2 Random tracks

Each particle track starts off at a given position, with initial direction and energy in accordance with the characteristics of the radiation source. Each simulated track can be viewed as a series of states  $r_n, E_n, \hat{d}_n$ , where  $r_n$  is the position of the  $n$ -th scattering event and  $E_n, \hat{d}_n$  are the energy and direction cosines of the directions of the direction of movement just after that event.



**Figure 2.3** (left) The model for analyzing the probability of being scattered. (right) Angular deflection in single-scattering events.

To get an intuitive picture of the microscale scattering, we can imagine each molecule as a sphere distributed uniformly in the medium (**Figure 2.3** left). The particle will have the probability of being scattered proportional to the ratio of cross section area to total area it can see. Suppose the number of incident particles is  $J$ , the number of scattered particle  $dJ$  after passing a slab medium of cross section  $A$  and of width  $ds$  can be expressed as follow:

$$dJ = J \frac{\text{target area}}{\text{total area}} = J \frac{n\sigma_T}{A} = JN\sigma_T ds \quad (2.17)$$

where  $N \equiv \frac{n}{A ds}$  is number of molecules per unit volume defined in equation (2.11). This

differential equation yields an exponential decaying solution:

$$J(s) = J_0 e^{-N\sigma_T s} \quad (2.18)$$

this indicates that the PDF of free jumping distance  $s$  (without being scattered) follows a similar form:

$$p(s) = N\sigma_T e^{-N\sigma_T s} \quad (2.19)$$

The mean free path length then is evaluated as:

$$\lambda_T = \int_0^\infty p(s)s ds = \frac{1}{N\sigma_T} \quad (2.20)$$

From equation (2.15), we can deduce that

$$\lambda_T^{-1}(E) = \lambda_A^{-1}(E) + \lambda_B^{-1}(E) \quad (2.21)$$

The PDF of equation (2.19) can also be expressed as

$$p(s) = e^{-\frac{s}{\lambda_T}} / \lambda_T \quad (2.22)$$

Then we can easily apply inverse-transform method by solving  $p(s) = \xi$  to sample the free jumping distance  $s$

$$s = -\lambda_T \ln \xi \quad (2.23)$$

After the particle flies such a distance  $s$ , the interaction occurs at the position

$$r_{n+1} = r_n + s\hat{d}_n \quad (2.24)$$

The type of interaction (A or B) is selected via discrete inverse-transform method based on the probability given by equation (2.16). The energy loss  $W$  and polar scattering angle  $\theta$  are sampled from the PDF given in equation (2.12). The azimuthal scattering is generated trivially as  $\phi = 2\pi\xi$  according to the uniform distribution in  $(0, 2\pi)$ . After sampling the values of  $W$ ,  $\theta$  and  $\phi$ , the energy is reduced to  $E_{n+1} = E_n - W$ , and the new direction of momentum  $\hat{d}_{n+1}(u', v', w')$  is obtained by performing a rotation of  $\hat{d}_n(u, v, w)$  by polar scattering angle  $\theta$  and azimuthal angle  $\phi$  (**Figure 2.3** right). It can be derived the new cosine components can be expressed as

$$\begin{aligned}
u' &= u \cos \theta + \frac{\sin \theta}{\sqrt{1-w^2}} [uw \cos \phi - v \sin \phi], \\
v' &= v \cos \theta + \frac{\sin \theta}{\sqrt{1-w^2}} [vw \cos \phi + u \sin \phi], \\
w' &= w \cos \theta - \sqrt{1-w^2} \sin \theta \cos \phi.
\end{aligned} \tag{2.25}$$

These expressions are indeterminate when  $|w| \cong 1$ . In the case, we will simply set

$$u = \pm \sin \theta \cos \phi, \quad v = \pm \sin \theta \sin \phi, \quad w = \pm \cos \theta \tag{2.26}$$

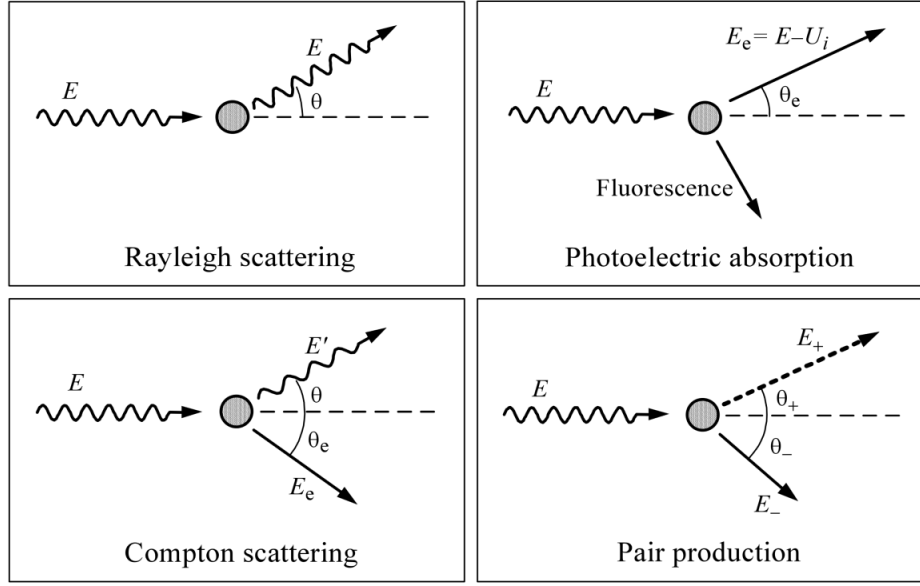
Note that equations (2.25) are not very stable numerically so it's necessary to renormalize  $\hat{d}_{n+1}$  periodically.

We will repeat this jump and knock steps to generate simulation tracks. A track is finished either when it leaves the material or when the energy goes below the pre-defined energy threshold  $E_{abs}$ , which is the energy where particles are assumed to be effectively absorbed locally in the medium

## 2.3 Photon interactions

In this section, we'll briefly discuss four kinds of interactions occurring in the simulation of photons, i.e. Rayleigh scattering, photoelectric absorption, Compton scattering, and electron-positron pair production (Figure 2.4). We will focus on the DCSs for different interactions and the efficient sampling algorithms. Other interactions, such as photonuclear absorption, occur with much smaller probability and can be ignored for most practical purposes [30].





**Figure 2.4** Four types of photon interactions with matter.

### 2.3.1 Rayleigh scattering

Rayleigh scattering is the process by which photons are scattered by bound atomic electrons without excitation of the target atom, i.e. the energies of the incident and scattered photons are the same. The DCS is given approximately by [31]

$$\frac{d\sigma_{Ra}}{d\Omega} = \frac{d\sigma_T}{d\Omega} [F(q, Z)]^2 \quad (2.27)$$

where

$$\frac{d\sigma_T(\theta)}{d\Omega} = r_e^2 \frac{1 + \cos^2 \theta}{2} \quad (2.28)$$

is the Thomson DCS for scattering by a free electron at rest,  $\theta$  is the polar scattering angle and  $F(q, Z)$  is the atomic factor. The quantity  $r_e$  is the classical electron radius and  $q$  is given by

$$q = 2(E/c) \sin(\theta/2) = (E/c) [2(1 - \cos \theta)]^{1/2} \quad (2.29)$$

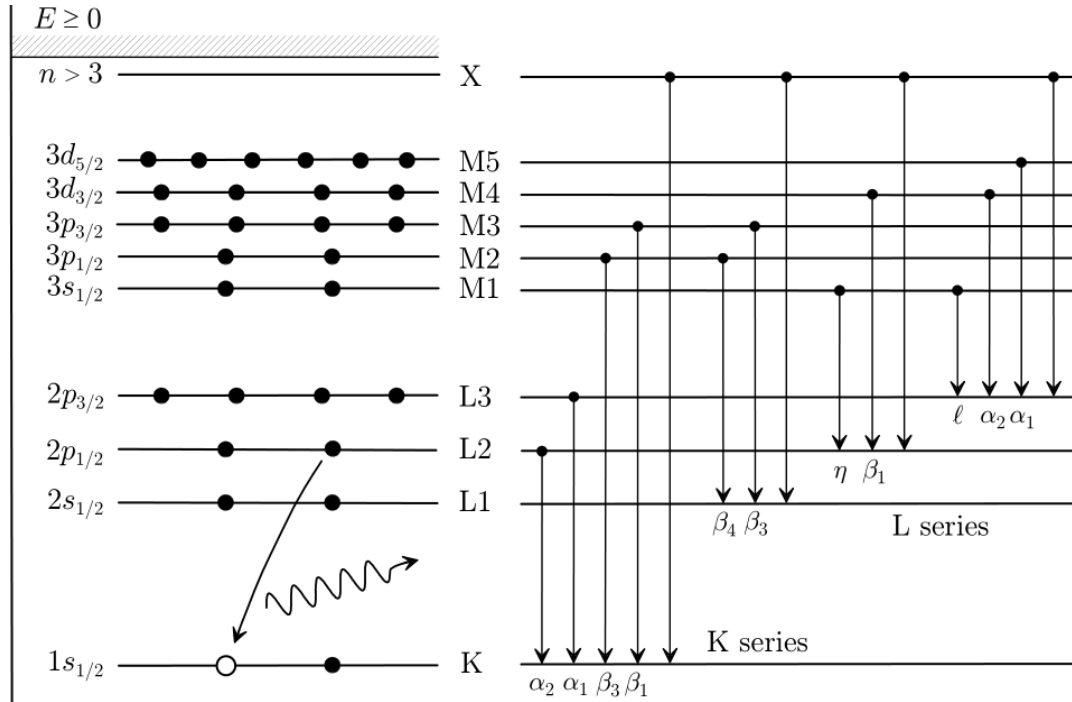
And  $F(q, Z)$  is given by

$$F(q, Z) = 4\pi \int_0^\infty \rho(r) \frac{\sin(qr/\hbar)}{qr/\hbar} r^2 dr \quad (2.30)$$

For such a complex distribution, we have to use rejection method to sample the deflection  $\cos\theta$ .

### 2.3.2 Photoelectric absorption

In the photoelectric effect, a photon of energy  $E$  is absorbed by the target atom, which makes a transition to an excited state. The photon beams found in radiation transport studies have relatively low photon densities and, as a consequence, only single photon absorption is observed. **Figure 2.5** shows the various notations for inner atomic electron shells and allowed radiative transitions between these shells.



**Figure 2.5** Various notations for inner atomic electron shells (left) and allowed radiative transitions (right) to these shells

The angular DCS is given by Sauter as

$$\frac{d\sigma_{\text{ph}}}{d\Omega_e} = \alpha^4 r_e^2 \left(\frac{Z}{\kappa}\right)^5 \frac{\beta^3}{\gamma} \frac{\sin^2 \theta_e}{(1 - \beta \cos \theta_e)^4} \left[ 1 + \frac{1}{2} \gamma(\gamma - 1)(\gamma - 2)(1 - \beta \cos \theta_e) \right] \quad (2.31)$$

where  $\alpha$  is the fine-structure constant,  $r_e$  is the classical electron radius and

$$\gamma = 1 + E_e/(m_e c^2), \quad \beta = \frac{\sqrt{E_e(E_e + 2m_e c^2)}}{E_e + m_e c^2} \quad (2.32)$$

We can define a new variable  $v = 1 - \cos \theta_e$  to simplify the PDF so rejection sampling can be applied.

### 2.3.3 Compton scattering

Compton scattering is the most important scattering event for photon interaction because in the energy range of radiation therapy, the cross-section of Compton scattering is largest among the four main interactions. The DCS for Compton interaction is given by Brusa in 1996 [32] as

$$\frac{d\sigma_{\text{Co}}}{d\Omega} \simeq \frac{r_e^2}{2} \left(\frac{E_C}{E}\right)^2 \left(\frac{E_C}{E} + \frac{E}{E_C} - \sin^2 \theta\right) S(E, \theta) \quad (2.33)$$

where  $r_e$  is the classical electron radius,  $E_c$  is defined as

$$E_c \equiv \frac{E}{1 + (1 - \cos \theta)E/(m_e c^2)} \quad (2.34)$$

and  $S(E, \theta)$  is defined as

$$S(E, \theta) = \sum_i f_i \Theta(E - U_i) n_i(p_{i,\text{max}}) \quad (2.35)$$

which represents atom binding effect. The final sampling equation can be expressed as

$$E' = E \frac{\tau}{1 - t\tau^2} \left[ (1 - t\tau \cos \theta) + \text{sign}(p_z) \sqrt{(1 - t\tau \cos \theta)^2 - (1 - t\tau^2)(1 - t)} \right] \quad (2.36)$$

where  $\tau \equiv 1/(1 + \kappa(1 - \cos\theta))$ , and  $t \equiv (p_z/m_e c)^2$ . Then the scattering angle and energy loss can be determined by conservation of momentum and energy:

$$\cos \theta_e = \frac{E - E' \cos \theta}{\sqrt{E^2 + E'^2 - 2EE' \cos \theta}}. \quad (2.36)$$

$$E_e = E - E' - U_i$$

### 2.3.4 Pair production

When the photon energy is high enough (greater than  $2m_e c^2$ ), it is possible for this photon to annihilate into an electron and a positron. This process is called pair production. The Nethe-Heitler DCS [33] derived from the Born approximation is

$$\frac{d\sigma_{pp}}{d\epsilon} = r_e^2 \alpha Z [Z + \eta] C_r \frac{2}{3} \left[ 2 \left( \frac{1}{2} - \epsilon \right)^2 \phi_1(\epsilon) + \phi_2(\epsilon) \right] \quad (2.37)$$

Similarly, we need to use rejection method to sample the distribution of  $\epsilon$ . The angular PDF is given [34, 35] as

$$p(\cos \theta_{\pm}) = a (1 - \beta_{\pm} \cos \theta_{\pm})^{-2} \quad (2.38)$$

where  $a$  is normalization constant and

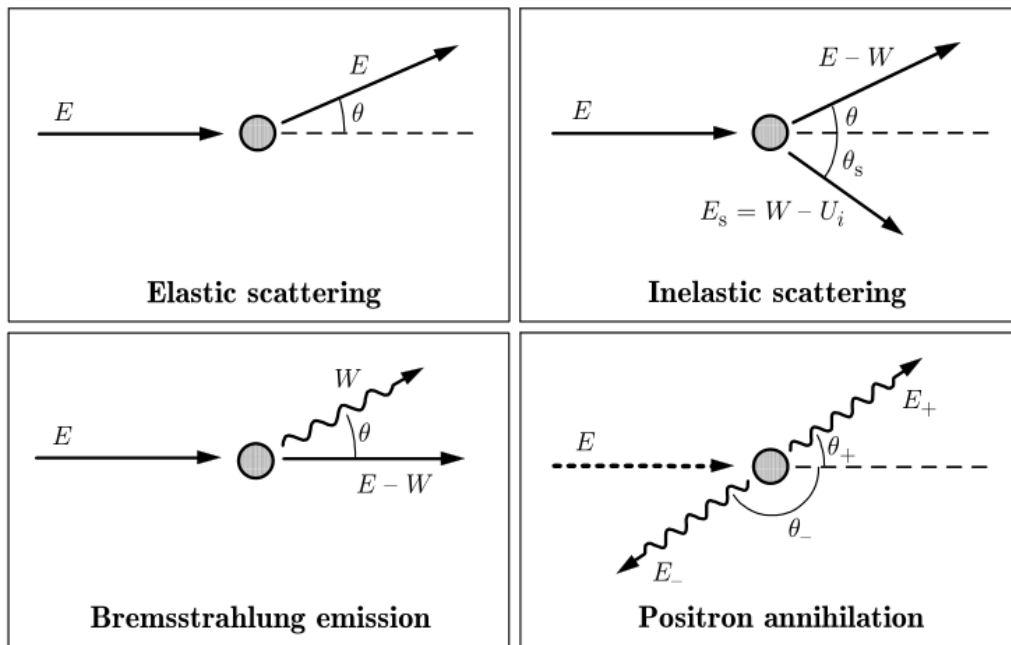
$$\beta_{\pm} = \frac{\sqrt{E_{\pm}(E_{\pm} + 2m_e c^2)}}{E_{\pm} + m_e c^2} \quad (2.39)$$

The sampling of deflection can be easily deduced via inverse-transform method as follow:

$$\cos \theta_{\pm} = \frac{2\xi - 1 + \beta_{\pm}}{(2\xi - 1)\beta_{\pm} + 1} \quad (2.40)$$

## 2.4 Electron/positron interactions

In this section, we'll briefly discuss four kinds of interactions occurring in the simulation of photons, i.e. elastic scattering, inelastic absorption, Bremsstrahlung emission, and positron annihilation (**Figure 2.6**). In principle, the simulation methods are similar to those of photon interactions [12]. However, the mean free path and energy loss between two interactions of a charged particle is much smaller than these of a photon. The charged particle may undergo order of  $10^6$  discrete interactions before coming to a complete stop. The number of photon interaction, on the other hand, is only order of  $10^2$  for one track. Therefore, it's impractical to simulate the charged particle in a detailed way as photon.



**Figure 2.6** Four types of electron interactions with matter

Fortunately, most of these interactions are elastic or semi-elastic, which means no energy or a tiny amount of energy is transferred to the surrounding material, and direction of movement is

only slightly scattered. This allows us to group many of those elastic/semi-elastic events into one condensed history (CH). This method was introduced by Berger in 1963 and is called CH technique ever since. Present-day CH implementations divide the interactions of charged particle into “hard” and “soft” events. The two interaction types are usually distinguished by a pre-defined energy loss threshold  $E_c$  and scattering angle threshold  $\Theta_c$ . That is, those losing energy less than  $E_c$  or being scattered with angle smaller than  $\Theta_c$  are called soft events, and the rest are called hard events. For hard events, we perform explicit detailed simulation like photon. For soft events, we group many continuous ones of them, and calculate the total energy loss based on “Continuous energy loss model”, the total scattering angle based on the “multiple scattering theory”, and the simulated path with “random hinge technique”. This strategy is called mixed simulation scheme [36].

### 2.4.1 Continuous energy loss

During a CH-step the charge particle continuously loss energy due to soft interactions. The average energy loss  $dE$  per CH step length  $ds$  at point  $\mathbf{r}$  is given by the restricted linear stopping power as follows:

$$L(\mathbf{r}, E, E_c, k_c) = -\frac{dE}{ds} = L_{col}(\mathbf{r}, E, E_c) + L_{rad}(\mathbf{r}, E, k_c) \quad (2.41)$$

where  $L_{col}(\mathbf{r}, E, E_c)$  and  $L_{rad}(\mathbf{r}, E, k_c)$  are restricted linear collision and radiation stopping powers. They can be calculated using the collision cross section and the bremsstrahlung production cross section by

$$\begin{aligned} L_{col}(\mathbf{r}, E, E_c) &= N(\mathbf{r}) \int_0^{E_c} dE' E' \sigma_{col}(\mathbf{r}, E, E') \\ L_{rad}(\mathbf{r}, E, k_c) &= N(\mathbf{r}) \int_0^{k_c} dk' k' \sigma_{rad}(\mathbf{r}, E, k') \end{aligned} \quad (2.42)$$

where  $N(\mathbf{r})$  is the number of scattering targets per unit volume at point  $\mathbf{r}$ . Note the stopping power integration is restricted to energies below  $E_c$  and  $k_c$ .

The step length  $s$  for an electron with initial energy  $E_0$  that loses energy  $\Delta E$  due in CH scheme can be calculated by

$$s = - \int_{E_0}^{E_1} \frac{dE}{L(\mathbf{r}, E, E_c, k_c)} = \int_{E_1}^{E_0} \frac{dE}{L(\mathbf{r}, E, E_c, k_c)} \quad (2.43)$$

where  $E_1 = E_0 - \Delta E$  is the electron energy at the end of the CH step. The function  $L(\mathbf{r}, E, E_c, k_c)$  should change accordingly if a boundary to a region with different material is crossed.

Obviously, the continuous energy loss model is an approximation in average sense. Analog Monte Carlo simulation shows that the real step end energies are random obeying a distribution with a mean value of  $E_1$ . This effect is known as energy straggling. In the mixed simulation scheme, the soft and hard energy straggling are handled in different approaches, where hard energy straggling is simulated explicitly with corresponding effects correctly taken into account, while soft energy straggling is simply neglected or simulated by an adequate straggling distribution function.

The strategy of ignoring soft energy straggling is valid and effective if we choose the parameter  $E_c$  and step size small enough. In this case, energy straggling is dominated by the explicitly modeled hard events with energy transfer larger than  $E_c$  and the soft energy fluctuations have a negligible impact on the simulation result.

### 2.4.2 Multiple Scattering

Different from the real curved path of charged particle, the simulated path is a simple straight line during one CH step (Figure 2.7). The combined effect of many small-angle soft collisions is simulated by sampling the angular deflection based on a dedicated multiple scattering theory.

We will introduce a simple Gaussian distribution developed by Fermi and Eyges [37]

$$p(\theta, \phi) d\theta d\phi = \frac{\theta}{\pi \overline{\theta^2}(s)} \exp\left(-\frac{\theta^2}{\overline{\theta^2}(s)}\right) d\theta d\phi \quad (2.44)$$

where  $\theta$  is the polar multiple scattering angle,  $\phi$  is the azimuthal scattering angle, and  $\overline{\theta^2}$  is the mean square deflection angle after step length  $s$ . This PDF leads to two separate cumulative probability functions

$$P(\theta) = 1 - \exp\left(-\frac{\theta^2}{\overline{\theta^2}(s)}\right), \quad P(\phi) = \frac{\phi}{2\pi} \quad (2.45)$$

The corresponding inverse-transform sampling forms are

$$\theta = \sqrt{-\overline{\theta^2}(s) \ln \xi_1}, \quad \phi = 2\pi \xi_2 \quad (2.46)$$

where  $\xi_1$  and  $\xi_2$  are uniform random numbers  $\in [0,1]$ . We need to enforce that  $\theta \leq \pi$  so we should reject the sampling result in equation (2.46) if  $\theta > \pi$ .

The quantity  $\overline{\theta^2}(s)$  can be calculated using the linear scattering power  $T_s(\mathbf{r}, E)$  at point  $\mathbf{r}$ .

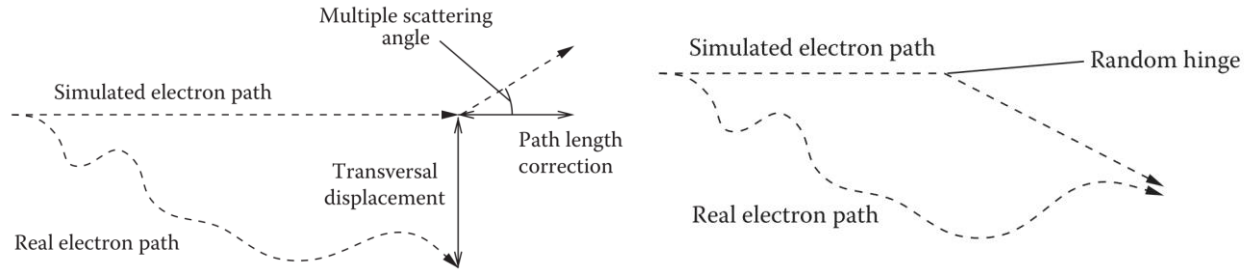
$$\overline{\theta^2}(s) = \int_0^s ds' T_s(s', E) \quad (2.47)$$

The Gaussian PDF in equation (2.44) can give a good approximation for small cumulative scattering angle  $\theta$ . Large scattering angles, however, are underestimated using this distribution.

A much more accurate multiple scattering theory is developed by Goudsmit and Saunderson [38]



to provide good approximation even for large scattering angle. All modern Monte Carlo simulation packages adapt this algorithm for accuracy concern. However, the calculation expense is much higher than the simple Gaussian PDF.



**Figure 2.7** (left) Real curved electron path vs simulated electron path as a straight line. The accumulated scattering angle is applied at the end of this CH step. (right) The scattering angle is applied at a random point along the path, i.e. through random hinge method.

### 2.4.3 Random hinge

As shown in **Figure 2.7** (left), the simplest multiple scattering scheme is to move the electron in a straight line until the final position and then apply the scattering of momentum. It's obvious that the electron range is over estimated since the real electron path is curved. In fact, the real electron range fluctuates around some mean value, and this effect is called range straggling. Note that range straggling is independent of the energy loss of electron so it should not be confused with the energy straggling discussed in last section.

To overcome the range overestimation, many Monte Carlo codes employ a path length correction (PLC) algorithm. Besides, a transverse displacement (TD) algorithm is included to handle the transverse fluctuations of the real electron end position relative to the lateral position as simulated during the CH step. A simple PLC and TD algorithm, called random hinge method, is demonstrated in **Figure 2.7** (right). This method divides the whole step length into two sub

steps  $\xi s$  and  $(1 - \xi)s$ , where  $\xi$  is a uniform random number  $\in (0,1)$ . The multiple scattering angle is applied when the electron has advance  $\xi s$  distance instead of at the end of current CH step. Though this method is very simple, it provides fairly good approximation to the real paths in an average sense and then is implemented in modern Monte Carlo packages such as PENELOPE [12] and XVMC [39]. Another similar random hinge approach determines the hinge point based on the energy interval instead of step length, and could achieve higher accuracy (implemented in DPM [19]).

The problems and algorithms discussed in this section could be neglected if the charged particle transport steps are limited to very small distance. However, the simulation becomes extremely inefficient in this way. We must carefully balance the max CH step length and those correction algorithms to achieve the maximum simulation efficiency.

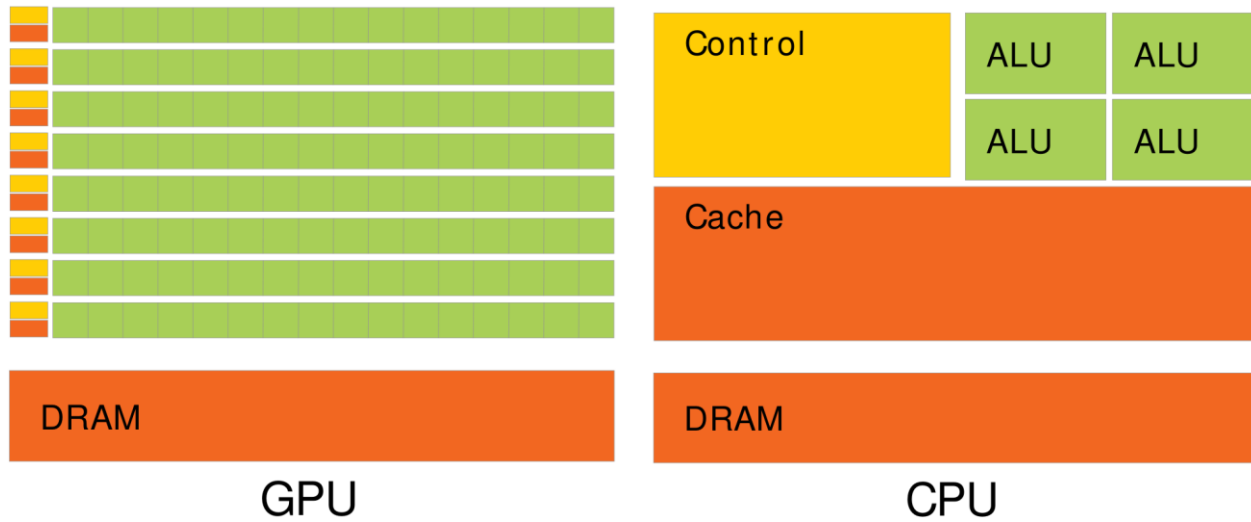
# Chapter 3: GPU acceleration

In chapter 1, we reviewed the recent development of MRI-guided radiation therapy, and the increasing demand of the capability of performing fast Monte Carlo simulations. We proposed four approaches to further increase the simulation efficiency. After reviewing all the essential components of Monte Carlo simulation in chapter 2, we will build our own fast MC codes from deploying GPU parallel computation first. We choose to start with GPU acceleration instead of other methods because GPU parallelization will not compromise the accuracy as long as the RNGs supplying the simulation are irrelevant with each other.

## 3.1 GPU architecture vs CPU architecture

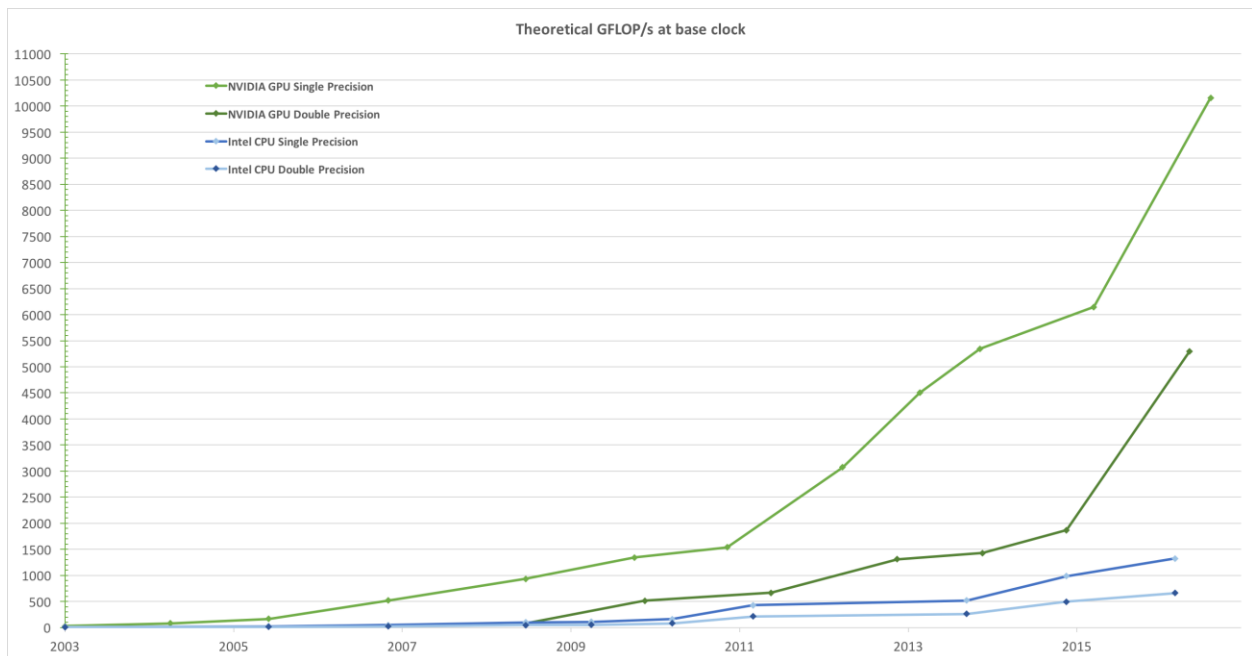
Graphic processing unit (GPU) was originally designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. It has numbers of arithmetic logic units (ALUs), super long instruction pipeline, simple controlling unit, and very limited amount of caches. Its memory access pattern is characterized by “single instruction multiple data” (SIMD), so it usually has very high memory throughput. Though the performance of each ALU in GPU is not impressive, the large “core” number leads to a great overall performance advantage compared to its component, i.e. central process unit (CPU).

Central processing unit (CPU) is the heart of modern computers and is designed for more general purposes. It doesn't have numbers of cores, but each core contains complex structures and can deliver very impressive performance. It has sophisticated controllers to reduce band latency via “branch prediction” and to reduce data latency via “data forwarding”. It also has multiple layers of cache to further reduce the data access latency.



**Figure 3.1** (left) The architecture of GPU (right) The architecture of CPU

Figure 3.1 summarizes the architectures of GPU and CPU. It's obvious that GPU contains numbers of ALUs, while CPU has sophisticated ALU, controllers and plenty of caches.



**Figure 3.2** Floating-Point Operations per Second for the CPU and GPU

Most popular Monte Carlo simulation packages were targeted to execute on CPU platform because it's easy to program and can achieve relative good performances. However, GPU can deliver much higher performance if all its ALUs can effectively utilized (see Figure 3.2). As the fast radiation dose calculation demands keep increasing nowadays, GPU acceleration becomes an attractive approach to improve the simulation performance, though it's more much more difficult to tune and optimize the codes on GPU than on CPU.

## 3.2 Introduction of CUDA

For programs running on CPU, there are many high-level general-purpose programming languages such as Fortran, C++, etc. For a long time, however, no corresponding programming languages exist for generating machine codes running on GPU. In November 2006, NVIDIA launched "Compute Unified Device Architecture" (CUDA), a general purpose parallel computing platform and programming model that leverages the parallel compute engine in NVIDIA GPUs to solve complex computational problems in a more efficient way than on a CPU. This new GPU programming language is actually a full extension of C and a partial extension of C++. So programmers familiar with C/C++ should find it easy to write CUDA programs. In the following sub-sections, we will introduce the basic concepts of CUDA programming.

### 3.2.1 CUDA Kernel

A complete sample of CUDA codes usually includes two parts: GPU specific codes and general CPU codes. A CUDA program, like many others, starts executing the part of CPU codes, which should later call the GPU codes, i.e. CUDA kernels to run parallel simulations. A CUDA kernel, marked by `__global__` declaration specifier, is a function that can be concurrently executed by a large number of GPU threads. When invoking this kernel, we must specify the number of threads by block number and block size within the `<<...>>` execution configuration syntax as follow:

```

// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}
int main()
{
    ...
    // Kernel invocation with N = 128*128 threads
    VecAdd<<<128, 128>>>(A, B, C);
    ...
}

```

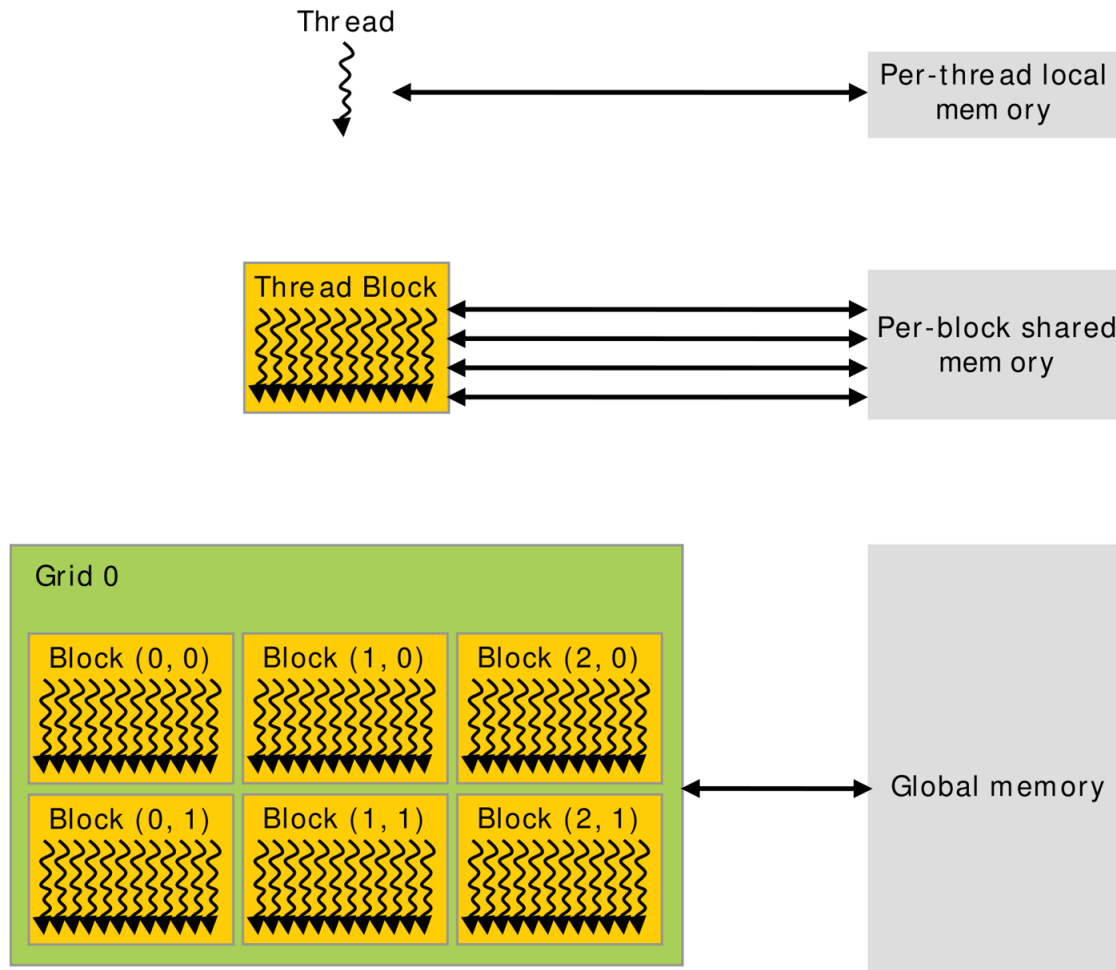
(3.1)

### 3.2.2 Thread Hierarchy

There could be thousands of or even more threads executing concurrently on GPU, so it's better to manage these threads in groups to maximize resource utilization. As discussed in section 3.1, GPU is designed in SIMD model for higher performance-power ratio. On CUDA platform, 32 threads are grouped into a “warp” for SIMD purpose. That is, the GPU will have maximum performance if the 32 threads execute the same instruction code (may access different data). If the warp bifurcates through *if/switch* statements, the execution will be serialized and become less efficient. A number of warps are then correlated to form a “block”, which is executed on a single “streaming multiprocessors” (SMs) of NVIDIA's GPU. Different blocks may be executed concurrently on the same SM or on several different SMs depending on available computing resources. Threads within a block can cooperate by sharing data through some shared memory and by synchronizing their execution to coordinate memory accesses. More precisely, one can specify synchronization points in the kernel by calling the `__syncthreads()` intrinsic function; `__syncthreads()` acts as a barrier at which all threads in the lock must wait before any is allowed to proceed. Moreover, a number of blocks form a grid, which could be one, two, or three dimensional.

### 3.2.3 Memory Hierarchy

CUDA threads may access data from multiple memory spaces during their execution as illustrated by Figure 3.3. Each thread has private local memory. Each thread block has shared memory visible to all threads of the block and with the same lifetime as the block. All threads have access to the same global memory.



**Figure 3.3** Memory architecture corresponding to thread architecture in CUDA

There are also two additional read-only memory spaces accessible by all threads: the constant and texture memory spaces. The global, constant, and texture memory spaces are optimized for different memory usages. Texture memory also offers different addressing modes,

as well as data filtering, for some specific data formats. The global, constant, and texture memory spaces are persistent across kernel launches by the same application.

### **Global memory**

Global memory resides in device memory and device memory is accessed via 32-, 64-, or 128-byte memory transactions. These memory transactions must be naturally aligned: Only the 32-, 64-, or 128-byte segments of device memory that are aligned to their size (i.e., whose first address is a multiple of their size) can be read or written by memory transactions. When a warp executes an instruction that accesses global memory, it coalesces the memory accesses of the threads within the warp into one or more of these memory transactions depending on the size of the word accessed by each thread and the distribution of the memory addresses across the threads. In general, the more transactions are necessary, the more unused words are transferred in addition to the words accessed by the threads, reducing the instruction throughput accordingly.

Global memory has the largest size (order of GB) among all the available memories on GPU. However, it is off-chip memory with long memory access latency (400 ~ 600 clock cycle) and is only cached by very small L2 cache, so the cache mechanism does not work effective as CPU that has multiple layers of cache structure.

### **Local memory**

Local memory accesses only occur for some automatic variables. Automatic variables that the compiler is likely to place in local memory are: (1) Arrays for which it cannot determine that they are indexed with constant quantities, (2) Large structures or arrays that would consume too much register space, (3) Any variable if the kernel uses more registers than available (known as register spilling).



The local memory space resides in device memory, so local memory accesses have same high latency and low bandwidth as global memory accesses and are subject to the same requirements for memory coalescing. Local memory is however organized such that consecutive 32-bit words are accessed by consecutive thread IDs. Accesses are therefore fully coalesced as long as all threads in a warp access the same relative address (e.g., same index in an array variable, same member in a structure variable).

### **Shared memory**

Because it is on-chip, shared memory has much higher bandwidth and much lower latency (20 ~ 40 clock cycle) than local or global memory. To achieve high bandwidth, shared memory is divided into equally-sized memory modules, called banks, which can be accessed simultaneously. Any memory read or write request made of  $n$  addresses that fall in  $n$  distinct memory banks can therefore be serviced simultaneously, yielding an overall bandwidth that is  $n$  times as high as the bandwidth of a single module. However, if two addresses of a memory request fall in the same memory bank, there is a bank conflict and the access has to be serialized. The hardware splits a memory request with bank conflicts into as many separate conflict-free requests as necessary, decreasing throughput by a factor equal to the number of separate memory requests. If the number of separate memory requests is  $n$ , the initial memory request is said to cause  $n$ -way bank conflicts. BTW, the size of shared memory for each SM is 64 KB or 96KB for latest NVIDIA GPUs.

### **Constant Memory**

The constant memory space resides in device memory and is cached in the constant cache. A request is then split into as many separate requests as there are different memory addresses in the initial request, decreasing throughput by a factor equal to the number of separate requests. The

resulting requests are then serviced at the throughput of the constant cache in case of a cache hit, or at the throughput of device memory otherwise. The size limit of constant memory for single CUDA instance is 64 KB, and the size of cache behind it is merely 8 KB.

### **Texture and Surface Memory**

The texture and surface memory spaces reside in device memory and are cached in texture cache, so a texture fetch or surface read costs one memory read from device memory only on a cache miss, otherwise it just costs one read from texture cache. The texture cache is optimized for 2D spatial locality, so threads of the same warp that read texture or surface addresses that are close together in 2D will achieve best performance. The texture cache is actually shared with L1 cache, whose size is 24 KB.

### **3.2.4 Difficulties of GPU programming**

Firstly, modern GPUs are designed in SIMD (single instruction multiple data) architecture instead of MIMD (multiple instruction multiple data) due to efficiency and complexity reasons. If threads diverge to different instruction flows through *if/switch* statements, the GPU work scheduler will simply execute the instruction flow in series and become less efficient. In general, more deeply nested diverging statements will result in lower efficiency. However, the algorithms behind each particle interaction are typically very sophisticated and the bifurcation caused by *if/switch* statements occurs frequently.

Secondly, GPU programming needs to handle several types of memories so it's not convenient as CPU which has unified memory access model. The memory hardware in GPU is not ideal for Monte Carol simulation either. The largest device memory (~GB) is only cached by a very small L2 cache, and so cache miss happens frequently and thus causes a long memory accessing latency. Shared and constant memories are hundreds of times faster than device

memory, but have a size (~KB) far less than that is necessary for Monte Carlo simulation that usually include large material data tables.

Therefore, we need to design dedicated MC algorithms suitable for GPU's SIMD feature and allocated the scarce fast memory carefully to improve the simulation efficiency.

### 3.3 Prototype: PENELOPE

It is obviously unwise to build a complex MC system from scratches. Instead, we should pick one mature MC system as prototype and develop GPU implementation based on it. In chapter 1, we introduced several outstanding “accuracy-oriented” Monte Carlo packages such as MCNP [10], Geant4[11], EGS4/EGSnrc [13], and PENELOPE [12]. Among these we choose PENELOPE as the prototype for five compelling reasons as follows:

- (1) It has a compact simulation kernel of roughly 3,000 FORTRAN lines. This saves a lot of work when adapting PENELOPE to GPU version.
- (2) It has been well validated by various experiments. This guarantees the GPU version is also of high accuracy.
- (3) It includes very detailed documents. We can easily find everything about the physics model and codes implementation details behind it.
- (4) It includes build-in support of simulation of charged particles in magnetic field. The ViewRay's MRIdian platform we are working on introduced a 3.5 Tesla magnetic field for MR imaging. It's convenient to have a module to refer to when building our own MC engine.
- (5) It covers 280 common materials and can generate data tables for new materials via composing from weighted elements.

Items (1) and (2) are the key reasons for PENELOPE to win over other opponents, and the rest offer us more convenience to develop the clinically applicable GPU MC system. We name it as gPENELOPE in order to honor the contribution of the original PENELOPE codes.

### **3.4 PENELOPE in C++: cPENELOPE**

PENELOPE was implemented in FORTRAN 77 with an archaic programming style (e.g. many antiquated “GOTO” statements). Although GPU programming for FORTRAN is available through the PGI compiler (via collaboration with NVIDIA), it lacks good object-oriented programming support and some general libraries, so convenient features like batch work and file compression cannot be implemented easily. Moreover, as the MRIdian head model was developed in C++, rewriting the PENELOPE kernel in C++ first will more readily enable its application to MRIdian platform. In this section, we will discuss the how to translate PENELOPE into C++ language, i.e. cPENELOPE, and the method to validate the build.

#### **3.4.1 Rewrite PENELOPE in C++**

To build PENELOPE in C++, we first extracted all relevant material data tables to a class called Material, and assigned shared data to global variables. All “jump” and “knock” functions were rewritten as member functions of this class in an optimized logic sequence. The lengthy code for generating data tables (over 7,000 lines) did not need to be translated to C++; instead, we added an interface function in PENELOPE that we compiled to a DLL module. We can call this DLL in C++ to preprocess materials and export the relevant data table to a file that will be addressed to memory by the C++ code later.

The original PENELOPE only supports single-thread processes, while multithreading through OpenMP is necessary to fully exploit modern multi-core CPUs. We ensure that all kernel functions are thread-safe by managing thread-related variables accordingly in a single

class. We additionally exploit MPI (with a set of workload balancing functions for optimizing overall performance) to enable parallel simulation on a distributed network. The random number generators (RNGs) are kept thread-private and are initiated with independent seeds, which are provided by a different type of RNG (e.g. 16807 RNG [24]) in our implementation.

### 3.4.2 User-friendly features

The original PENELOPE configuration file has strict formatting restrictions that consequently require changes to the source code when adding or deleting certain configuration items. We thus developed an elegant script module that supports “C”-style free writing, declaring nested cells, and macro definitions for ease of use (Figure 3.4).

```
SOURCEHEAD =
{
    # Slightly adjust the MLC open width to match the experiment
    Leaf spacing adjustment = -0.07 # unit cm. Currently default to -0.07

    # Two kinds of beam files are accepted: (1)mfQADData.beams (2)PlanOverview.txt
    Beams file = $cdir$PlanOverview.txt

    # If Beams file doesn't exists, the following customized beams will be used
    isocenter = 0 0 0 #unit cm. Isocenter's position relative to the patient phantom cen
    Beam =
    {
        gantry angle = 0 #in degree
        head index = 1 # The head index may be used in future version
        # The "segment" block can be repeated as many times as needed
        segment =
        {
            on time = 10 #uint second
            leaf position = 0 29 0 0
            leaf position = 13 16 -2.10 2.10 #the later one will overlap part of previou
        }
    }
}
```

**Figure 3.4** The sample of “C”-style configuration script

A powerful log module was also developed to manage file records, run batch tasks, implement dose reuse, and provide e-mail notifications. We also developed a binary file manager module powered by real-time compressing and decompressing algorithms (lz and quicklz) to

handle large phantom and dose output files that would otherwise cost a lot of storage space and bandwidth for synchronizing remote simulations.

### 3.4.3 Transport in heterogeneous phantom

The original PENELOPE was originally designed for simulation in homogenous materials.

However, the heterogeneous phantoms captured from CT scan of human body is more common in radiation therapy. These phantoms can be approximated by a 3D density grid made up of numbers of tiny homogenous cubic voxels (order of mm width). Then PENELOPE is able to simulate particle transport in this grid through a dedicated boundary crossing (BC) algorithm. However, the computation cost of BC algorithms is usually not cheap, and a better algorithm called Woodcock tracking [40] was invented to avoid boundary crossing of photons.

For photon simulation, we added Woodcock tracking to the original PENELOPE to treat a heterogeneous phantom as uniform. In order to obtain an invariant MFP  $\lambda = m_0/(\rho_i\sigma)$  across the whole phantom ( $m_0$  is the molecular mass,  $\rho_i$  is the voxel density, and  $\sigma$  is the total scattering cross-section), we add a virtual scattering cross-section  $\sigma_i$  in each voxel  $i$  to maintain  $\rho_i(\sigma + \sigma_i) = \rho_{\max}\sigma$  constant everywhere. Then the probability for this virtual scattering to happen is

$$p(\sigma_i) = \frac{\sigma_i}{\sigma + \sigma_i} = 1 - \frac{\rho_i}{\rho_{\max}} \quad (3.2)$$

If this virtual interaction is sampled during a “knock” event, we just continue to propagate the photon without changing direction or losing energy since the virtual event is not real. A shortcoming of this technique is that it could result in low efficiency for a phantom composed mainly of low density material (e.g. lung) because the virtual interaction will most likely be sampled thus wasting random numbers without any energy transfer. We thus instead try to

improve the sampling efficiency by forcing the real interactions to always happen, with the secondary particles' weight reduced by factor  $\rho_i/\rho_{max}$  and only  $\rho_i/\rho_{max}$  of the primary photons' status (energy and direction) being changed. This ensures the probability distribution of deposited energy is unbiased.

For electron and positron simulation, PENELOPE applied the “mixed” condensed history scheme, which treats large energy transfer collisions in an analogue way and uses the continuous slowing down approximation (CSDA) [41] to model small-loss collisions. Since the CSDA range  $\bar{s}$  is much smaller than the photon's mean free path  $\lambda$ , we implemented a simple grid detection algorithm to trace the CSDA jump between heterogeneous voxels. Unlike photons, electrons and positrons will cross just a few voxels before being completely stopped. Though soft collisions occur at a high frequency, most of these are determined not to cross the voxel boundary by a rapid test that roughly estimates the nearest distance to the boundary, and so the necessary time for calculating exact crossing points at voxel boundaries is actually not expensive.

### 3.4.4 Transport in magnetic field

Given that the CSDA range  $\bar{s}$  of electrons and positrons is typically very small, the magnetic field in each voxel can be treated as uniform in most applications. The particles will undergo spiral motion in a uniform field  $\mathbf{B}$  at the relativistic angular velocity

$$\vec{\omega} = -\frac{e \vec{B}}{\gamma m_e} \quad (3.3)$$

where  $e$  denotes elementary charge,  $m_e$  is electron mass, and  $\gamma$  is the Lorentz factor. The corresponding location after advancing length  $s$  in a uniform phantom can be easily evaluated as shown in the PENELOPE user manual to be

$$\vec{r}(s) = \vec{r}_0 + s\hat{v}_0 - \frac{s}{v_0} \vec{v}_{0\perp} + \frac{1}{\omega} [1 - \cos(s\omega/v_0)](\hat{\omega} \times \vec{v}_{0\perp}) + \frac{1}{\omega} \sin(s\omega/v_0) \vec{v}_{0\perp} \quad (3.4)$$

where  $\mathbf{r}_0$  is the initial particle location and  $\mathbf{v}_0$  is the particle velocity (with  $\mathbf{v}_{0\perp}$  being the velocity component perpendicular to  $\mathbf{B}$ ). For a heterogeneous voxelized phantom, however, the intersection between the spiral curve and the voxel boundary must be calculated due to variation of the density in each voxel. Accurate evaluation is messy and inefficient due to many inverse trigonometric function calls. As the CSDA range  $\bar{s}$  is relatively small in comparison to the spiral radius  $R$ , we can approximate the spiral motion by small straight-line segments that change direction gradually. Taking the allowed error in one segment move to be  $\Delta_{\max}$ , the maximum segment length  $s_m$  is expressed as

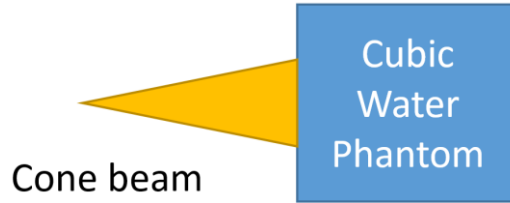
$$s_m = \sqrt{(R + \Delta_{\max})^2 - R^2} \approx \sqrt{2R\Delta_{\max}} \quad (3.5)$$

If  $s > s_m$ , we only advance a distance  $s_m$  and change direction by angle  $\theta \approx s_m/R$  (continuing until  $s$  is exhausted). This straight-line advancing procedure uses the same voxel tracking implementation as the situation without magnetic field. While moving a distance  $s$  may cross a voxel boundary, the particle direction may point back to the original voxel when  $\mathbf{v} \cdot (\mathbf{v} + d\mathbf{v}) < 0$ , and so the current voxel index must be corrected accordingly.

### 3.4.5 Validate cPENELOPE

Before adapting the C++ implementation onto a GPU, we must validate that the C++ version produced identical results to the original FORTRAN code. We setup a simple cone beam incident on a cubic water phantom (Figure 3.5), ran the two versions with various incident energies, angles, and cutoff energies in single-thread mode, and then exported the particle status of  $10^8$  serial steps for comparison. Considering the possible runtime library differences between C++ and FORTRAN, we set the allowed error of position, direction, and energy for each step to be  $10^{-10}$  cm,  $10^{-10}$  and  $10^{-10}$  keV, respectively. We obtained 100% identical step-status outputs, suggesting that our C++ code is completely equivalent to the original FORTRAN code.





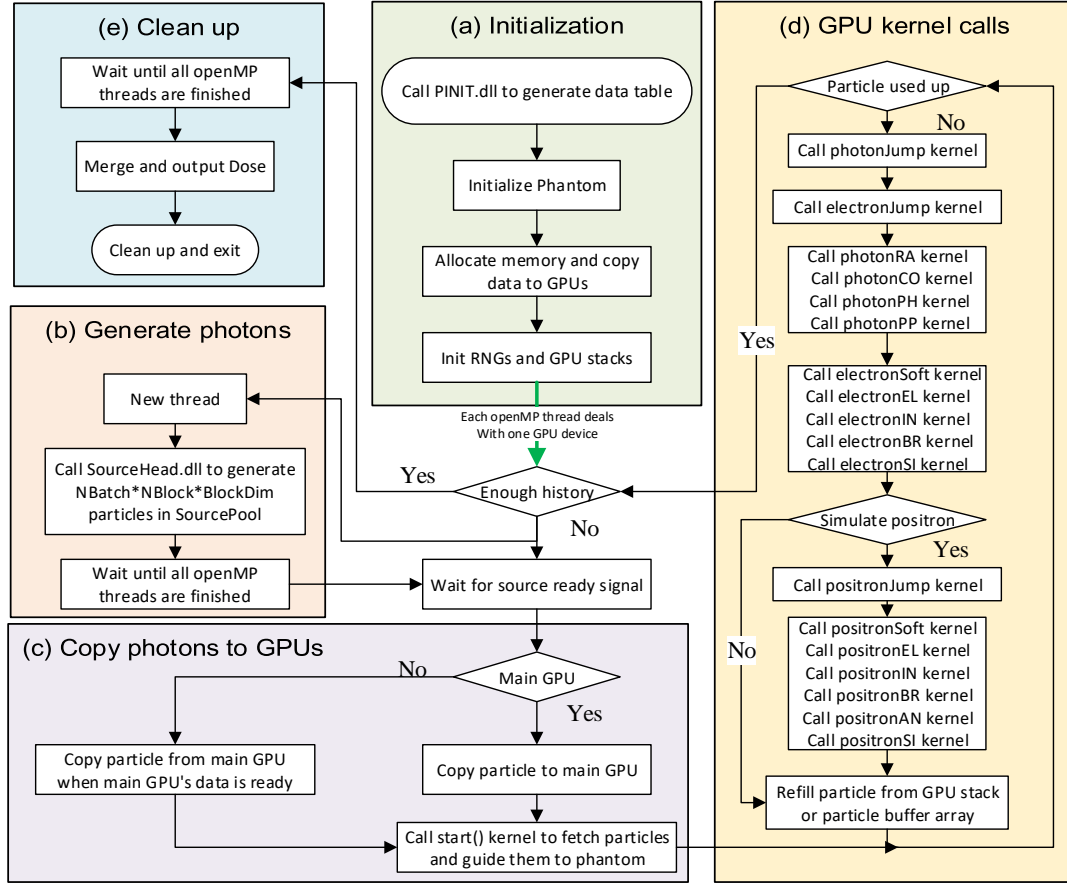
**Figure 3.5** The setup with cone beam and cubic water phantom to test the output of cPENELOPE

## 3.5 cPENELOPE to gPENELOPE

Upon confirming the integrity of our C++ code, we proceeded to port the C++ code to CUDA, a C-extended GPU programming language. Though CUDA greatly simplified parallel programming on GPUs, it suffers from two main restrictions introduced in section (3.24) when comparing to CPU programming. As the PENELOPE includes sets of very large data table, it's very difficult to optimize the memory access pattern. Therefore we decided to devote the majority of our effort to reduce the thread divergence by designing dedicated workflow optimized for GPU.

### 3.5.1 GPU workflow

In all Monte Carlo codes, instruction divergence is common so we aim to improve efficiency by minimizing the number of nested diverging statements in each CUDA kernel function. Instead of organizing all the simulation code in one kernel function, we decided to split the code into several independent kernel functions which process different types of scattering events and let the CPU call these kernel functions in a loop within a main function as shown in Figure 3.6.



**Figure 3.6** Workflow of gPENELOPE including (a) Initialization (b) Generate photons (c) Copy photons to GPUs (d) GPU kernel calls: RA, CO, PH and PP are short for Rayleigh, Compton, photoelectric and pair production while EL, IN, BR, SI, AN are short for elastic, inelastic, bremsstrahlung, shell ionization and annihilation respectively. The “kernel-by-kernel” calls in sequence can help reducing instruction divergence. (e) Clean up.

As shown in Figure 3.6(a), the program reads and parses the configuration file which includes details regarding the GPU devices, phantom (geometry and materials), and source head. The program then calls a DLL (see section 3.4.1) to generate necessary data tables for relevant materials. These data and phantom information are then copied to GPU device memory with pointers to large arrays and some small data tables being copied to constant memory on the GPU instead for improving accessing speed. In addition, a random number generator and particle stack

is initiated for each GPU thread. As our workstation includes multiple GPU cards, we next launch multiple threads through OpenMP to call GPU kernel functions on each card simultaneously.

Meanwhile, the main thread launches another thread calling the vendor-provided head source module to prepare incident photons as shown in Figure 3.6 (b). As specified in Figure 3.6(c), photons are then copied to the main GPU and in turn transferred to other GPUs in order to save I/O time. The GPU kernel function *start()* is then called to guide photons to the phantom via free propagation. As summarized in Figure 3.6(d), distinct “jumping” kernels for photons and charged particles are called to advance relevant particles and label them with the type of interaction that will happen next. These interaction kernel functions are called sequentially such that particles labeled with a different interaction type will simply exit their threads. Though this schedule does not completely resolve the instruction divergence problem, it lowers the level of nested diverging statements, and thus reduces the total pausing time. After all interaction kernels finish, we refill the current particle variable either from the stack storing secondary particles or the incident photon buffer array, thus improving efficiency by enabling constant renewal of particles on all threads.

In addition, we provide an option for toggling positron simulation as the primary photon energy of  $^{60}\text{Co}$  is just slightly higher than the threshold for pair production (twice the electron rest mass). We also allow for source particle reuse as occasionally the head model lags the GPU and cannot provide new particles at a sufficient rate. Our simulation comparisons show that the dose differences in “hot areas” ( $D > 10\% \times D_{\text{max}}$ ) caused by reusing source particles are almost totally (99.73%) within the targeted 1% uncertainty for a large ( $10^9$ ) history number. Moreover,

we setup the RNGs to refill their buffers after N loops in order to reduce instruction divergence, where N is the average number of loop iterations when an RNG buffer is exhausted.

### **3.5.2 MRIdian head model**

The vendor-provided MRIdian head model provides phase space data including the energy spectrum and flux for a given solid angle for the  $^{60}\text{Co}$  source. Each IMRT beam consists of a collection of segments configuring the MLC shape and beam-on time. In our code, each segment is treated as a simulation unit and the history number assigned to each unit is weighted by its beam-on time. The MLC shape determines how many photons will be exported from the  $^{60}\text{Co}$  head for each history, which is a non-fixed number due to the patient-specific MLC configuration.

To maximize its efficiency, the GPU should process a fixed number N of photons per batch. Therefore, we designed a class to buffer the photons supplied by the head in a multi-threading queue such that N photons are fetched in a batch by the GPU when the class is filled with slightly over N photons. The excess photons are then moved to the head of the queue to continue the buffering.

### **3.5.3 Performance benchmark**

The hardware for our tests is a server that includes an Intel Xeon E5 2630 v3 CPU and an NVIDIA Tesla K80 GPU card. The CPU can provide 16 true simultaneous threads, giving an overall processing rate of 47 thousand histories per second. The GPU achieves a simulation rate of 245 thousand histories per second, which is about 5 times faster than the CPU platform. In other words, gPENELOPE can finish simulating one treatment plan in one hours while cPENELOPE has to spend 5~6 hours. Considering the original PENELOPE engine only supports one thread, our GPU code can roughly accelerate PENELOPE by a factor of 80.

## 3.6 Validate gPENELOPE

Before gPENELOPE can be used any application, we must verify that it keeps the same accuracy of the original PENELOPE. In this section, we will define the criteria of comparing two sets of 3D dose, and perform single-thread and multi-thread dose comparisons to prove gPENELOPE works as we expected.

### 3.6.1 3D dose comparisons

For 3D comparisons, we use both gamma passing rates and statistical histograms to reveal differences between Monte Carlo systems. The gamma index for each voxel  $\vec{r}$  is defined by [42]

$$\gamma(\vec{r}) = \min\{\Gamma(\vec{r}, \vec{r}')\} \forall \{\vec{r}'\},$$

$$\Gamma(\vec{r}, \vec{r}') = \sqrt{\frac{|\vec{r} - \vec{r}'|^2}{\Delta d^2} + \frac{(D(\vec{r}) - D'(\vec{r}'))^2}{\Delta D^2}} \quad (3.6)$$

where  $|\vec{r} - \vec{r}'|$  represents the distance between voxels  $\vec{r}$  and  $\vec{r}'$ ,  $\Delta d$  is the distance-to-agreement (DTA) value, and  $\Delta D$  is the dose tolerance value. We label a gamma index at voxel  $\vec{r}$  as passing if  $\gamma(\vec{r}) \leq 1.0$ , and count the passing rate for those voxels where  $D(\vec{r}) > t \cdot D_{max}$ , where  $t$  is a dose threshold. Higher gamma passing rates for smaller  $\Delta d$  and  $\Delta D$  tolerances usually suggest stronger agreement between two dose distributions. Here we use strict criteria (DTA = 0, i.e. grid-to-grid comparison,  $\Delta D = 0.5D_{max}$ , threshold  $D(\vec{r}) > 0.1D_{max}$ ) to amplify the differences as the three Monte Carlo engines behave quite similarly to each other.

A statistical histogram, on the other hand, visually indicates the distribution of dose differences spanning all voxels. Here we define a statistical variable z-score for each voxel as

$$z(\vec{r}) = \frac{D_{test}(\vec{r}) - D_{ref}(\vec{r})}{D_{ref}(\vec{r})} \cdot \frac{1}{\sigma_{tot}} \quad (3.7)$$

where  $D_{test}(\vec{r})$  and  $D_{ref}(\vec{r})$  are test and reference dose at voxel  $\vec{r}$  respectively, and  $\sigma_{tot}$  is the standard deviation of the distribution  $(D_{test}(\vec{r}) - D_{ref}(\vec{r}))/D_{ref}(\vec{r})$  spanning all voxels, which functions as a normalization factor and an indicator of the difference level. If two engines are identical, the histogram of z-scores will be a standard Gaussian distribution, and  $\sigma_{tot}$  will approach  $\sqrt{\sigma_{test}^2 + \sigma_{ref}^2}$ , where  $\sigma_{test}$  and  $\sigma_{ref}$  are the uncertainties achieved by the two algorithms in question.

Aside from accuracy, we also performed detailed efficiency comparisons between Monte Carlo engines. Since the uncertainty  $\sigma$  is approximately proportional to  $1/\sqrt{N}$  where  $N$  is the history number which is in turn proportional to simulation time  $T$ , i.e.  $\sigma \cong 1/\sqrt{\eta T}$ , we define this constant of proportionality  $\eta$  as the calculation efficiency:

$$\eta \cong \frac{1}{\sigma^2 T} \quad (3.8)$$

As the dose uncertainty  $\sigma_i$  varies by voxel,  $\sigma$  is calculated by squared root averaging, i.e.

$$\sigma = \sqrt{\frac{\sum_{i=1}^N \sigma_i^2}{N}} \quad (3.9)$$

where  $N$  is the number of voxels whose dose is above a given threshold ( $D(\vec{r}) > 0.1D_{max}$  in our tests).

### 3.6.2 Comparisons in single thread

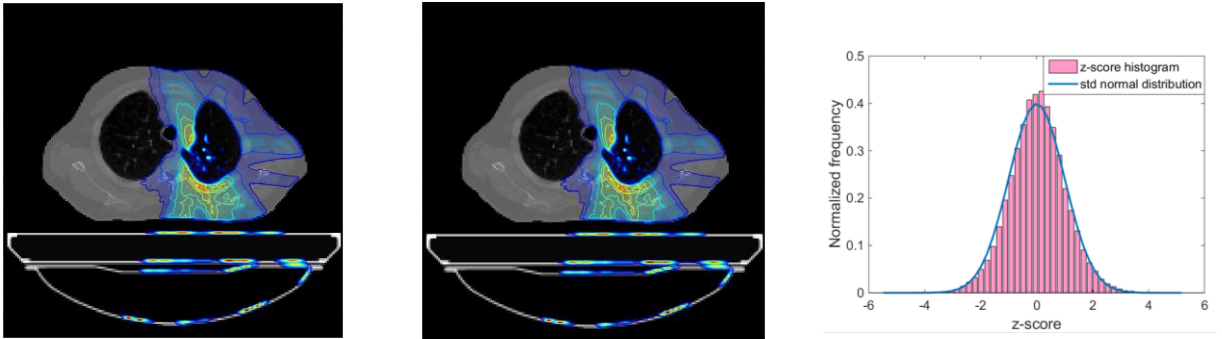
In section 3.4.5, we showed that C++ PENELOPE performs equivalently to the original code written in FORTRAN by a detailed step-by-step comparison. Here we apply a similar approach to convincingly show that gPENELOPE performs equivalently to C++ PENELOPE in single-thread operation. We simulate a complex lung IMRT plan (shown in Figure 3.7) using both

platforms and output particle statuses (position, velocity and energy) in  $10^7$  knocking steps for comparison. The maximum and average differences are summarized in Table 3.1. The energy difference  $|dE|$  is negligible in comparison to the incident energy ( $>1$  MeV).

**Table 3.1** Particle status differences of  $10^7$  steps between gPENELOPE and C++ PENELOPE

$\max( d\vec{x} ) = 1.42 \times 10^{-8} \text{ cm}$	$\max( d\hat{v} ) = 1.82 \times 10^{-9}$	$\max( dE ) = 2.25 \times 10^{-5} \text{ eV}$
$\text{mean}( d\vec{x} ) = 4.54 \times 10^{-14} \text{ cm}$	$\text{mean}( d\hat{v} ) = 1.26 \times 10^{-14}$	$\text{mean}( dE ) = 7.29 \times 10^{-9} \text{ eV}$

We additionally run  $10^6$  histories to check differences in dose distributions directly, which turn out to be  $\max(|dD|) = 7.63 \times 10^{-5} \text{ Gy}$ ,  $\text{mean}(|dD|) = 6.78 \times 10^{-8} \text{ Gy}$ ,  $\sigma(dD) = 7.17 \times 10^{-7} \text{ Gy}$ . The prescription dose for this patient is 50 Gy. That is, the maximum relative dose error is  $1.36 \times 10^{-9}$ . Considering the possible runtime library differences between the GPU and CPU, the status tracking and dose deposition comparisons together show that gPENELOPE and C++ PENELOPE can effectively be considered identical in single-thread mode.



**Figure 3.7** Dose distributions for a lung case calculated by C++ PENELOPE (left) and gPENELOPE (middle). The two distributions are identical within 1%. The frequency distribution of z-scores in comparison to a standard normal distribution (right).

### 3.6.3 Comparisons in multithreads

Though impractical to compare particle status with gPENELOPE in multi-threaded operation (considering thousands of threads simultaneously), we can compare dose distributions generated by gPENELOPE and C++ PENELOPE directly. We thus run a large number of histories ( $4 \times 10^9$ ) to ensure that the target area ( $D > 10\% \times D_{\max}$  region) reaches less than 0.5% uncertainty so that the maximum allowed difference would be less than 1% if gPENELOPE behaves equivalently to C++ PENELOPE. Comparing doses in these voxels, we found that

$$\frac{\max(|dD|)}{\max(D_{ref})} = 0.93\%, \quad \frac{\text{mean}(|dD|)}{\max(D_{ref})} = 0.12\%, \quad \frac{\sigma(dD)}{\max(D_{ref})} = 0.15\% \quad (3.10)$$

where  $D_{ref}$  is the dose calculated by C++ PENELOPE. The results indicate the equivalency assertion between gPENELOPE and PENELOPE is valid. In addition, we compared the frequency distribution of z-scores to a standard normal distribution as shown in Figure 3.7(c). This comparison indicates that the z-score distribution follows a standard normal distribution.

Since gPENELOPE is effectively equivalent to C++ PENELOPE in single-thread mode, and dose distributions generated by the two agree well within expected statistical uncertainties in multi-thread operation, we safely deduce that gPENELOPE is a faithful adaptation of PENELOPE that does not compromise accuracy.

## 3.7 Application 1: validate MRIdian head model

Beyond demonstrating that the gPENELOPE simulation kernel is both fast and accurate, we must confirm that the entire validation system is correctly modeled (especially the  $^{60}\text{Co}$  head) in order to ensure safe deployment in the clinic. We thus investigate several vital comparisons to experimental measurements to validate its overall accuracy.



### 3.7.1 Depth dose

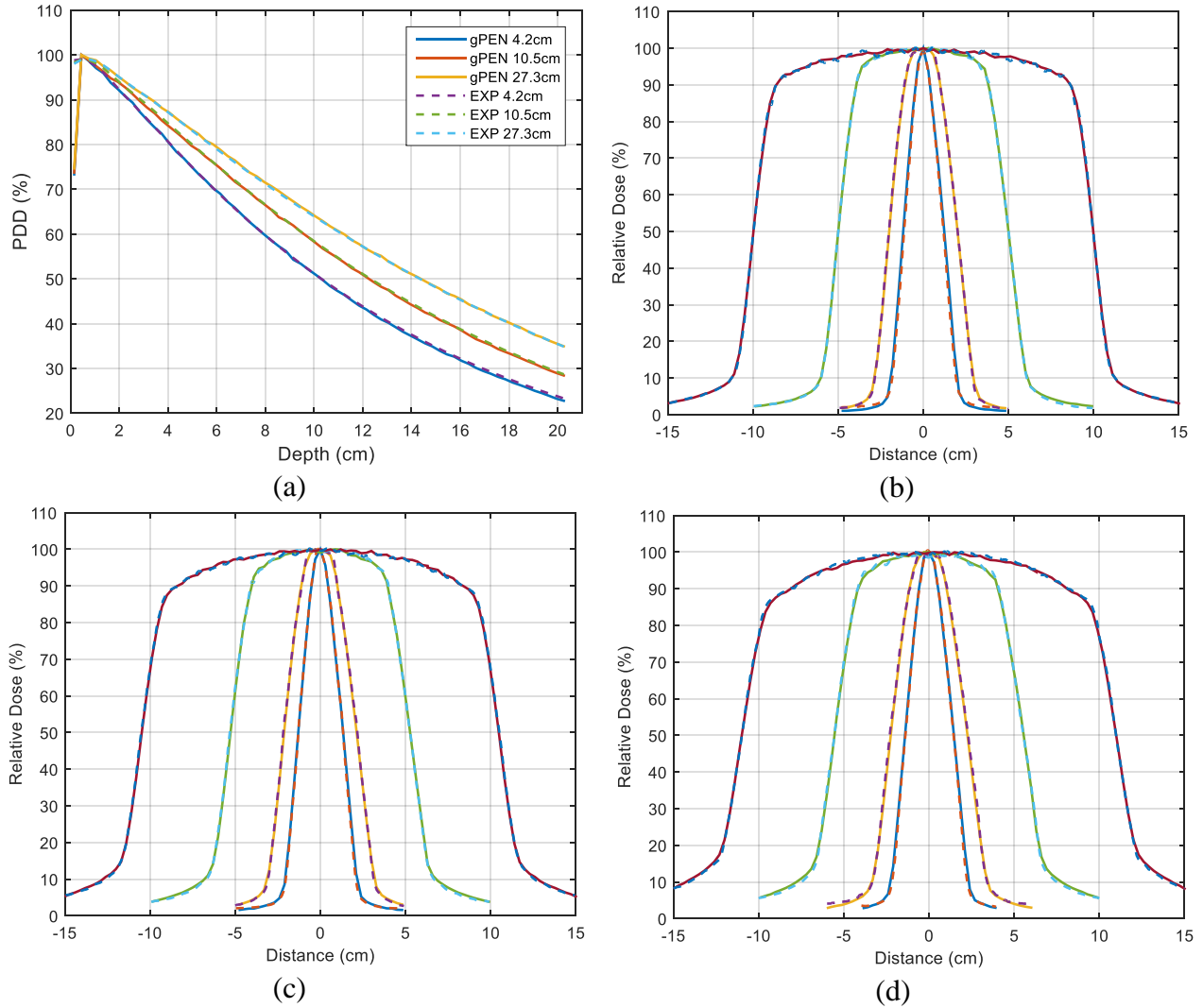
Measurements were performed in a cubic water phantom ( $30 \times 30 \times 30 \text{ cm}^3$ ) placed at  $\text{SSD} = 100 \text{ cm}$  using small, medium and large field sizes ( $4.2 \times 4.2 \text{ cm}^2$ ,  $10.5 \times 10.5 \text{ cm}^2$  and  $27.3 \times 27.3 \text{ cm}^2$ ).

The data was collected using an Extradin A18 ion chamber. Note that the chamber is manually positioned at different depths as an MRI compatible beam scanning device is not commercially available now. Considering the cylindrical dimensions of the ion chamber (radius = 2.5 mm, height = 6.4 mm), we set the voxel size of the phantom to be  $3 \times 3 \times 3 \text{ mm}^3$  in simulation, and run  $10^9$  histories to ensure sufficiently small statistical uncertainty ( $< 0.5\%$  for  $D > 50\% D_{\text{max}}$  region).

Figure 3.8 (a) shows comparisons between simulation and experimental data, yielding less than 1% difference.

### 3.7.2 Off-axis profile

We used EBT2 radiochromic films placed at depths of 5, 10 and 15 cm to sample planar doses for comparisons to simulation results. As shown in Figure 3.8 (b) (c) (d), the simulated dose profiles agree well with measured data to within 2% or 2 mm distance-to-agreement (DTA).



**Figure 3.8** Percentage-depth-dose comparisons between gPENELOPE and ionization chamber measurement (spline interpolated) for field sizes  $4.2 \times 4.2$ ,  $10.5 \times 10.5$  and  $27.3 \times 27.3$  cm<sup>2</sup> at 100 cm SSD (a). Off-axis profile comparisons between gPENELOPE and radiochromic film measurement for field sizes  $2.1 \times 2.1$ ,  $4.2 \times 4.2$ ,  $10.5 \times 10.5$ , and  $21.0 \times 21.0$  cm<sup>2</sup> at depths of 5 (b), 10 (c) and 15 cm (d) at 100 cm SSD.

### 3.7.3 Output factor

Both square and rectangular field output factor measurements were performed in a cubic water phantom ( $30 \times 30 \times 30$  cm<sup>3</sup>) placed at SSD = 100 cm. The Extradin A18 ion chamber was placed at 5 cm below the surface, i.e. the isocenter. Note that in order to verify the small field output factor,

we closed the central ten leaves incrementally from 10.5 to 0.6 cm, as recommended by ViewRay. As shown in Table 3.2, the calculated output factors match well with experimental data (<2%).

**Table 3.2** Output factor comparison for square and rectangular fields

Field shape	Size (cm <sup>2</sup> )	OF (gPEN)	OF (Exp)	Diff. (%)
Square field	4.2 × 4.2	0.8839	0.8780	0.67
	6.3 × 6.3	0.9414	0.9380	0.36
	10.5 × 10.5	1.0000	1.0000	NA
	14.7 × 14.7	1.0293	1.0410	-1.12
	27.3 × 27.3	1.0624	1.0700	-0.71
Rectangular field	0.6 × 10.5	0.2103	0.2070	1.58
	0.8 × 10.5	0.2839	0.2825	0.51
	1.0 × 10.5	0.3607	0.3568	1.08
	1.5 × 10.5	0.5256	0.5246	0.19
	2.0 × 10.5	0.6741	0.6721	0.30
	2.5 × 10.5	0.7953	0.7859	1.19
	3.0 × 10.5	0.8730	0.8583	1.71
	4.0 × 10.5	0.9222	0.9119	1.13
	6.0 × 10.5	0.9636	0.9582	0.56
	8.0 × 10.5	0.9837	0.9822	0.16
	10.5 × 10.5	1.0000	1.0000	NA

### 3.7.3 AAPM TG-119

The AAPM **Task Group 119** (TG-119) recommends that six cases be considered (two non-IMRT and four IMRT) for IMRT commissioning, including AP-PA, Bands, Multi-target, C-shape, Head & Neck, and Prostate. These treatment plans were planned using MRIdian's inverse treatment planning system and delivered to a 30×30×15 cm<sup>3</sup> water-equivalent plastic phantom containing ionization chambers and EDR2 radiographic film.

#### TG-119 point dose

All point dose measurements were made using an ADCL calibrated ionization chamber (Extradin A18). The dose at each plan's isocenter (except C-shape) is measured to evaluate high dose accuracy while a few adjacent points are chosen to examine low dose accuracy.

**Table 3.3** TG-119 point dose comparisons: gPENELOPE vs. ionization chamber (IC)

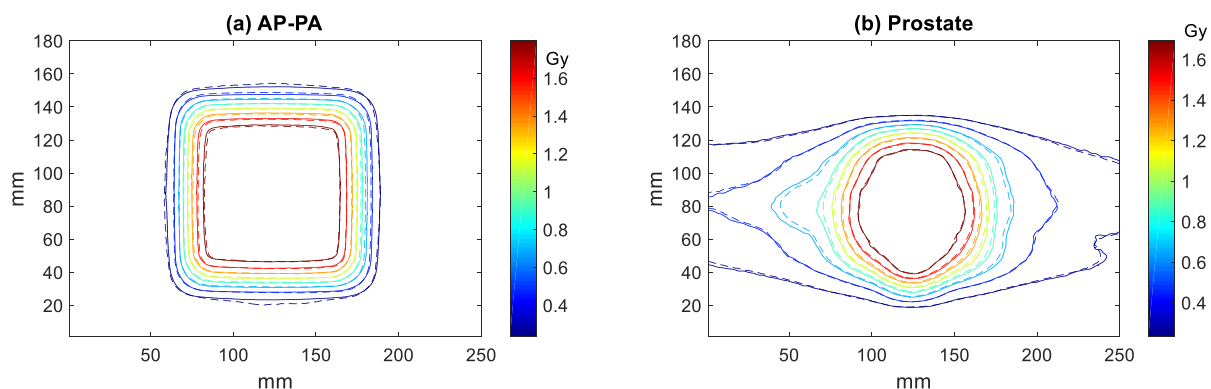
TG-119 plans	Location	IC (Gy)	gPENELOPE (Gy)	Diff. (%)
AP-PA	Isocenter	1.988	1.991	0.16
Bands	Isocenter	1.422	1.426	0.31
Multi-target	Isocenter	2.085	2.058	-1.29
Multi-target	4 cm superior	1.062	1.038	-2.22
Multi-target	4 cm inferior	0.621	0.593	-4.43
C-shape	2.5 cm anterior	2.152	2.131	-0.96
C-shape	1 cm posterior	0.917	0.882	-3.77
Head & neck	Isocenter	2.215	2.265	2.26
Head & neck	5 cm posterior	0.917	0.919	0.27
Prostate	Isocenter	1.817	1.85	1.82
Prostate	4.5 cm posterior	0.372	0.374	0.62

Table 3.3 compares calculated doses from gPENELOPE to experimental measurements. For flat high-dose regions, gPENELOPE gives excellent agreement with measurements (error < 0.31% for non-IMRT plans, and error < 2.26% for IMRT plans). All results for the low-dose points are within the TG-119 confidence limit of 4.5%, with Multi-target (4 cm inferior) and C-shape (1 cm posterior) yielding the largest discrepancies. By examining the dose distributions, we find that the two points are located in high-gradient regions (c.f. **Figure 3.9(e), (f)**) where small chamber positioning error could induce large measurement difference. For these two cases, we use DTA instead to evaluate gPENELOPE performance where we search around the dose matrix grid with interpolation to find the nearest point that has the exact same dose as measurement, with DTA

defined as the distance from this point to the measurement point. Calculated DTAs are less than half of the voxel size (0.91 and 1.42 mm, respectively).

### TG-119 film dose

For the 6 plans listed above, radiographic film measurements were made at the isocenter parallel to the coronal plane. Films were digitized and then exported to perform gamma analysis using gamma parameters recommended by TG-119: (a) absolute dose comparison, (b) 3% dose difference threshold, (c) global normalization for percent dose difference, (d) 3 mm DTA threshold, and (e) 10% low dose threshold. The gamma passing rates are 100.0%, 96.2%, 95.5%, 97.7%, 99.9% and 94.4% for AP-PA, Bands, C-shape, Head & neck, Multi-target and Prostate cases respectively, yielding a mean value of  $97.3\% \pm 2.3\%$  (1 SD), which is within the TG-119 recommended confidence limit of 88%. **Figure 3.9** summarizes the agreement between gPENELOPE and experiment using isodose line overlay. For IMRT plans, only the low dose contours (around 10%) show relatively obvious disagreements.



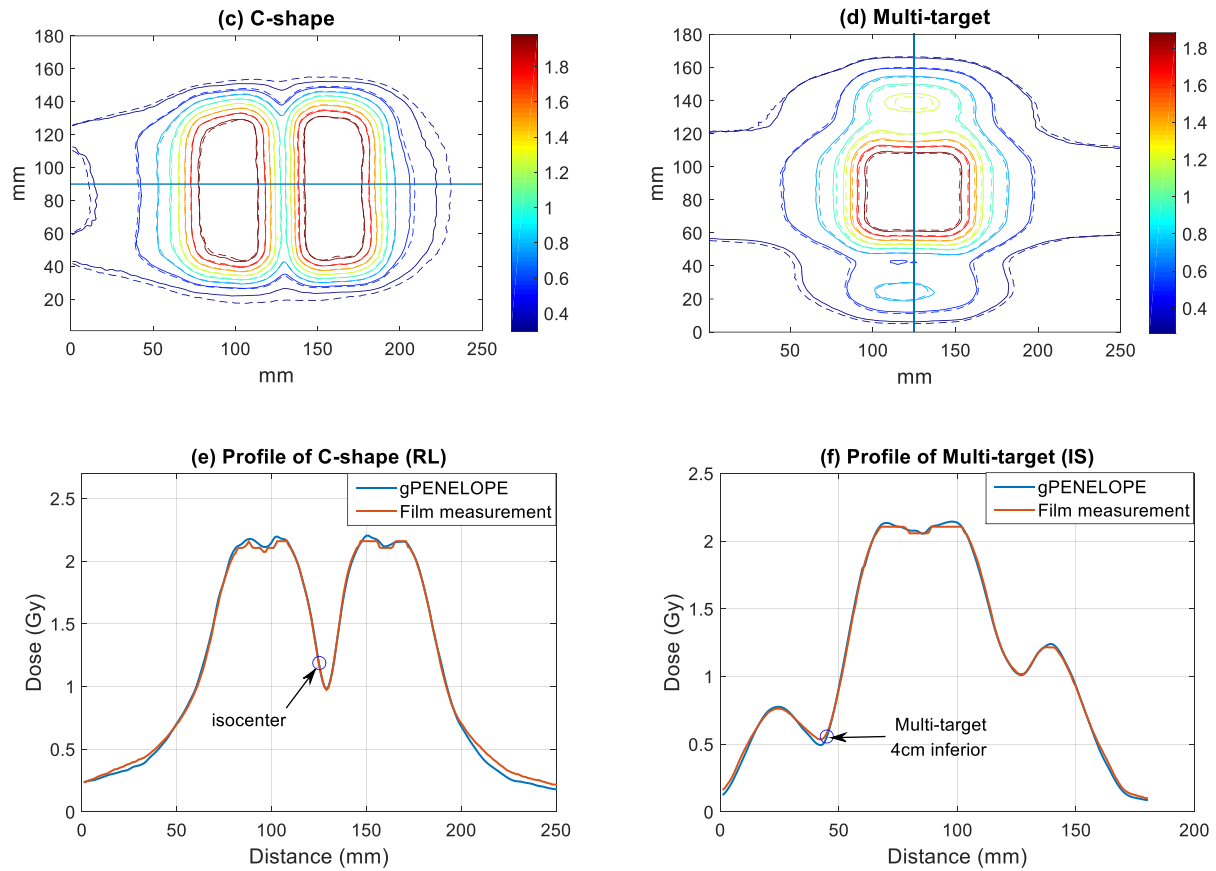
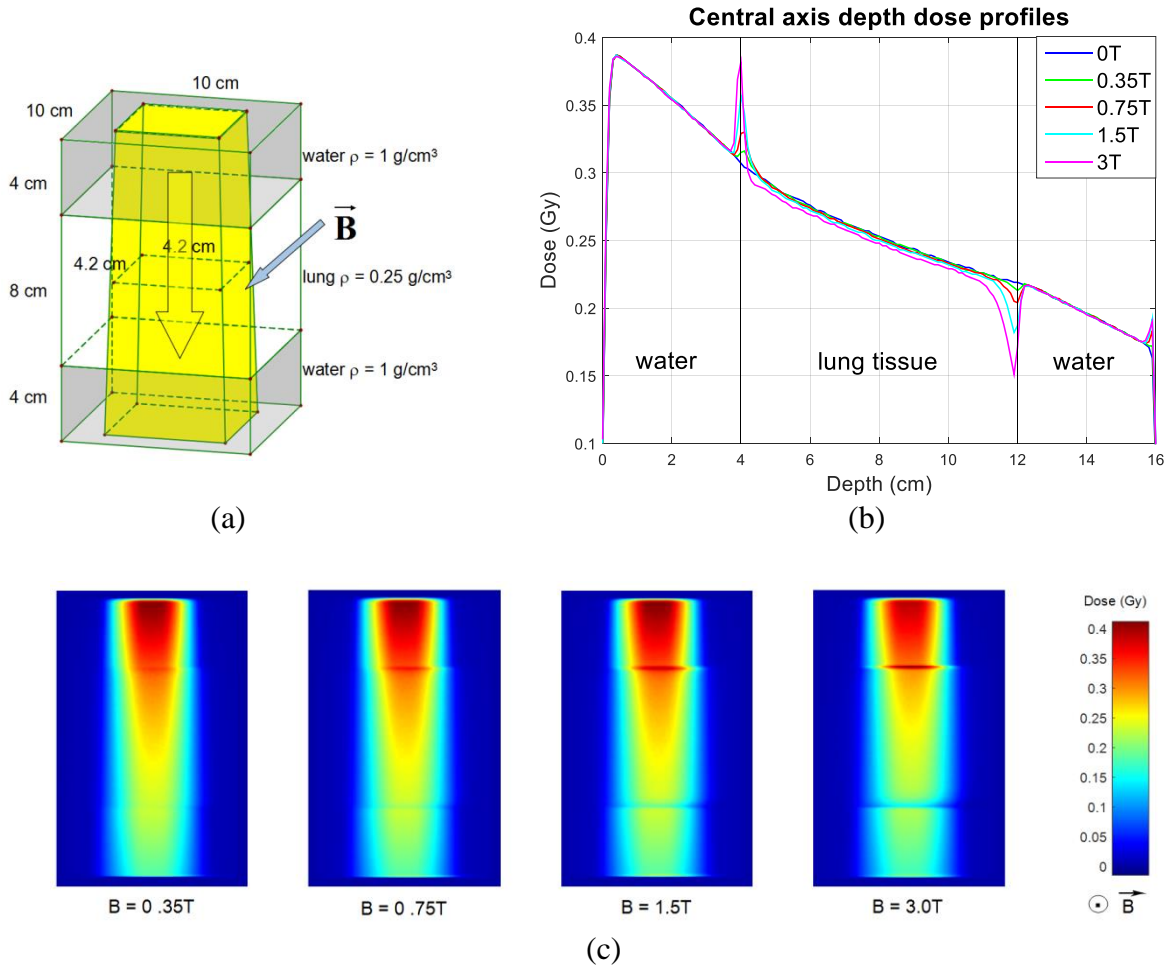


Figure 3.9 Isodose and profile comparison between gPENELOPE and radiographic film measurement: (a) AP-PA, (b) Prostate, (c) C-shape, (d) Multi-target, where solid lines represent gPENELOPE and dashed lines represents film measurement. (e) Profile of C-shape along the right-to-left central axis, (f) Profile of Multi-target along the inferior-to-superior central axis. Note that the circled points are located in the high gradient region.

### 3.8 Application 2: Magnetic effect on MRIdian

By integrating an MR scanner into the radiation delivery system, the MRIdian system must consider magnetic field effects on dose distributions. Raaijmakers [43] performed a detailed simulation study using Geant4 of magnetic field effects on dose distributions for a 6 MV LINAC beam. Although the field strength of the MR scanner on MRIdian is relatively weak (0.35 T), the

electron return effect (ERE) might still be non-trivial because the lower energy of a primary photon from the  $^{60}\text{Co}$  source tends to result in a smaller spiral radius.



**Figure 3.10** Water-lung-water phantom,  $^{60}\text{Co}$  beam and magnetic field configuration. Voxel size is  $1 \times 1 \times 1 \text{ mm}^3$ . (b) Central axis depth dose profiles. Larger magnetic field results in larger dose distortion. (c) Dose distributions for configuration in (a) at indicated magnetic field strengths.

In homogeneous phantoms, dose distortion caused by ERE is generally negligible; however it will become apparent in heterogeneous phantoms at the interfaces. Here we simulate the radiation delivery for a  $10 \times 10 \times 16 \text{ cm}^3$  water-lung-water phantom, where the lung tissue is represented by an 8 cm slab of water with a density of  $0.25 \text{ g/cm}^3$  (Figure 3.10 (a)). A  $4.2 \times 4.2 \text{ cm}^2$   $^{60}\text{Co}$  beam consisting of  $10^9$  photons was incident on the phantom, and a small dose scoring

voxel size was set to  $1 \times 1 \times 1 \text{ mm}^3$  to probe for dose distortion. The simulation was repeated with 0.35, 0.75, 1.5 and 3 T magnetic field strengths and the corresponding central axis depth dose profiles were compared as shown in Figure 3.10 (b). The results are similar to those of Raaijmakers except that the distortion layer is much thinner than for the 6 MV LINAC beam. The dose wash images in the x-z plane are presented in Figure 3.10 (c). Besides stronger dose accumulation effect, the lateral dose shift will also become more obvious as the magnetic field strength goes up.

The simulation suggests that the 0.35 T magnetic field has a minor effect on the dose distribution in a heterogeneous phantom (spike-shape dose accumulation  $< 3\%$  of max dose within a 3 mm thin layer accompanied by a 1 mm lateral shift), which is consistent with Wooten *et al.* [44]’s experimental results using radiographic film. Wooten *et al.* noted that such perturbation effect would be mitigated by multiple overlapping beams, as in the case of an IMRT plan for instance. It is interesting that this effect would become almost imperceptible when the voxel size increases to  $3 \times 3 \times 3 \text{ mm}^3$ , which is the voxel size that most clinics use in treatment planning.

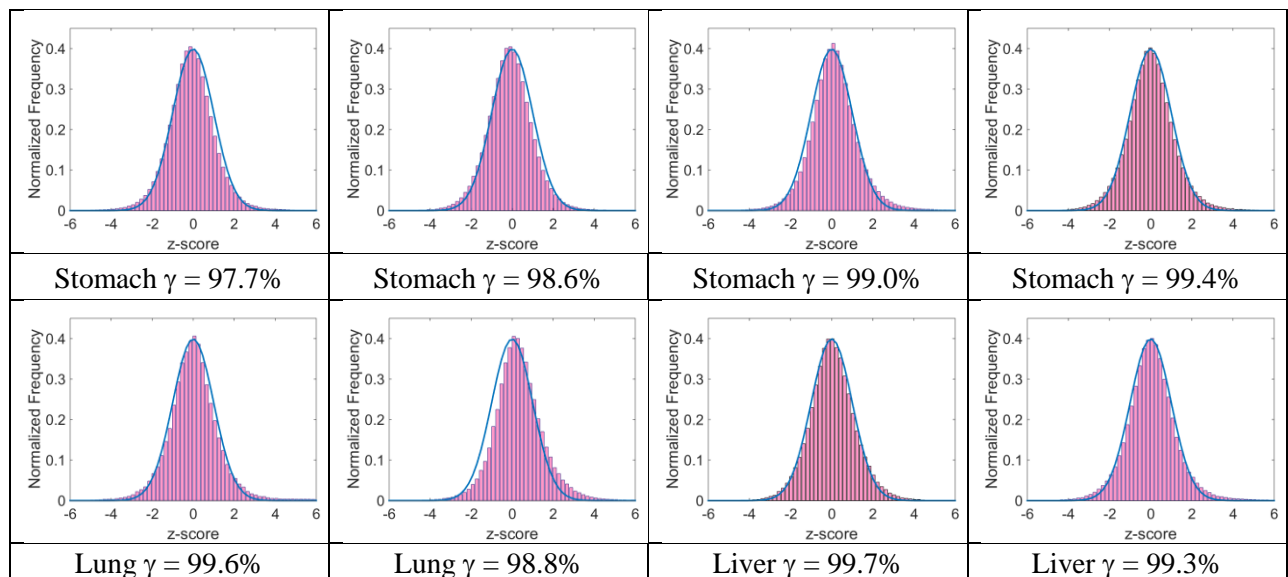
### **3.9 Application 3: validate MRIdian’s treatment plans**

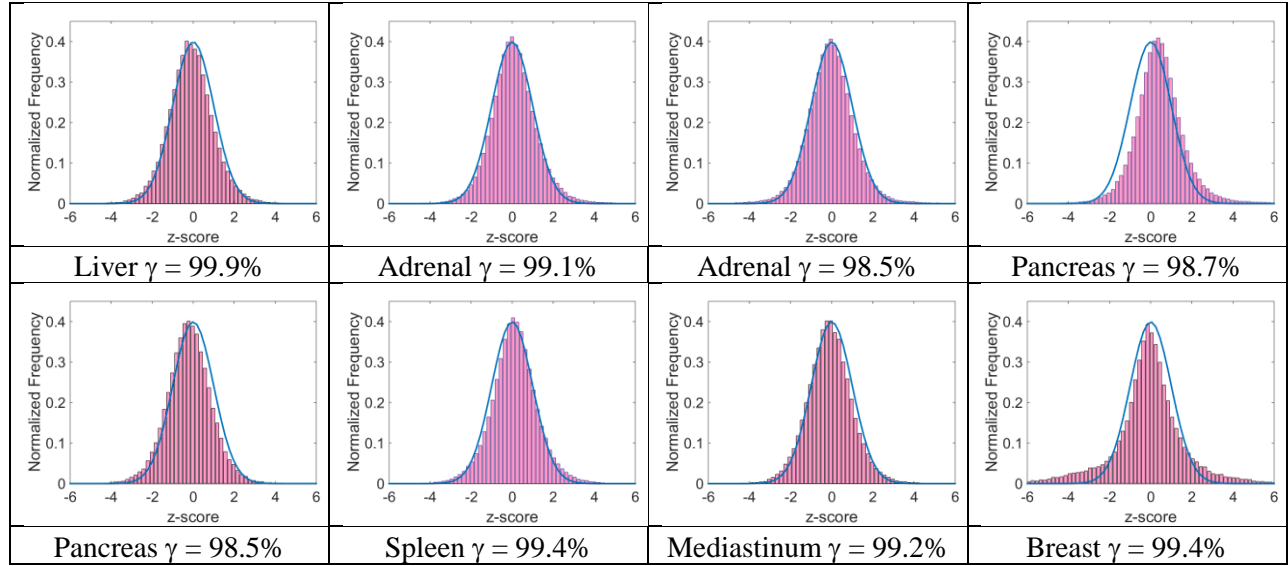
The KMC algorithm on the MRIdian TPS adopted many approximations and variance reductions [39] to increase calculation speed. KMC’s accuracy should thus be confirmed using a third-party Monte Carlo system devoid of approximations through 3D dose comparisons. Thus we selected 16 recent patient plans (from the ViewRay patient registry at Washington University in St. Louis) created by the MRIdian TPS with treatment sites including stomach (4), lung (2), liver (3), adrenal gland (2), pancreas (2), spleen (1), mediastinum (1), and breast (1). 3D gamma analysis results (2%/2 mm DTA and 10% threshold criteria) and histograms of z-scores (in comparison to



standard normal distributions) are listed in **Table 3.4**. The table shows that KMC matches gPENELOPE well (15 out of 16 plans with dose gamma passing rates  $\geq 98\%$  and most closely fitting Gaussian distributions) except that KMC occasionally tends to result in a little higher dose than gPENELOPE. Some z-score distributions (second lung case, first pancreas case, and the breast case) are noticeably offset from the standard distribution, thus indicating that the physical modeling is somewhat affected by the approximations and variance reductions implemented by KMC for calculating a complex  $^{60}\text{Co}$  IMRT plan. The statistical gamma passing rates are as high as  $99.1\% \pm 0.6\%$  for the two dose distributions, proving that KMC generally predicts dose consistent with our “accuracy-oriented” Monte Carlo engine. During MRIdian’s commissioning, Wooten *et al.* designed a custom heterogeneity phantom to acquire ionization chamber measurements [44]. They report that the mean ionization chamber measured dose for 27 measurements for 5 plans is within 1% vs. KMC.

Table 3.4 Gamma passing rates (2%/2 mm and 10% threshold) and z-scores distributions comparing gPENELOPE and MRIdian’s KMC





### 3.10 Discussion and conclusion

The recent clinical use of the MRIdian radiation therapy system represents a significant advance in cancer care, enabling clinicians, for the first time, to deliver highly conformal IMRT with real-time MRI guidance. However, the rapid advances in the technology to deliver such radiation treatments seem to have not been paralleled by corresponding advances in the ability to verify these treatments subject to a permanent magnetic field. For conventional IMRT, despite its widespread utilization at modern radiation therapy clinics, precise dosimetric commissioning remains a challenge [45]. In the era of MRI- guided IMRT, the permanent magnetic field is augmenting another dimension of error and uncertainty to the already error-prone IMRT process.

As a result of many limitations to experimental approaches, largely due to the dearth of appropriate multidimensional water-equivalent dosimeters, a hybrid approach that includes a computational component is needed for MRI-IMRT commissioning and validation. For example, Ding *et al.* [46] studied the feasibility of using a Monte Carlo method to commission stereotactic radiosurgery beams shaped by micro multi-leaf collimators. This hybrid approach is especially

valuable for MRI-IMRT where the Monte-Carlo method may be the only method that is capable of dealing with complex dose deposition in a heterogeneous medium subject to a magnetic field [47, 48]. The Monte-Carlo methods like KMC, on the other hand, may require many approximations in order to be practical in the clinic, and these approximations may not be thoroughly communicated to an end-user for proprietary reasons. We therefore developed a fast, GPU-accelerated Monte Carlo dose calculation system based on PENELOPE. Unlike some other GPU implementations, the accuracy of our adaptation is at the same level as the original code. Our implementation achieved 80 times faster speed than the original PENELOPE implementation. Furthermore, we integrated the  $^{60}\text{Co}$  head model of the MRIdian system into our system and performed a series of experimental benchmarks to examine the accuracy of the entire system. Finally, when comparing to MRIdian's KMC for a number of patients that span multiple disease sites, an average of  $99.1\% \pm 0.6\%$  gamma passing rates at 2%/2 mm provides another layer of confidence in treating patients that may benefit from IMRT with simultaneous MRI guidance.

In the clinic, gPENELOPE should be applicable to nearly any application requiring high dose accuracy, such as beam modeling [49], IMRT optimization [50], dosimeter response modelling [51, 52], dose validation [53], dose accumulation [54] among others. As an example, due to the three-source nature of the MRIdian system, quasi-3D dosimeters, such as ArcCHECK (Sun Nuclear Corp., Melbourne, FL), are quite useful for dosimetry measurements. However, the combined field size dependence and angular dependence of an ArcCHECK have been reported to be on the order of 10-15% for a LINAC delivery. This can be corrected by using look-up tables as a function of beam angle and field size, for which the beam angle must first be determined using a virtual inclinometer in the ArcCHECK software. However, this cannot be

corrected for the MRIdian system due to the simultaneous delivery of all three sources. One possibility to solve this problem is to model the dosimeter response using gPENELOPE so that the radiation transport in the diodes and surrounding buildup/backscatter material can be explicitly simulated. As a result, dose to individual diodes instead of to water can be calculated and subsequently compared to diode's raw response during measurements. By doing this, we can not only convert the ArcCHECK from a relative, 3D dosimeter to an absolute one; more importantly, tighter criteria can be used for the gamma analysis, for example, 2%/2 mm. Nelms *et al.* [55] have recently made a convincing case that adoption of more sensitive metrics/tighter tolerances enables continual improvement of the accuracy of radiation therapy dose delivery not only at the end-user level, but also at the level of product design by the manufacturer. This is especially important for MRI-guided IMRT which is at the early stage of its clinical implementation.

In conclusion, a GPU version of PENELOPE has been developed with its accuracy completely faithful to the original code. The comparisons with MRIdian dose calculation results suggest that MRIdian's fast dose calculation for the  $^{60}\text{Co}$  source subject to a 0.35 T magnetic field is accurate using 2%/2 mm criteria. gPENELOPE will be useful for many MRI-IMRT applications including dose validation and accumulation, IMRT optimization, and dosimetry system modeling.

# Chapter 4: Transport simplification & variance reduction

In chapter 3, we deployed GPU parallelization to build a fast Monte Carlo dose calculation engine, i.e. gPENELOPE. The advantage of gPENELOPE is that it runs 5 times faster than PENELOPE without compromising any accuracy. However, this engine is not fast enough for many time sensitive applications, such as the QA of online adaptive radiation therapy (ART), motion dose accumulation, etc. In this chapter, we will apply transport simplification and variance reduction all together to build an extremely fast MC engine, gDPMvr, and discuss its application to the online ART.

## 4.1 Introduction of online ART

Online adaptive radiation therapy (ART) enables treatment adjustment based on on-board imaging immediately before treatment delivery to account for physical or functional changes to the target volume and organs at risk [47, 56]. However, with the patient on the treatment couch it is not feasible to validate the newly created intensity modulated radiation therapy (IMRT) plan using conventional patient-specific quality assurance (QA) approaches that rely heavily on comparison between planned dose and dose measured in a physical phantom. An alternative approach is performing a second dose calculation for the plan with an independently commissioned dose calculation engine to verify the dose distribution provided by the online treatment planning system [57]. A fast Monte Carlo platform for independently verifying dose distributions in MRI-guided ART is preferable to accurately simulate charged particle transport in external magnetic fields [43]. Venerable Monte Carlo simulation packages such as GEANT4 [58], EGS4/EGSnrc [59], and PENELOPE [60] have been demonstrated to agree excellently with experimental data under a wide range of conditions, but these packages typically require

many hours or even days to achieve an adequate statistical uncertainty (e.g. 1%), which is far beyond the time constraint imposed by an online adaptive scheme. Ideally we need a fast Monte Carlo engine that can complete a 3D dose calculation in a few minutes while maintaining sufficiently high accuracy.

Three approaches have been considered for improving Monte Carlo calculation efficiency [4] including: (1) simplifying particle transport mechanisms, thus reducing the necessary time for each particle history [19], (2) using variance reduction techniques such as particle splitting, Russian roulette, and interaction forcing to reduce the total history number required to achieve a given uncertainty [39], and (3) enhancing computational capability by parallelizing the simulation with multiple CPU or GPU threads [20-23]. Approaches (1) and (2) alter the physical mechanisms of particle transport, and thus may compromise accuracy.

To facilitate computational dosimetry for the MRIdian system (ViewRay, Inc., Cleveland, OH), the only MRI-guided radiotherapy (MRgRT) system currently in clinical use, we recently developed and experimentally validated a GPU-accelerated Monte Carlo dose calculation package called gPENELOPE based on PENELOPE that employs approach (3) only [61] [62]. This package can simulate a tri-<sup>60</sup>Co IMRT plan subject to a 0.35 T magnetic field in about one hour with less than 1% average local uncertainty in a volume where the dose is greater than 10% of the dose maximum [61]. While substantially faster than PENELOPE, gPENELOPE is still not fast enough for online adaptive plan verification.

An alternative fast Monte Carlo code is the Dose Planning Method (DPM) [19] that employs a simplified coupled electron–photon transport scheme in order to achieve high computational performance. More recently, DPM was accelerated via a GPU implementation,

gDPM [20]. According to our benchmarks, the mean time for calculating dose to  $<1\%$  local uncertainty is 13.8 minutes using gDPM for 18 clinical MRIdian IMRT plans. While substantially faster than gPENELOPE, the median time for online re-adaptation measured at our institution to date is 26 minutes [56]. A faster dose calculation platform – likely with additional simplifications – is thus required for consideration in our ART protocol.

In this study, we incorporate the vendor-provided MRIdian head model into DPM and allow for consideration of magnetic fields. We then accelerate the code via GPU implementation, yielding an MRIdian-specific version of gDPM (referred to as gDPM for simplicity in this manuscript). In addition to GPU acceleration (i.e. gDPM), we further accelerate gDPM by introducing 1) variance reduction techniques and 2) additional physical simplifications enabled by details of the MRIdian platform to enable fast and accurate Monte Carlo dose calculation for online ART. We present detailed comparisons of the resulting code, gDPMvr, against gPENELOPE and gDPM in a variety of phantoms to demonstrate that gDPMvr achieves the required calculation efficiency for ART while maintaining sufficient accuracy to engender physicists’ confidence in adaptive plan QA.

## **4.2 Transport simplification**

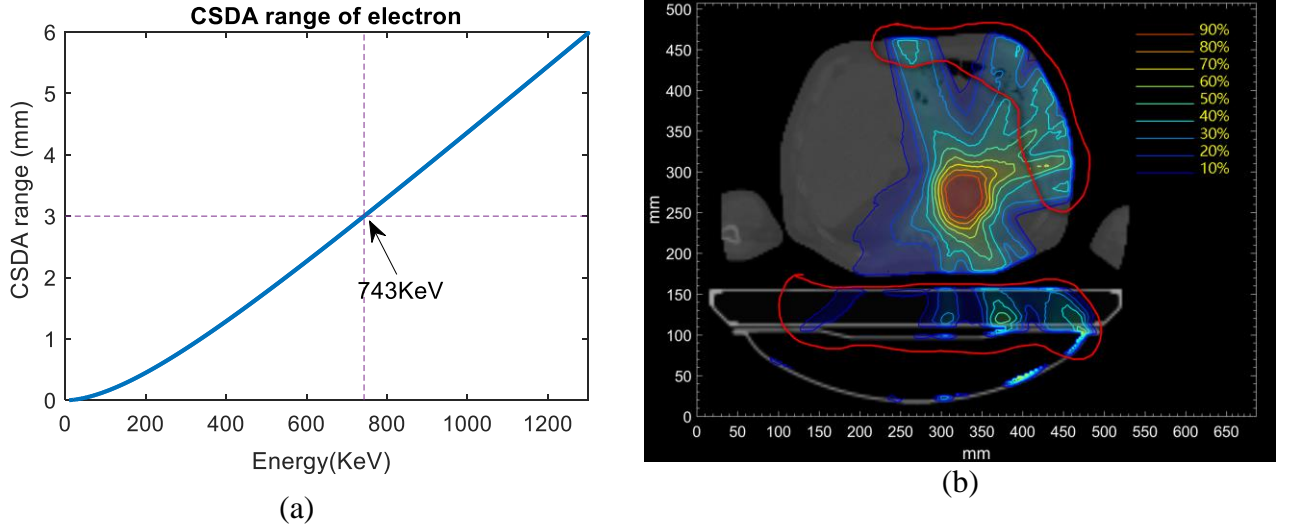
Compared to the general-purpose package PENELOPE, DPM includes a number of simplifications and optimizations for dose calculation in a patient [19]. (1) DPM covers a narrower range of energy so all relevant cross-sections can be obtained using spline or linear interpolation instead of having to perform interpolation on a logarithmic scale, thus enabling faster data access. (2) DPM ignores several types of interaction with either low probability of occurrence in the energy range of interest or little impact on the final dose, such as Rayleigh scattering and inner shell ionization. (3) DPM modifies the pair-production mechanism by

tracking the positron as an electron and emitting two annihilation photons in randomly selected opposite directions to compensate for the latent energy from the positron. This simplification reduces the data table and code length by almost a third. (4) DPM uses simpler physics models to describe scattering events. Taking Compton scattering as an example, DPM applies the Klein-Nishina formula [63] which treats the electrons as free and at rest despite the fact that electrons are bound to atoms in a shell structure with specific energies. (5) DPM uses a random energy hinge for multiple electron scattering instead of a random step hinge used in PENELOPE. This modification enables larger path lengths in multiple scattering events, thereby improving simulation efficiency.

Like PENELOPE, DPM uses a “mixed” scheme for electron transport by treating large energy transfer collisions in an analogue sense and using the continuous slowing down approximation (CSDA) to model small-loss collisions. For online adaptive treatments on the MRIdian system, two facts can be exploited to simplify this scheme. First,  $^{60}\text{Co}$  emits two gamma rays with energies of just 1.17 and 1.33 MeV, much lower than the photon energy generated by a typical LINAC. Second, our clinic uses a voxel size of  $3\times 3\times 3\text{ mm}^3$  for treatment planning. **Figure 4.1** (a) shows how an electron’s CSDA range varies with energy in water. For gamma rays emitted by  $^{60}\text{Co}$  decay, secondary electrons will never travel a distance of more than two voxels. Our simulation profiles with gPENELOPE also show that approximately 80% of secondary electrons have energy less than 743 keV, i.e. the threshold energy for travelling one voxel distance. The remaining secondary electrons are mostly generated around the beam entrance, which is usually far away from the target region as shown in **Figure 4.1** (b). In other words, most electrons will exhaust all their energy in their voxel-of-origin and neighboring voxel, and so detailed analogue simulation is hardly necessary for determining in which voxel an



electron deposits its energy. We therefore can just apply the CSDA approximation in order to greatly simplify the code at the expense of minor accuracy loss. Since the CSDA implementation is simple and fast, we can process secondary electrons immediately in each photon event, instead of storing them in stacks for subsequent processing. Likewise, the 1.33 MeV maximum photon energy allows for a single pair-production event at most per history, and so similarly no stack structure is required. Eliminating stack requirements mitigates the thread divergence phenomenon on GPUs, and hence improves execution efficiency.



**Figure 4.1** (a) CSDA range of an electron vs. kinetic energy in water. (b) Dose distribution calculated for a clinical IMRT plan. The iso-dose lines are shown relative to the maximum dose. The area enclosed by the red lines indicate the area of energy deposition by photons with energies greater than 743 KeV.

In summary, Compton scattering is modeled via the Klein-Nishina equation, pair production is modeled by sampling energy uniformly between 0 and  $E_p - 2E_s$  where  $E_p$  is the photon energy and  $E_s$  the electron rest mass energy, and the photoelectric effect is modeled by simply changing the particle property label from photon to electron. Woodcock tracking [40] is applied to handle photon transport in a heterogeneous phantom effectively. As mentioned above, only the CSDA

scheme is used to handle electron transport with changes in direction being derived from a random energy hinge. A fixed energy loss segment  $E_d$  (e.g. 200 KeV) is split in two sub-steps [19]

$$E_A = \xi E_d \text{ and } E_B = E_d - E_A, \quad (4.1)$$

where  $\xi$  is a random number distributed uniformly between 0 and 1. The electron will first advance a certain distance that exhausts  $E_A$  energy along the initial direction, then get deflected by multiple-scattering and advance another distance that exhausts  $E_B$  along the new direction. If the electron's initial energy  $E_e < E_d$ , then  $E_d$  is replaced by  $E_e$  in the above equation.

### 4.3 gDPM with variance reduction

To further improve computational efficiency, we applied particle splitting and Russian roulette variance reduction methods proposed by Kawrakow *et al.* [39], which can significantly reduce the necessary number of photon scattering events. Suppose we split the incident photon into  $N_S$  photons. The sampled distance to the interaction site of the  $i$ -th photon can then be set as

$$s_i = -\lambda \log(1 - \frac{\xi + i}{N_S}) \quad (4.1)$$

where  $\xi$  is a uniform random number between 0 and 1 and  $\lambda$  is the mean free path of photon. This equation distributes  $N_S$  photons along the initial trajectory according to a log distribution at the expense of a single random number  $\xi$ . These photons undergo the same interaction with the weight assigned to produced secondary particles reduced to  $1/N_S$ . Only one primary photon is randomly selected to continue its history and its weight is recovered to 1. The idea of this method is to generate  $N_S$  electrons spread along a path for one photon interaction, saving approximately  $(N_S - 1)/N_S$  of the photon simulation time. Since the cost of photon scattering is much more expensive than the concise CSDA model of electron transport, this method significantly

improves overall efficiency. Although the approximation inappropriately assumes that secondary electrons at these sites share the same energy and direction, the defect is blurred by a large history number of random scatterings in the subsequent electron transport. Benchmarks confirm that these variance reduction methods can effectively improve the calculation speed at the expense of minor loss in accuracy

## 4.4 GPU implementation: gDPMvr

In contrast to CPU architecture, the GPU was originally designed for parallel graphic processing where single float precision is sufficient. On modern GPUs, single precision float operation is usually 2-3 times faster than double precision. To maximize performance, we decided to use single precision float numbers throughout our GPU code. Resulting accuracy loss has been proven to be negligible [21].

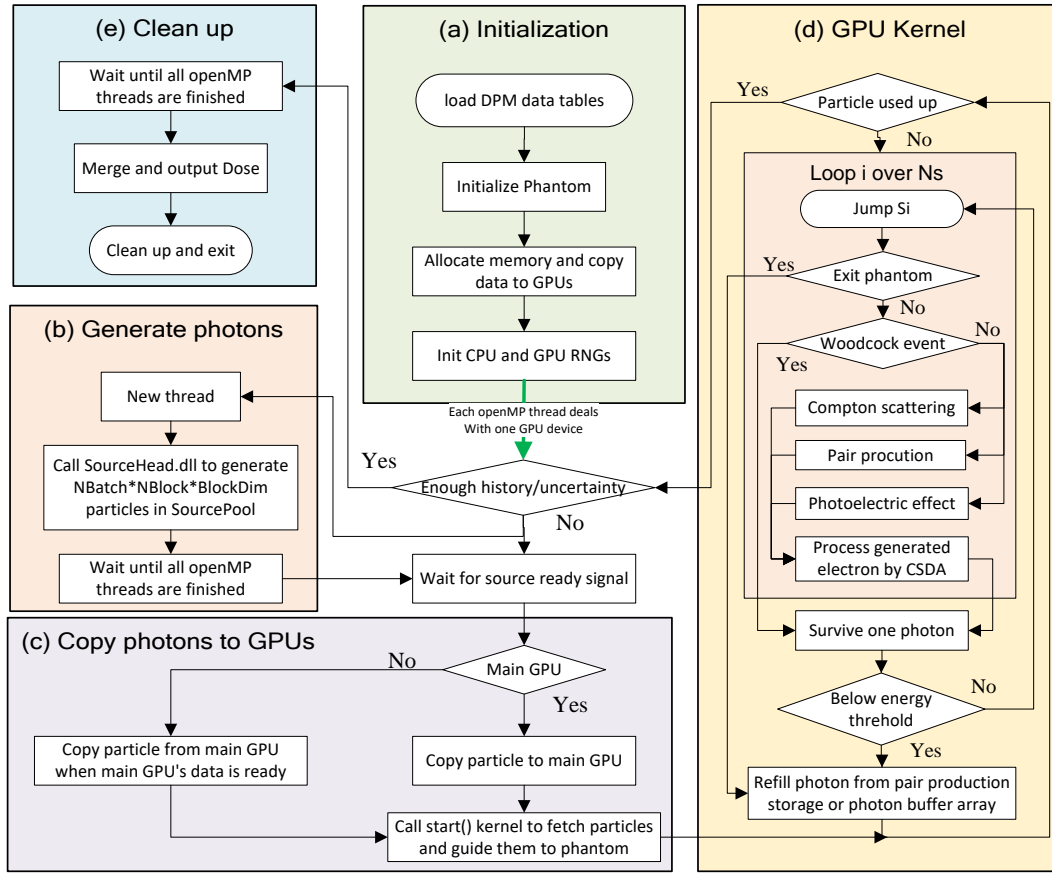
In the original DPM implementation, spline interpolation is employed to calculate cross-sections. Though more accurate, spline interpolation costs four times the memory required by linear interpolation. In GPU architecture, device memory is large but also has a long accessing latency since it is only cached by a small L2 cache. Shared and constant memory are hundreds of times faster than device memory but have limited sizes. As shared memory cannot persist across different thread blocks, the best choice for cross-section tables is constant memory whose size is unfortunately only 64 KB for most Nvidia GPU devices. If spline interpolation were to be used, data tables would need to be loaded to device memory, and performance would be seriously compromised. Therefore, we decided to employ linear interpolation to shrink the data table size and thus utilize constant memory. Our simulation results presented later show marginal effects on final dose.

The workflow of our gDPMvr implementation is shown in Figure 4.2. The program first reads and parses the configuration file which includes details regarding the GPU devices, phantom (geometry and materials), and source head. It then loads the DPM data tables, creates a digital phantom, allocates GPU memories and copies data tables to GPU devices as shown in Figure 4.2 (a). It also initializes the random number generator for each thread with a unique seed. The K80 GPU card (Nvidia), on which we develop and test gDPMvr, includes two GPU devices. To enable multiple GPU support, our program launches multiple threads through OpenMP to call GPU kernel functions on each device simultaneously.

Meanwhile, the main thread launches another thread calling the vender-provided head source module to prepare one batch of incident photons as shown in Figure 4.2 (b). As specified in Figure 4.2 (c), the photons are first copied to the main GPU and then transferred to other GPUs to save I/O time. The GPU kernel function *start()* is called to guide photons to the phantom via free propagation, a reasonable approximation in air. The simulation kernel thread, as shown in Figure 4.2 (d), will split one incident photon  $N_S$  times in a loop, with each photon jumping distance  $s_i$  given by equation (4.1).

If not exiting the phantom, the program will test whether a Woodcock virtual event is selected. If yes, no scattering will happen. Otherwise, one event among Compton scattering, pair-production and photoelectric interaction will be selected based on their cross sections. Meanwhile, one or two electrons will be generated and then processed by using the CSDA scheme immediately. Since our electron transport model is fairly simple, it will not cause serious thread divergence on GPUs and thus it is not necessary to separate photon and electron transport as in the scheme used by gDPM (requiring  $N_S$  times bigger stacks and causing heavy data access with variance reduction).

If the energy of the surviving photon is above a specified cut-off energy, we repeat the splitting process. Otherwise, we refill the current particle variable either from pair-production storage or the incident photon buffer array, thus improving efficiency by enabling constant renewal of particles on all threads. We collect the dose and calculate the accumulated uncertainty after each batch is finished. When a given number of histories is finished or a specified uncertainty is reached, we merge the dose on each GPU device and free allocated memories as shown in Figure 4.2 (e).



**Figure 4.2** Workflow of gDPMvr including 5 modules: (a) Initialization, (b) Generate photons, (c) Copy photons to GPUs, (d) GPU kernel, and (e) Clean up.

Incident photons are generated by a vendor provided DLL module running on a CPU. In certain cases, the head model lags the GPU and cannot provide new particles at a sufficient rate. Our code provides an option to automatically balance the GPU and CPU workload by reusing certain photons. Simulation comparisons show that the dose differences caused by reusing source particles in “hot areas” ( $D > 0.1D_{max}$ ) are mostly (99.73%) within the targeted 1% uncertainty for a large ( $10^9$ ) history number [61].

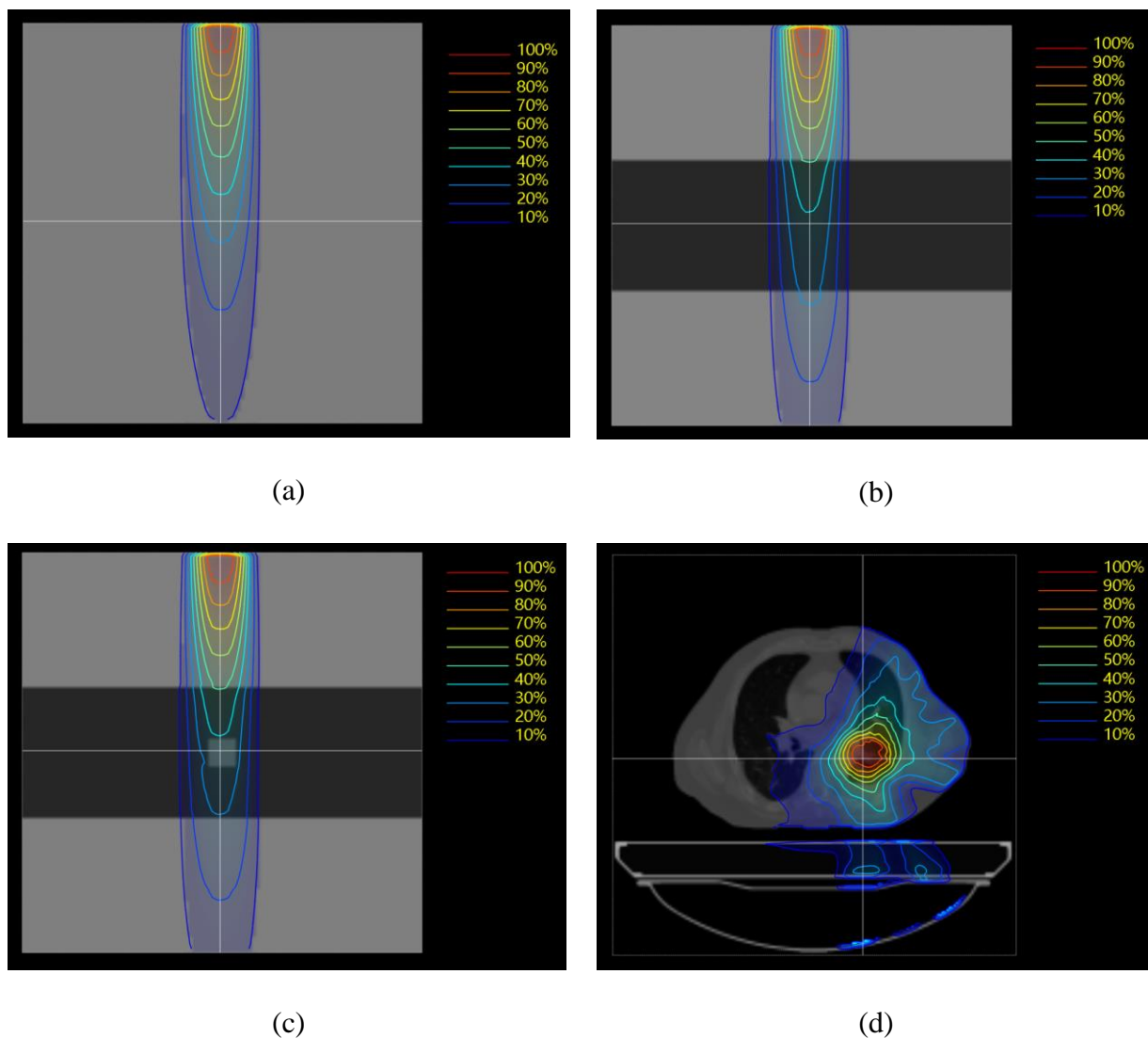
## 4.5 Accuracy and performance benchmarks

Our previously developed dose calculation system gPENELOPE [61] has been validated to be as accurate as the original PENELOPE code with significantly improved efficiency. We thus consider gPENELOPE as the standard throughout these comparisons. To evaluate efficiency and accuracy changes introduced by the variance reduction (rarely used in GPU code), we decided to cross-compare the results of gDPM and gDPMvr as well. For short we denote gPENELOPE as gPEN, the dose differences  $D(\text{gDPM}) - D(\text{gPEN})$  as err1,  $D(\text{gDPMvr}) - D(\text{gPEN})$  as err2, and  $D(\text{gDPMvr}) - D(\text{gDPM})$  as err3 in the following figures. All simulations are performed on a single K80 GPU card (including two units) produced by NVIDIA.

### 4.5.1 Phantoms

As shown in **Figure 4.3**, four different types of phantoms [58] were used to compare the accuracy and performance of the two DPM-based codes, gDPM and gDPMvr, to gPENELOPE. The first three phantoms are all synthetic phantoms sharing the same dimensions of  $30.3 \times 30.3 \times 30.3 \text{ cm}^3$ . The first is a homogeneous water phantom and the second is a slab of uniform lung in water whose density is set to  $0.3 \text{ g/cm}^3$  according to PENELOPE’s material database. The third phantom additionally includes a uniform tumor cube ( $2.1 \times 2.1 \times 2.1 \text{ cm}^3$ ) with a density of  $0.7 \text{ g/cm}^3$ . In addition, we exported 15 patients’ planning CT data from MRIdian’s treatment

planning system, with sites including stomach (4), lung (2), liver (3), adrenal gland (2), pancreas (2), spleen (1), and mediastinum (1). Calculating dose using the four types of objects provides a comprehensive evaluation of how well these algorithms perform in uniform, partially heterogeneous, and highly heterogeneous objects irradiated simultaneously by three  $^{60}\text{Co}$  sources subject to a 0.35 T magnetic field. The voxel size in all cases is set to  $3\times3\times3\text{ mm}^3$ , the same as that we use in the clinic on MRIdian.

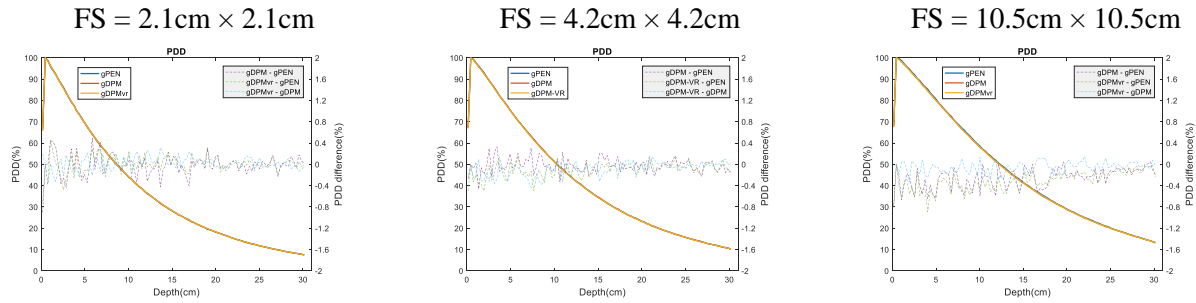


**Figure 4.3** Phantoms used to evaluate accuracy and performance of gDPM, gDPMvr, and gPENELOPE. (a)  $30.3\times30.3\times30.3\text{ cm}^3$  uniform water phantom, (b) water-lung-water phantom where lung's height is 9.9

cm, (c) water-lung-tumor-water phantom where tumor size is  $2.1 \times 2.1 \times 2.1 \text{ cm}^3$ , and (d) patient. Voxel sizes are all set to  $3 \times 3 \times 3 \text{ mm}^3$ .

#### 4.5.2 Homogeneous water phantom

**Figure 4.4** compares vertical and lateral dose profiles in a homogeneous water phantom for three field sizes. All the profiles from the three codes agree with each other to within 0.8%. Figure 4.5 compares the histograms of z-scores. For field sizes of  $2.1 \times 2.1 \text{ cm}^2$  and  $4.2 \times 4.2 \text{ cm}^2$ , the histograms for all three algorithms are close to Gaussian. For a field size of  $10 \times 10 \text{ cm}^2$ , systematic differences are observed: gDPM and gDPMvr score less dose than gPEN, which may be a result of the different ways of handling below-threshold energy photons in DPM (ignore) and PENELOPE (score). For larger field sizes, more energy (with a squared growth rate) is deposited so the difference becomes appreciable. We note that the histograms are normalized by  $\sigma_{tot}$ , so being observable does not imply large absolute differences. In fact the standard deviation of the  $10 \times 10 \text{ cm}^2$  field is actually smaller than that of smaller field sizes. The z-score histograms between gDPM and gDPMvr are always close to Gaussian, indicating that the introduction of variance reduction has negligible effect on accuracy.





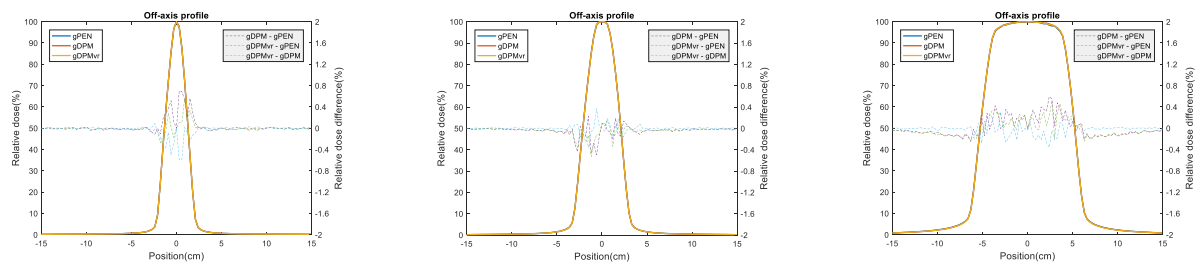


Figure 4.4 Percentage depth dose (upper row) and off-axis profiles (lower row, 5 cm depth) for a homogeneous water phantom. FS: field size.

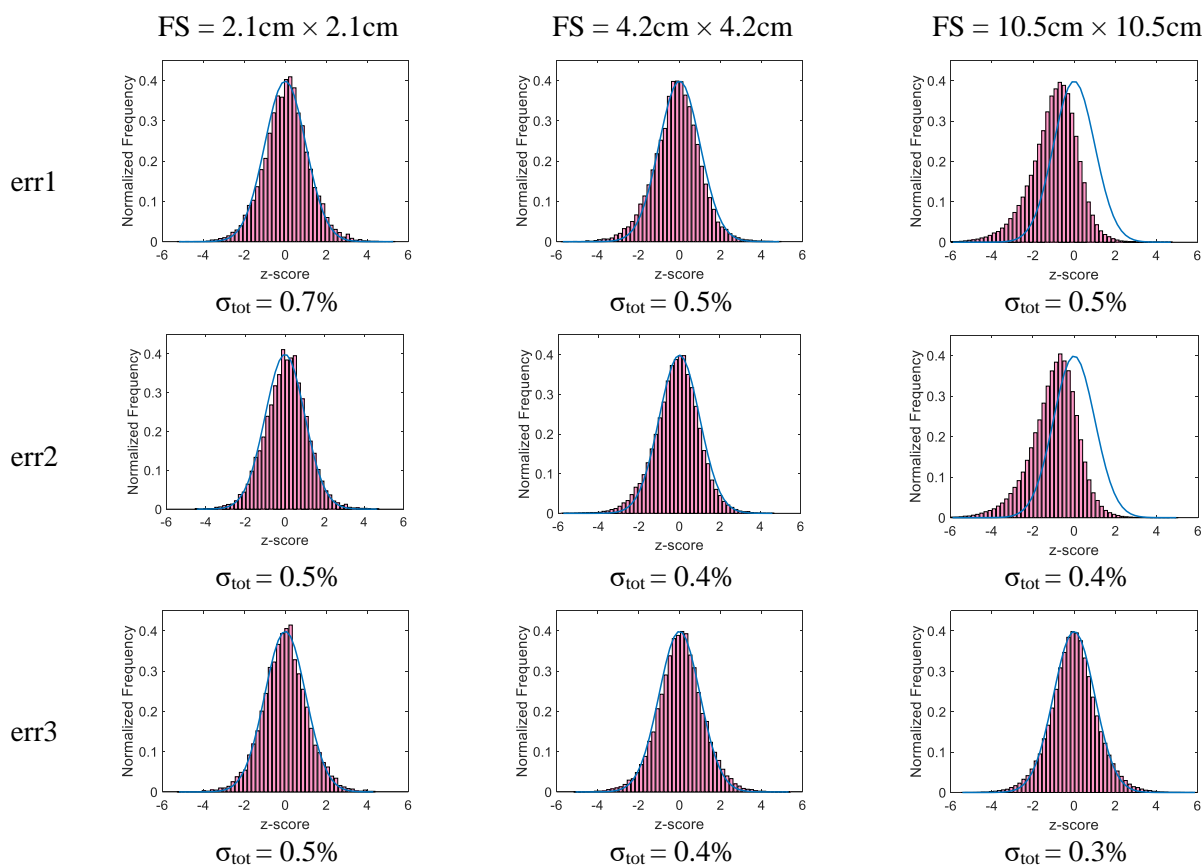


Figure 4.5 z-score histograms among gPEN, gDPM and gDPMvr for a homogeneous water uniform phantom. FS: field size.

Table 4.1 summarizes the performances achieved by the three algorithms. Both gPEN and gDPM ran  $10^9$  histories to achieve less than 0.5% uncertainty, while gDPMvr ran  $10^8$  histories to

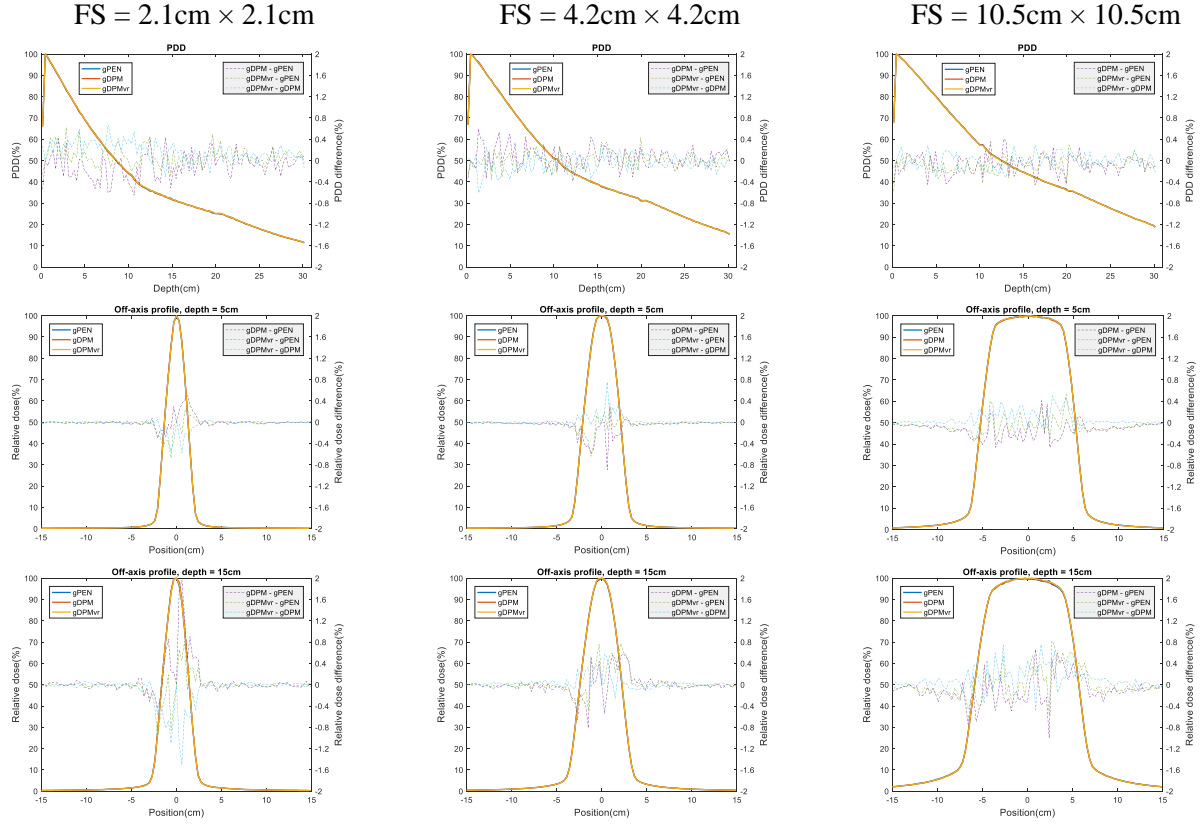
reach an even smaller uncertainty. The mean efficiency ratio, gPEN:gDPM:gDPMvr, is 1:2:66 in a homogenous water phantom, indicating that variance reduction techniques can significantly increase the calculation efficiency.

**Table 4.1** Performance benchmarks in a homogeneous water phantom

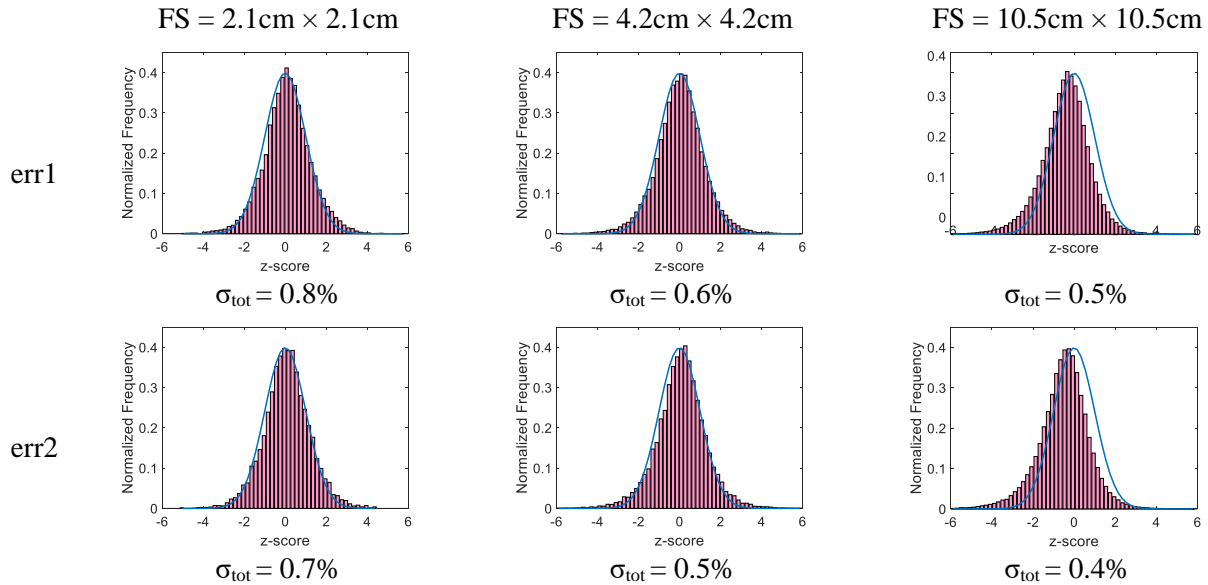
FS/cm <sup>2</sup>	gPEN			gDPM			gDPMvr			Relative $\eta$		
	T(min)	$\sigma$	$\eta$	T(min)	$\sigma$	$\eta$	T(min)	$\sigma$	$\eta$	gPEN	gDPM	gDPMvr
2.1×2.1	30.01	0.52	0.12	28.44	0.5	0.14	3.39	0.23	5.58	1.00	1.14	45.25
4.2×4.2	48.92	0.39	0.13	47.2	0.38	0.15	4.51	0.17	7.67	1.00	1.09	57.09
10.5×10.5	299.47	0.34	0.03	87.51	0.33	0.10	14.3	0.16	2.73	1.00	3.63	94.57
Average										1.00	1.96	65.63

### 4.5.3 Water-lung-water phantom

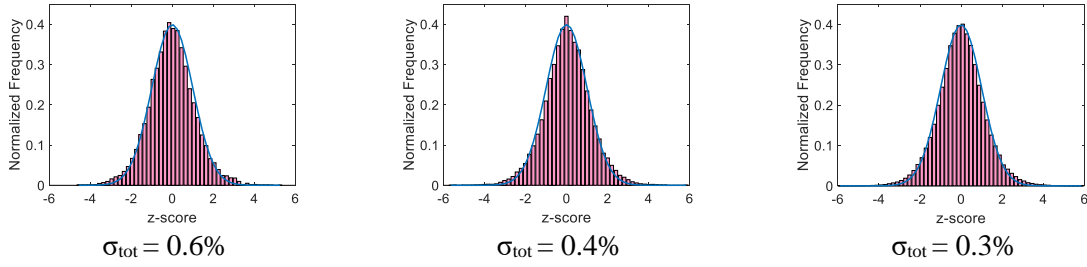
Figure 4.6 shows comparisons of PDD and off-axis profiles in a water-lung-water phantom for three different field sizes. The off-axis profiles at 5 cm depth (inside the water) generated from the three algorithms agree with each other to within 0.8%. However, the off-axis profiles at 15 cm depth (inside the lung slab) show a slightly larger difference of around 1%. The PDD profiles become unsmooth at the water-lung interface due to the electron-return-effect (ERE). The z-score histograms (Figure 4.7) exhibit similar patterns as those shown in **Figure 4.5**, except for slightly more noticeable deviations from being Gaussian with larger  $\sigma_{tot}$  values. The introduction of heterogeneity triggers the ERE, thus augmenting the dose differences between gPEN and gDPM /gDPMvr. Nevertheless, the latter two algorithms still show almost identical statistical behaviors.



**Figure 4.6** Percentage depth dose (upper row) and off-axis profiles (middle row: 5 cm depth and inside the water, lower row: 15 cm depth and inside the lung) for a water-lung-water phantom. FS: field size.



err3



**Figure 4.7** z-score histograms among gPEN, gDPM and gDPMvr for a water-lung-water phantom.

Table 4.2 lists the performances achieved by the three algorithms in the water-lung-water phantom. The history number for gPEN and gDPM remains at  $10^9$  histories while gDPMvr still runs  $10^8$  histories. The mean efficiency ratio, gPEN:gDPM:gDPMvr, remains almost unchanged (1:2:65) in a more heterogeneous phantom.

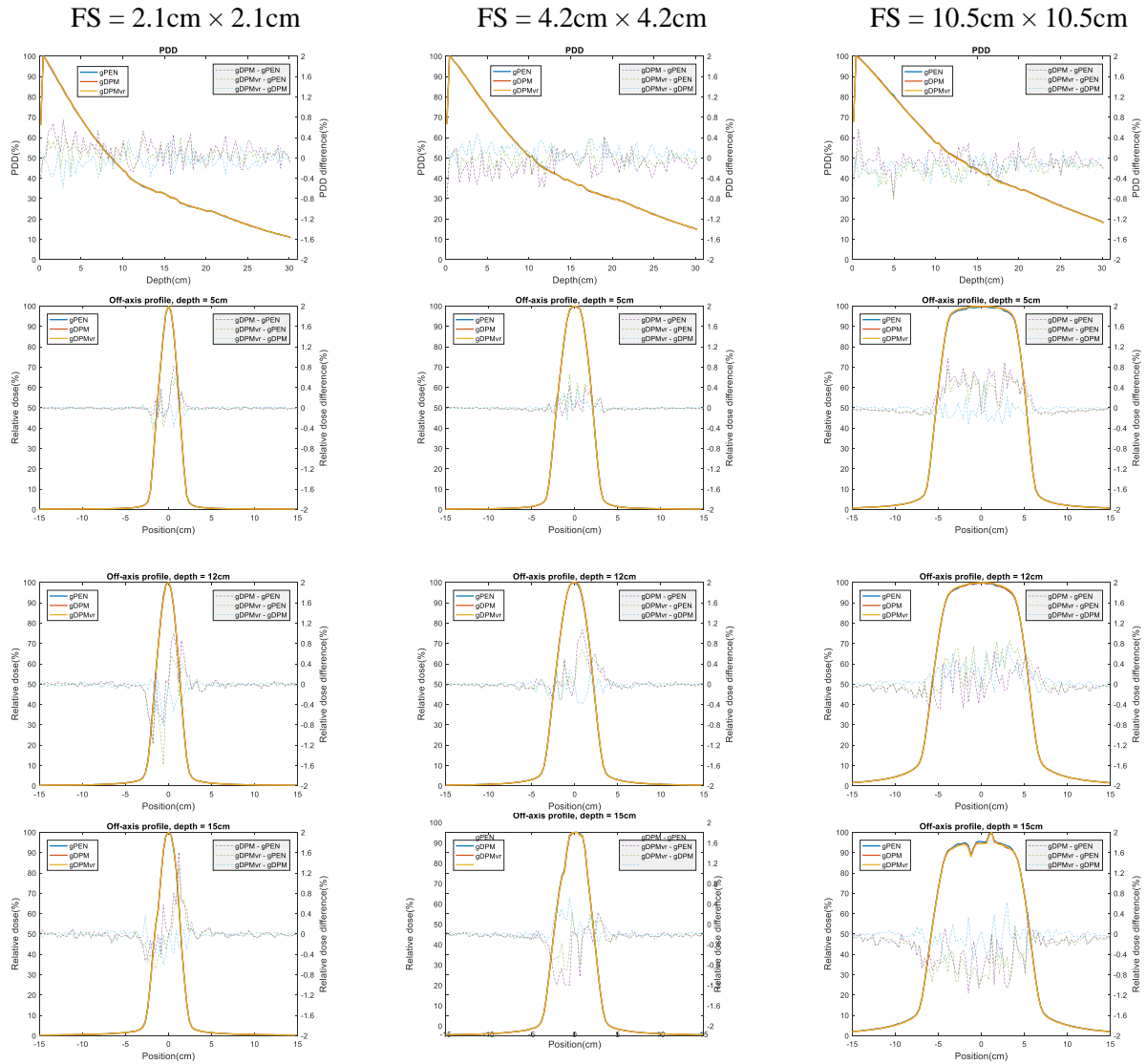
**Table 4.2** Performance benchmarks in a water-lung-water phantom

FS	gPEN			gDPM			gDPMvr			Relative $\eta$		
	T(min)	$\sigma$	$\eta$	T(min)	$\sigma$	$\eta$	T(min)	$\sigma$	$\eta$	gPEN	gDPM	gDPMvr
2.1cm	27.57	0.59	0.10	36.94	0.55	0.09	3.16	0.26	4.68	1	0.86	44.93
4.2cm	45.45	0.45	0.11	42.53	0.42	0.13	3.99	0.2	6.277	1	1.23	57.67
10.5cm	276.42	0.37	0.03	81.26	0.35	0.10	15.9	0.16	2.46	1	3.80	92.97
Average										1	1.96	65.19

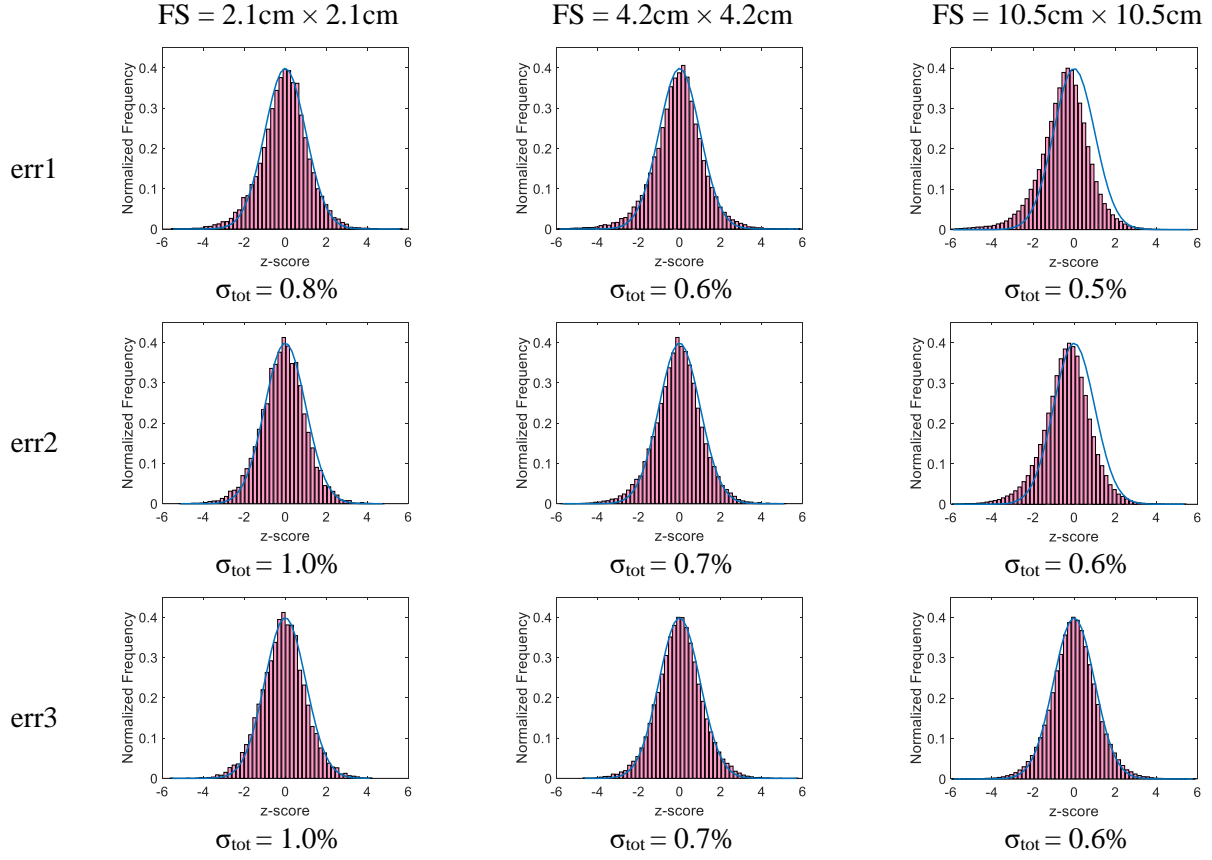
#### 4.5.4 Water-lung-tumor-water phantom

Figure 4.8 shows profile comparisons in the water-lung-tumor-water phantom. With the introduction of a tumor cube, the PDDs and off-axis profiles at depths of 5 and 12 cm show similar patterns as shown in **Figure 4.6**. However, the off-axis profiles at 15 cm depth traversing both the tumor and the lung show obvious ERE. As shown in Figure 4.9, z-score histogram plots are similar to those shown in **Figure 4.5** and **Figure 4.7** except that the observed systematic

differences for a field size of 10 cm x 10 cm are less severe, which may be explained by the ERE effect better localizing dose deposition, thus offsetting the adverse effects of ignoring the below-threshold energy photons.



**Figure 4.8** Percentage depth dose and off-axis profile for a water-lung-water phantom. First row: percentage depth dose. Second to fourth rows: off-axis profiles at depth of 5, 12 and 15 cm, i.e. in the water, the lung, the tumor respectively. FS: field size.



**Figure 4.9** z-score histograms among gPEN, gDPM and gDPMvr for a water-lung-tumor-water phantom

Table 4.3 summarizes the performances achieved by the three algorithms in the water-lung-tumor-water phantom. The history numbers remain the same as above and a similar efficiency ratio, gPEN:gDPM:gDPMvr, is observed (1:2:58), although gDPMvr's efficiency decreases in the more heterogeneous phantom.

**Table 4.3** Performance benchmarks in a water-lung-tumor-water phantom

FS	gPEN			gDPM			gDPMvr			Relative $\eta$		
	T/min	$\sigma$	$\eta$	T/min	$\sigma$	$\eta$	T/min	$\sigma$	$\eta$	gPEN	gDPM	gDPMvr
2.1cm	27.63	0.59	0.10	27.3	0.56	0.12	3.56	0.26	4.16	1	1.12	39.97
4.2cm	47.09	0.45	0.10	47.46	0.42	0.12	4.43	0.2	5.64	1	1.14	53.81
10.5cm	281.37	0.37	0.03	85.51	0.35	0.10	18.56	0.16	2.10	1	3.68	81.07

Average										1	1.98	58.28
---------	--	--	--	--	--	--	--	--	--	---	------	-------

#### 4.5.4 Clinical patients

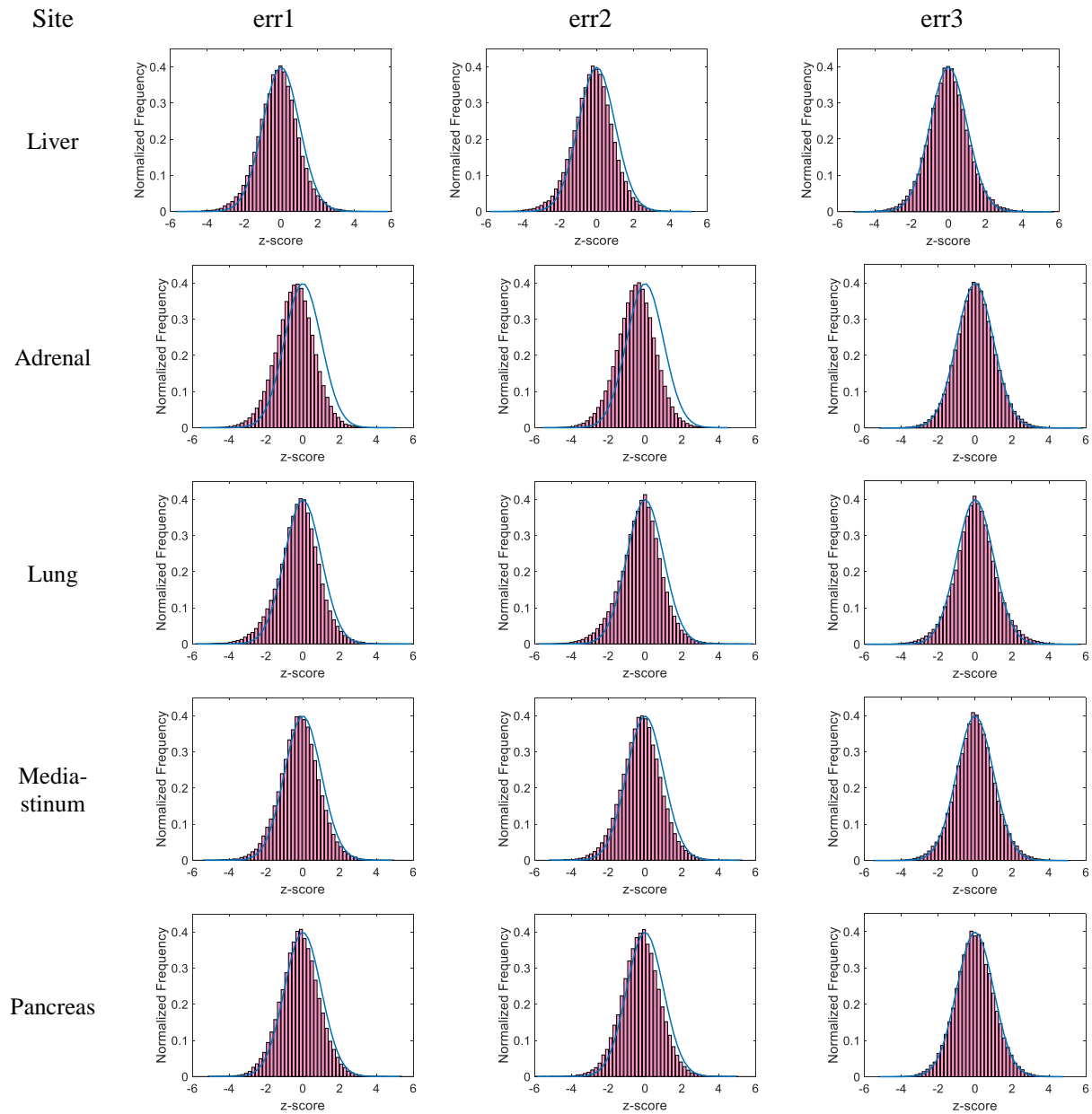
Table 4.4 shows the gamma passing rates with strict criteria ( $DTA = 0$  mm,  $\Delta D = 0.5\%D_{max}$ ,  $D > 0.1D_{max}$  threshold) and standard deviations of relative differences among gPEN, gDPM and gDPMvr for 15 IMRT treatment plans. The selected z-score histograms shown in Figure 4.10 reveal systematic differences between gPEN and gDPM/gDPMvr. Nevertheless, the systematic difference is not big with  $\overline{\sigma_{tot}} = 0.9\%$  averaged over all 15 plans. gDPM and gDPMvr, on the other hand, share almost the same statistical behavior. The average gamma passing rate is 98.9% between gPEN and gDPM, 99.4% between gPEN and gDPMvr, and 99.9% between gDPM and gDPMvr. In other words, the chances for dose differences to exceed  $0.5\%D_{max}$  are as small as 1.1% (err1), 0.6% (err2) and 0.1% (err3), respectively.

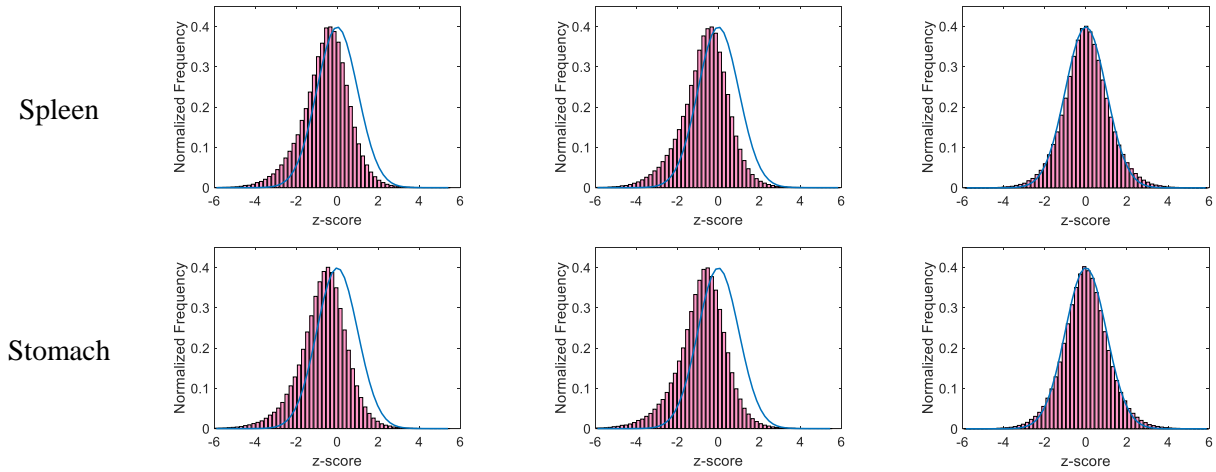
**Table 4.4** Gamma passing rates and standard deviations of relative differences for 15 clinical IMRT plans

Treatment site	err1		err2		err3	
	$\gamma$ (%)	$\sigma_{tot}$ (%)	$\gamma$ (%)	$\sigma_{tot}$ (%)	$\gamma$ (%)	$\sigma_{tot}$ (%)
Liver	99.90	0.9	99.97	0.8	99.99	0.7
Liver	98.39	0.9	99.21	0.8	99.82	0.7
Liver	99.84	1.0	99.94	0.8	99.98	0.8
Adrenal	99.18	0.9	99.52	0.8	99.92	0.8
Adrenal	99.31	1.0	99.63	0.9	99.90	0.9
Lung	99.70	0.8	99.83	0.7	99.97	0.7
Lung	99.77	0.7	99.88	0.7	99.99	0.6
Mediastinum	99.52	1.0	99.73	0.9	99.97	0.9
Pancreas	99.66	0.7	99.78	0.6	99.99	0.6
Pancreas	97.86	1.1	98.82	0.9	99.63	0.9
Spleen	98.82	0.9	99.49	0.8	99.84	0.7
Stomach	97.76	1.0	98.40	0.9	99.63	0.8



Stomach	97.43	1.0	98.56	0.8	99.61	0.8
Stomach	98.62	0.8	99.36	0.7	99.85	0.7
Stomach	98.24	0.9	98.93	0.8	99.70	0.8





**Figure 4.10** Z-score histograms for 7 IMRT plans

Table 4.5 lists the performances achieved by the three algorithms in real patient phantoms. Here we set the termination condition as reaching 1% uncertainty instead of finishing a given history number in order to imitate the real treatment planning system. Moreover, the source particle reuse is toggled on to maximize performance. The mean efficiency ratio of gPEN:gDPM:gDPMvr is about **1:7:43**. That is, in highly heterogeneous phantoms, gDPM suffers less performance loss than gPEN due to its much simplified code. The variance reduction scheme can increase calculation efficiency from gDPM by as much as six-fold on average, with almost the same accelerating factor being achieved from gPEN to gDPM. gDPMvr can finish calculating a treatment plan in **2.3 minutes** on average with only 0.5% accuracy loss compared to the “golden standard” gPEN. Thus, gDPMvr is well-suited to fulfill the purpose of verifying adaptive treatment plans in a fast and accurate way.

**Table 4.5** Performance benchmarks in real patient phantoms

FS	gPEN			gDPM			gDPMvr			Relative $\eta$		
	T(min)	$\sigma$	$\eta$	T(min)	$\sigma$	$\eta$	T(min)	$\sigma$	$\eta$	gPEN	gDPM	gDPMvr
Liver	41.83	1.12	0.019	9.28	1.08	0.092	1.16	1.07	0.753	1.00	4.85	39.51
Liver	86.85	1.45	0.005	17.28	1.2	0.040	2.52	1.15	0.300	1.00	7.34	54.79
Liver	22.49	1.25	0.028	3.71	1.13	0.211	0.61	1.1	1.355	1.00	7.42	47.61
Adrenal	61.09	1.42	0.008	10	1.2	0.069	2.21	1.13	0.354	1.00	8.55	43.65
Adrenal	77.44	1.1	0.011	24.21	1.04	0.038	3.21	1.03	0.294	1.00	3.58	27.52
Lung	37.98	1.41	0.013	8.79	1.15	0.086	1.56	1.09	0.540	1.00	6.50	40.74
Lung	48.88	1.37	0.011	10.06	1.19	0.070	2.08	1.12	0.383	1.00	6.44	35.16
Mediastinum	37.65	1.37	0.014	7	1.19	0.101	1.59	1.15	0.476	1.00	7.13	33.61
Pancreas	33.65	1.9	0.008	5.32	1.58	0.075	1.36	1.46	0.345	1.00	9.15	41.90
Pancreas	38.5	1.37	0.014	7.03	1.2	0.099	1.03	1.17	0.709	1.00	7.14	51.25
Spleen	78.49	1.46	0.006	14.11	1.21	0.048	2.1	1.18	0.342	1.00	8.10	57.22
Stomach	120.73	1.33	0.005	21.49	1.15	0.035	4.99	1.1	0.166	1.00	7.51	35.37
Stomach	125.55	1.37	0.004	24.67	1.15	0.031	3.41	1.12	0.234	1.00	7.22	55.09
Stomach	101.16	1.33	0.006	22.25	1.14	0.035	2.69	1.11	0.302	1.00	6.19	53.99
Stomach	110.13	1.19	0.006	21.92	1.08	0.039	4.15	1.06	0.214	1.00	6.10	33.45
Average	<b>68.2</b>			<b>13.8</b>			<b>2.3</b>			1.00	6.88	43.39

## 4.6 Discussion and conclusion

The recent clinical use of the MRIdian radiation therapy system represents a significant advance in cancer care, enabling clinicians to deliver highly conformal IMRT with real-time MRI guidance. More importantly, the advent of online soft tissue image guidance enables delivery of online adaptive radiation therapy, which is a dramatic departure from conventional treatments employing a single static plan throughout the entire treatment course [56]. In a recent Point/Counterpoint debate [64], it has even been proposed that within the next few years, adaptive hypofractions will become the most common form of radiation therapy. However, this potential paradigm-changing treatment scheme challenges the clinician's ability to assure the safe delivery of the online re-optimized and re-calculated IMRT treatment plans [57],

particularly where the dose deposition is subject to a magnetic field. The magnetic field exerts a force on secondary electrons that complicates dose deposition in highly heterogeneous phantoms such as the human body. Monte Carlo is the preferred form of calculation for achieving adequate uncertainty under these challenging conditions. To facilitate the wide adoption of online adaptive radiation therapy that can benefit many patients, development of tools such as rapid and accurate Monte Carlo dose verification is a pressing requirement [57].

Recently we developed a GPU-accelerated Monte Carlo C++ code based on the venerable PENELOPE system, namely gPENELOPE [61]. In this work, we accelerated Monte Carlo dose calculation with parallel computation while maintaining original accuracy, with the intention of deploying the platform for complementing experimental dosimetry for treatment subject to a permanent magnetic field which is limited by measurement uncertainty, dimensionality and spatial resolution [53]. We validated gPENELOPE according to AAPM TG-105 [4] guidelines by virtue of a number of measurements with both homogenous and heterogeneous phantoms. An acceleration factor of 80 was demonstrated in comparison to the original single-thread FORTRAN implementation with the original accuracy being preserved. Despite this drastic acceleration, the code remains not fast enough for online quality assurance [56].

Recently Acharya *et al.* [56] reported that the median time for online ART including recontouring, re-optimization, and QA is 26 minutes for their institution's first patients treated via online ART, with recontouring being the most time-consuming aspect of the procedure. Any Monte Carlo platform for QA thus should require at most several minutes for completion in order to contribute meaningfully to the ultimate goal of minimizing the time required by the ART workflow. DPM, developed by Sempau *et al.* [19], was a major milestone in the development of fast Monte Carlo code for routine clinical use. DPM significantly accelerates

Monte Carlo simulation largely by simplifying various charged-particle transport mechanisms. Jia *et al.* later introduced gDPM which accelerated DPM through deployment on a GPU [20]. While gDPM substantially accelerates DPM, we found gDPM – adapted to incorporate the MRIdian head model and external magnetic fields – is not adequately fast for implementation in our institution’s ART workflow particularly when considering a threshold of 1% *local* uncertainty (vs. the 1% *global* average uncertainty used in Jia’s work). Achieving 1% local uncertainty is imperative especially in hypofractionated deliveries that include high maximum doses with steep gradients [56, 64]. Without adequate reduction in local uncertainty, poor understanding of dose gradients could have severe clinical consequences such as acute toxicities in normal tissues.

In this study, we built upon gDPM by introducing variance reduction and several system-specific simplifications in order to achieve competitive calculation times for implementation in MRgRT ART. These simplifications stem from the mean photon energy in  $^{60}\text{Co}$  decay, the small magnetic field strength of the imaging system, and the 3 mm voxel size utilized for treatment planning in our clinic. The resulting platform – gDPMvr – increases calculation speed of clinical plans by factors of 43 and 6 relative to gPENELOPE and gDPM respectively while preserving adequate ( $<1\%$ ) statistical uncertainty within regions of dosimetric interest. We demonstrated that gDPMvr can achieve 1% mean *local* uncertainty in the  $D > 0.1D_{max}$  region in 2.3 minutes on average on one Nvidia K80 GPU card for complicated tri- $^{60}\text{Co}$  IMRT plans. ViewRay provides its users with a CPU based Monte Carlo secondary dose calculation engine for online ART plan verification QA. For a typical pancreatic IMRT plan, the computation time for 50 million histories is 18 mins on a Windows 7 PC with an Intel Core i7 3770 processor (3.4 GHz

base frequency, 4 cores and 16 GB RAM). In contrast, the computation time for gDPMvr is 2 mins.

This performance is largely enabled by simplifications of electron transport that facilitate GPU implementations of variance reduction techniques that traditionally suffer from thread divergence and limited register number. In fact, we previously attempted to apply a variance reduction technique proposed by Kawrakow et al. [39] in gPENELOPE, but the implementation actually worsened calculation times. Similar results have been reported on another GPU-based Monte Carlo system, namely GPUMCD [22]. Simplification of electron transport using CSDA only may compromise accuracy at tissue/air interfaces – such as at the bowel, esophagus, and skin – in a magnetic field. One strategy could be to implement a mixed scheme that can be applied in the regions that are of particular concern. Also, a post-treatment offline recalculation using gPENELOPE can be performed if desired for assessing problematic sub-volumes in the online calculation.

With the imminent introduction of MRI-guided LINAC devices [64], integrating variance reduction techniques on GPUs will be a challenging and pressing problem. Moreover, the required accuracy of an online Monte Carlo system and the appropriate QA metrics, e.g. conventional gamma criteria vs. dose-volume-histogram metrics [65], must be established. Nonetheless, gDPMvr achieves speeds beyond those required by in-use workflows for MRgRT ART while preserving accuracy comparable to that achieved by more traditional Monte Carlo code.

# Chapter 5: DVH constraint

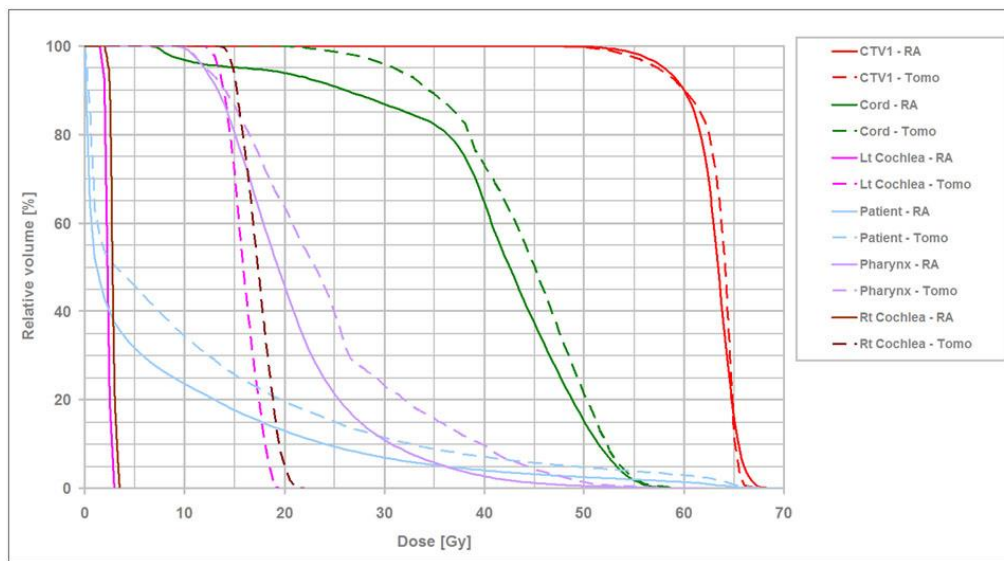
In last chapter, we build a new Monte Carlo radiation simulation engine called gDPMvr, which combines two acceleration methods -- transport simplification and variance reduction. We successfully reduced the in-patient simulation time to 2.3 minutes in average with less 1% accuracy loss. Though impressive the performance of gDPMvr is, further acceleration is possible through focusing the calculation on the important regions which are used to generated DVH curves. We name this acceleration method as “DVH constraint”.

## 5.1 Introduction of DVH

A dose-volume histogram (DVH) is a histogram relating radiation dose to tissue volume in radiation therapy planning (Figure 5.1). DVHs are most commonly used as a plan evaluation tool and to compare doses from different plans or to structures. DVHs were introduced by Michael Goitein [66](who introduced radiation therapy concepts such as the "beam's-eye-view," "digitally reconstructed radiograph," and uncertainty/error in planning and positioning, among others) and Verhey in 1979 [67]. DVH summarizes 3D dose distributions in a graphical 2D format. The "volume" referred to in DVH analysis is a target of radiation treatment, a healthy organ nearby a target, or an arbitrary structure.

DVHs can be visualized in either of two ways: differential DVHs or cumulative DVHs. A DVH is created by first determining the size of the dose bins of the histogram. Bins can be of arbitrary size, e.g. 0–1 Gy, 1.001–2.000 Gy, 2.001–3.000 Gy, etc. In a differential DVH, bar or column height indicates the volume of structure receiving a dose given by the bin. Bin doses are along the horizontal axis, and structure volumes (either percent or absolute volumes) are on the vertical. The differential DVH takes the appearance of a typical histogram. It reads like the volume of the organ that receives the dose of the correspondent dose - bin. It is built by the sum

of the number of voxels characterized by a specified range of dosage for the organ considered. It is helpful in providing information about changes in dose within the structure considered and to easily visualize minimum and maximum dose. The cumulative DVH is plotted with bin doses along the horizontal axis, as well. However, the column height of the first bin (0–1 Gy, e.g.) represents the volume of structure receiving greater than or equal to that dose. The column height of the second bin (1.001–2.000 Gy, e.g.) represents the volume of structure receiving greater than or equal to that dose, etc. With very fine (small) bin sizes, the cumulative DVH takes on the appearance of a smooth line graph. The lines always slope and start from top-left to bottom-right. For a structure receiving a very homogenous dose (100% of the volume receiving exactly 10 Gy, for example) the cumulative DVH will appear as a horizontal line at the top of the graph, at 100% volume as plotted vertically, with a vertical drop at 10 Gy on the horizontal axis.



**Figure 5.1** Cumulative DVHs from a radiotherapy plan



A DVH used clinically usually includes all structures and targets of interest in the radiotherapy plan, each line plotted a different color, representing a different structure. The vertical axis is almost always plotted as percent volume (rather than absolute volume), as well.

## **5.2 Gamma passing rate vs DVH consistency**

For current routine of quality assurance for online ART, only gamma passing rate is calculated to assess the consistency between the dose generated by TPS and the dose generated by another MC engine. However, the paper by Heming Zhen et al (2011) [68] pointed out that gamma passing rate has weak correlation to critical patient DVH errors. That is, the dose error within the critical regions may be significant while the gamma passing rate remains high. Therefore it's not secure to only calculate the gamma passing rate for QA process. Besides, the criteria of calculating gamma passing rate is not standardized, so a high gamma passing rate may not necessarily indicate the two sets of doses are very close with each other since the criteria could be loose. Gamma passing rate, is a single number that weights every voxel equally. However, the organs at risk (OAR, e.g. spinal cord) regions and planning target volume (PTV) are clinically more sensitive to overdose than other regions., and worth assigning more weight of importance. Unfortunately, the weighted gamma passing rate hasn't widely studies or applied yet in radiation oncology.

DVH, on the other hand, treats each region separately so it well resolved the problem of importance weight in gamming passing rate system. Physicians usually pay more attention to DVHs as they directly shown the dose coverage in OARs and PTV. Considering those advantages, we propose the consistency of DVHs as additional criteria of the quality assurance in radiation therapy.

## 5.3 Focus on the important region

The DVHs are calculated within a few small regions instead of the whole phantom. We name these regions as “Important Region” (IR) and the rest “Unimportant Region” (UR). As the important regions only take a small proportion of the whole volume, it is naturally to come up a strategy that we perform detailed simulation only in the IR while do rough estimation in the UR, which will save us a large amount of time.

We first need to propose an appropriate algorithm for the rough estimation in unimportant region. Let’s recall the two features of MRIdian platform introduce in section 4.2: (1) The electrons generated from low-energy photon emitted from Co60 source head can only travel one or two voxels before making a complete stop. (2) The electron returning effect (RE) caused by the magnetic field shortens the distance electrons can travel. In other words, electrons become very localized, and distribution of scattered photons dominates the final dose distribution in those unimportant regions. Therefore, we can simply quit simulating the electron and deposit the whole energy into current voxel as a way of rough estimation.

However, there is one problem for this strategy. The electrons in these voxels belonging to unimportant region but adjacent to the important region will have chance to enter important region. Simply quitting these electrons will jeopardize the dose accuracy in important region. To count the electron energy in IR correctly, we will label each voxel with a variable  $S(ix, iy, iz)$  representing the “effective” nearest possible distance to the important regions. Then our simulation algorithm works in the following way. Whenever an electron is generated in voxel  $(ix, iy, iz)$  with energy  $E$ , we calculate the CSDA range  $R(E)$ . If  $R(E) < S(ix, iy, iz)$ , deposit all the energy  $E$  in voxel  $(ix, iy, iz)$  and return. Otherwise we perform detailed CSDA simulation, and deposit energy gradually along the path.

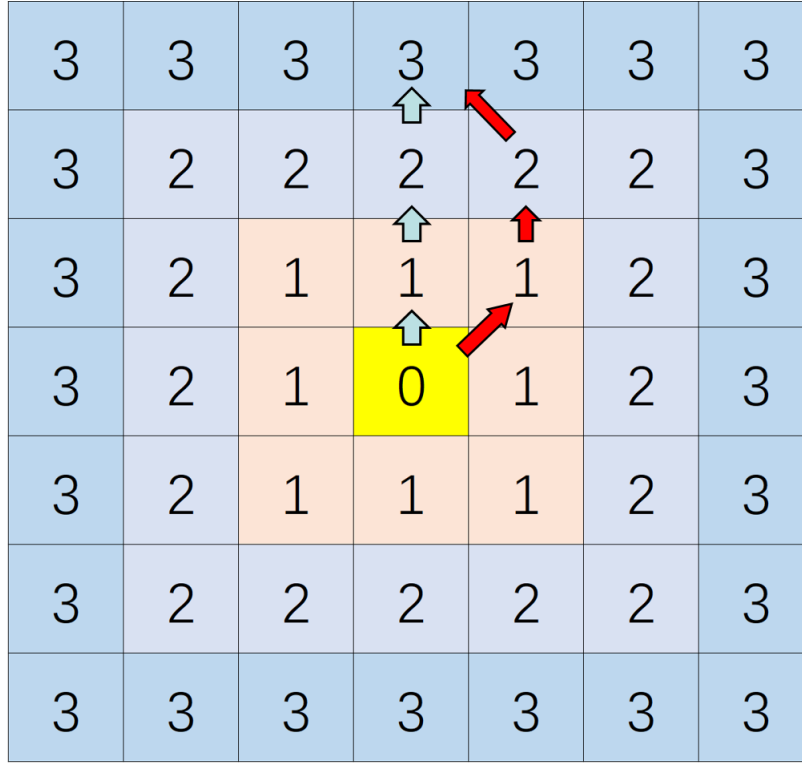


Figure 5.2 2D illustration of the reverse free walk. The yellow pixel belongs to the important region while the rest belong to the unimportant region.

Now the remaining question is how should the variable  $S(ix, iy, iz)$  for each voxel be calculated. Instead of testing if each voxel can enter the important regions, we can perform reverse free walk from the important regions, and mark the shortest accumulated “effective” path, which is weighted by the density of each voxel. The algorithm is illustrated in 2D diagram as **Figure 5.2**. The yellow pixel belongs to the important region and the rest belong to the unimportant region. Starting from the yellow pixel, we can visit 8 adjacent pink pixels. Again from each of these pink pixels, we can visit less than 8 outer adjacent grey pixels. That is, each step must go further away from the yellow center. Note there are different paths that reach the same voxel from the IR. We need to record the shortest during the free walk and we can stop it

once the accumulated “effective” walk length is greater than 2 voxels since no electron can travel over two voxels (with water density) in MRIdian platform. Finally, the labeling algorithm can be implemented via recursive depth-first search as following pseudo code:

```

1 Initialize IR voxels with mark = -1 and UR voxels with mark = 10e9;
2 Call recursiveWalk(ix, iy, iz, 0, 0) for each IR voxel
3
4 void recursiveWalk(int ix, int iy, int iz, int layer, float walked){
5     mark(ix, iy, iz) = min(mark(ix, iy, iz), walked);
6     if (walked > 2) return;
7     iterate adjacent 26 voxels (index ax, ay, az) {
8         If (voxel(ax,ay,az) within phantom, and at layer+1, and not an IR voxel) {
9             recursiveWalk(ax, ay, az, layer + 1, walked + density(ax, ay, az))
10        }
11    }
12 }

```

The recursive algorithm could be time-consuming and memory-exhausting when the surroundings of the important regions are of low density. Fortunately, the labeling matrix can be reused once it was done. The new GPU code, called gDVH, is easily built based on gDPMvr with a few modifications.

## 5.4 Accuracy and performance

We choose a patient plan with pelvis tumor as the input, and calculate the dose distribution by gDPM and gDVH with only OARs and PTV as important region. **Figure 5.3** shows the transaxial iso-dose lines of the two distributions. We can observe that the two dose distributions are very similar to each other except the voxels around the PTV boundary. This phenomenon is caused by the change of electron simulation strategy around the PTV adjacent voxels. The line dose profile comparison across the PTV shown in Figure 5.4 (left) further confirmed the conclusion.

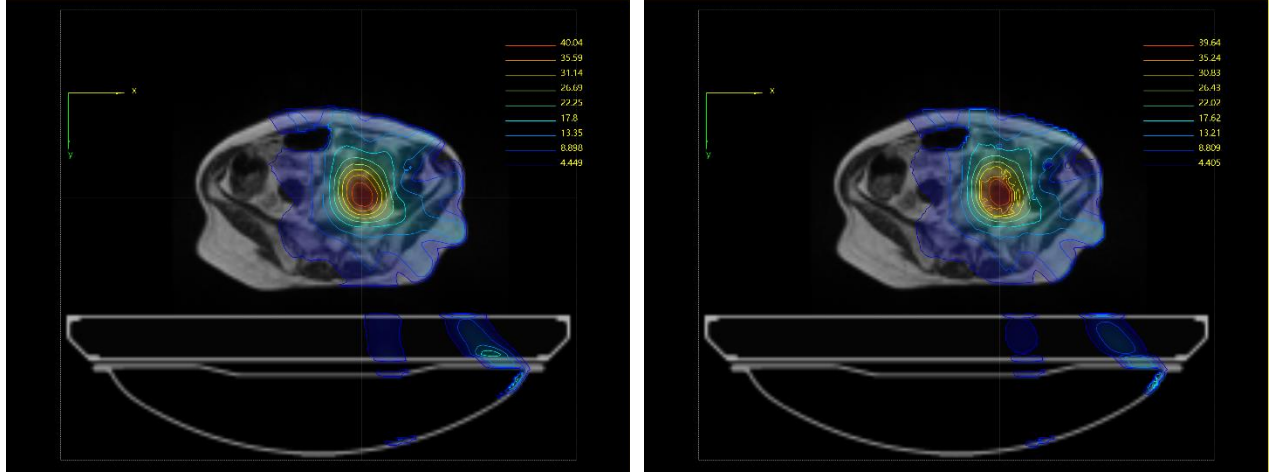


Figure 5.3 (left) dose distribution calculated by gDPMvr. (right) dose distribution calculated by gDVH with PTV as the important regions.

Figure 5.4 (right) shows the comparison of DVHs generated by gDPMvr and gDVH respectively. They are consistent with each other within 1% statistical error (each simulation yields 1% uncertainty so the difference could be 2%). Therefore we can assert that gDVH generates identical DVHs to gDPMvr with significant performance improvement that will be shown later.

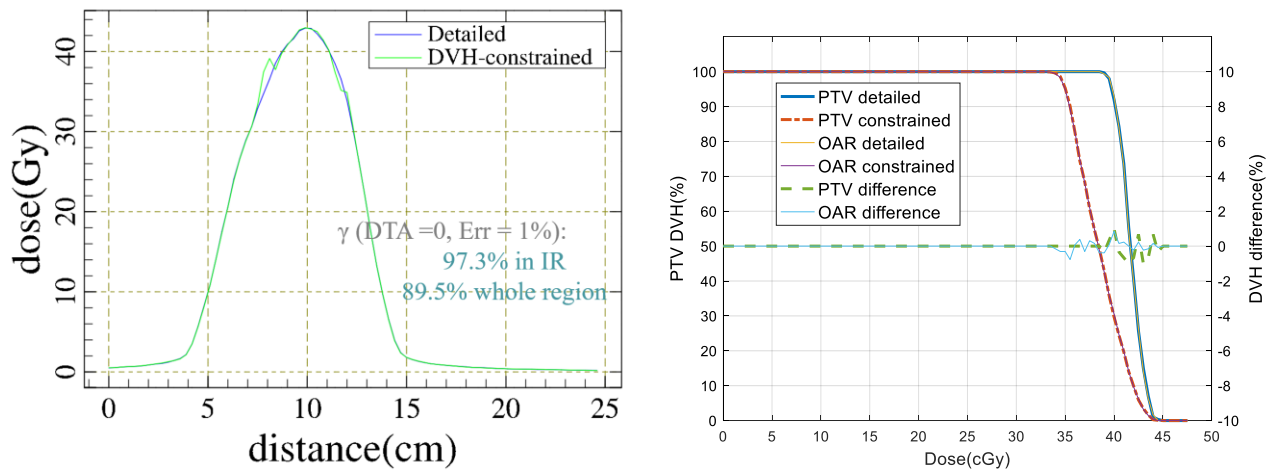
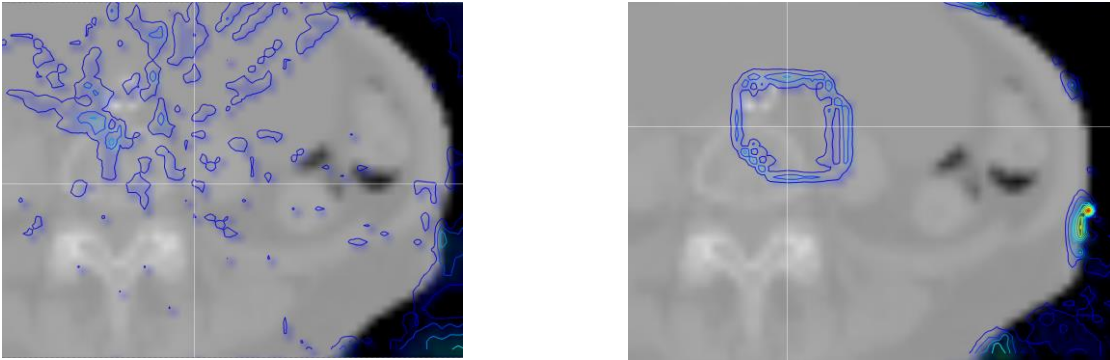


Figure 5.4 (left) comparison of line dose profiles. (right) comparison of DVHs (PTV).

Figure 5.5 demonstrated the change of gamma index distribution when we use gDVH instead of gDPMvr to QA the treatment plans. Note that the colors only mean relative values so it doesn't represent the absolute quantities. We can see the large gamma values (meaning larger error) spread randomly in the case of gDPMvr vs KMC, while distribution locally around the boundary of PTV in the case of gDVH vs KMC. This coincides with our previous observations. Now let us check the change of gamma passing rates for above two scenarios. Since it is comparison between two MC simulations, we should set  $DTA = 0$  as no extra placement error is introduced in both MC systems. If we set  $\Delta D = 2\%D_{max}$ , the gamma passing rate will drop from 98.9% to 93.4% when applying "DVH constraint". However, if we only take the PTV volumes into consideration, the gamma passing rate has merely changed.



**Figure 5.5** (left) gamma distribution of gDPMvr vs KMC. (right) gamma distribution of gDVH vs KMC.

The performance benchmarks were run on the same server equipped with a Tesla K80 GPU card from NVIDIA and a Xeon E5 2630 v3 CPU from Intel. The average run time for this pelvis patient is listed in Table 5.1. As the Co60 radiation head is provide by vendor in C++ module running on CPU, our simulation efficiency is restricted by the CPU's capability. For both simulations, the CPU will cost 42 seconds to generate incident particles. The overall acceleration ratio is **1.7**. If not considering the CPU part, the acceleration ratio becomes as high as **2.7**. Now

the fastest gDVH can reach 0.5% overall uncertainty in **1.2 minutes** (0.5 billion effective histories).

**Table 5.1** Performance comparison between gDPMvr and gDVH

Type	GPU time (s)	CPU time (s)	Overall time (s)
gDPMvr	71	42	113
gDVH	26	42	68

## 5.5 Discussion and conclusion

In this chapter, we introduced the concept of “DVH constraint”, and applied it to accelerated our previous gDPMvr code by around two times. Accordingly, we name the new code “gDVH”. The idea is to do detailed simulation in the regions of our interest but only perform rough estimation in the rest regions. This strategy successfully reduced the simulation time of a pelvis treatment plan from 1.9 minutes to 1.1 minutes while maintaining a 0.5% overall uncertainty. The simulation efficiency can be further improved if the Co60 head model is reimplemented in GPU code as well. In fact, the code design of Co60 head model is quite complex including several features that are unfriendly to GPU programming (e.g. virtual functions). So it requests a lot of efforts to rewrite the infrastructure before it becomes readily implementable in CUDA language.

## Chapter 6: Geometry system

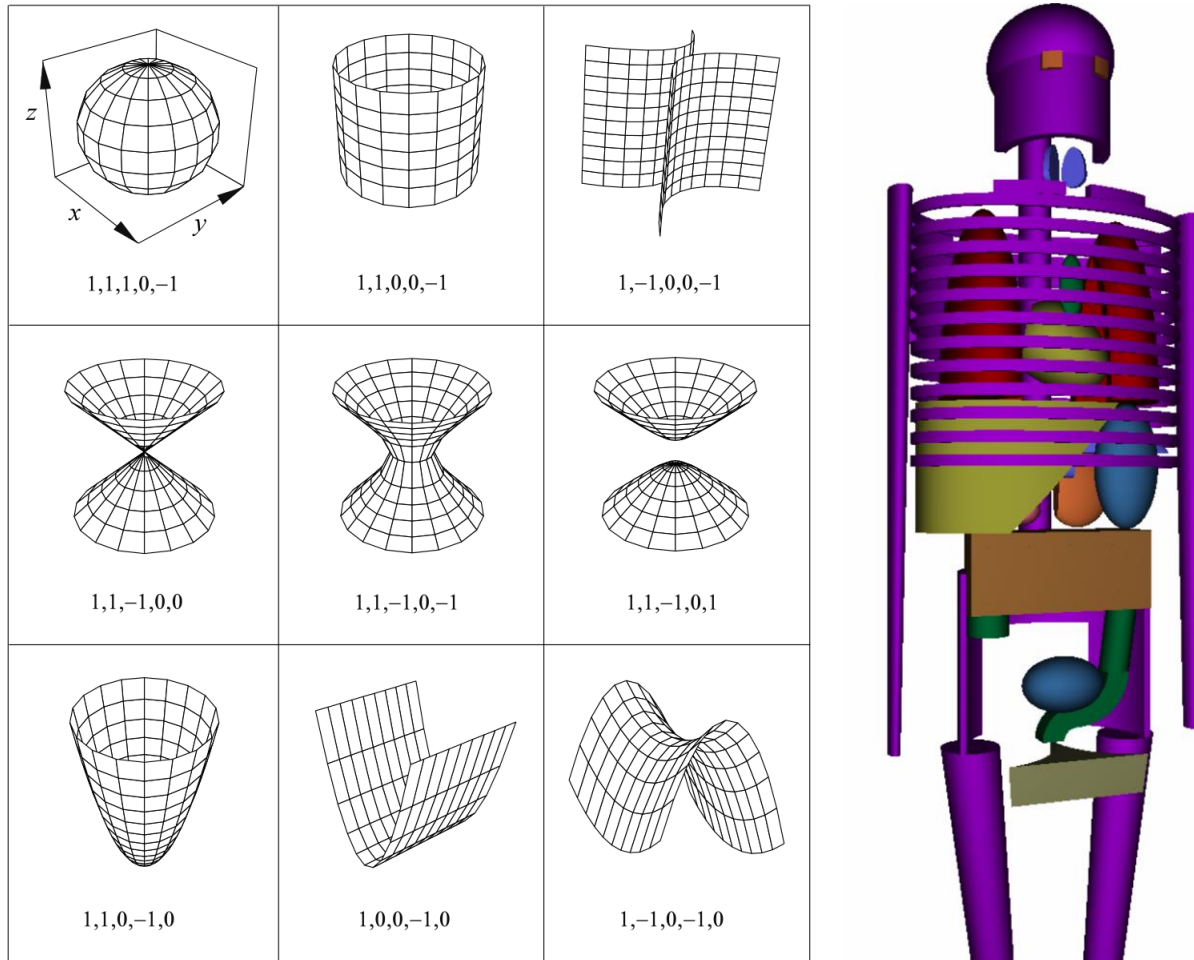
In previous chapters, we introduced the principles of Monte Carlo simulation and four methods to accelerate the simulation. Accordingly, we built three MC packages (gPENELOPE, gDPMvr and gDVH) that are suitable for different applications. However, these packages are all targeted at the in-patient dose calculation, i.e. simulating the particle transport through uniform 3D grids. Other scenarios, such as modeling ion-chamber, film response, etc., will require different geometry supports. It should be either very accurate or very convenient for deformable structures. Moreover, it should be easy to use. That is, the geometry system is able to import and utilize 3D models that are constructed or easily converted by modern CAD tools.

The most frequently used MC package in this dissertation, PENELOPE, includes a quadric geometry system shown in Figure 6.1. It applies quadric surfaces (Figure 6.1 left) and planar surfaces to describe the geometry configuration, and can build complex models shown in Figure 6.1 (right). This geometry system has very high transport efficiency because all surfaces are determined by the quadratic function and the intersection point with a ray can be easily calculated by solving a quadratic equation. However, the disadvantage is also obvious. The surfaces can only be described by combination of quadric and planar surfaces that are unable to precisely model many objects. The skeleton shown in Figure 6.1 (right) demonstrates how poor the quadric model is. An alternative geometry module that describe the geometry precisely is strongly here.

The job of the geometry routines is to steer the simulation of particle histories in the actual material system. They must determine the active medium, change it when the particle crosses an interface (i.e. a surface that separates two different media) and, for certain simulation algorithms,



they must also keep control of the proximity of interfaces. In this chapter, we will build three geometry modules that are suitable for regularly shaped model, arbitrary triangle-mesh model and arbitrary tetrahedron-mesh model respectively.

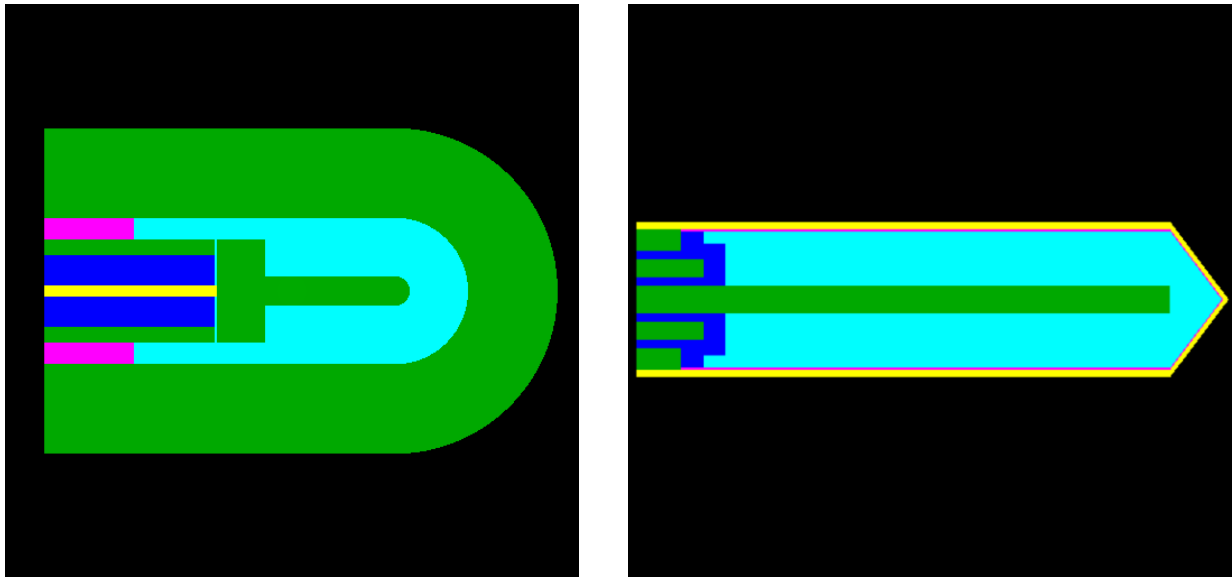


**Figure 6.1** (left) Non-planar reduced quadric surfaces and their indices in PENELOPE. (right) A skeleton constructed by PENELOPE's quadratic surfaces.

## 6.1 Regularly shaped model

Many measurement devices such as icon-chamber, EBT film, Arch-Check, etc. are made of regularly shaped primitives. For the response simulation research of these devices, it is natural to model them by combination of many primitives. Though PENELOPE's geometry module can

construct some primitives, its complex configuration and limited surface types prevent itself being applied in those scenarios. We need another geometry module that offers many types of primitives and convenient ways to glue these primitives into the target objects. The motto of “Don’t reinvent wheels” drive us to firstly look for ready- to-use libraries instead of building something from scratches. We found that another popular MC package, EGSnrc [13], is shipped with a powerful geometry module named “egs++”. It provides many primitives such as boxes, spheres, cylinders, etc., and various boolean operations (unions, logical, or, etc.) to put together more complicated objects. We can extract the source code, and merge it to our PENELOPE-based systems. Figure 6.2 shows two ion-chamber models constructed by egs++. They are used to simulate the dose response via PENELOPE with the presentence of strong magnetic fields.



**Figure 6.2** (left) Model of A18 ion-chamber. (right) Model of Farmer ion-chamber.

The egs++ geometry package considers geometrical structures at the highest possible level of abstraction: any object that is able to provide a certain set of geometry related methods is considered to be a "geometry". No distinction is made between surfaces or solids, or between

simple geometrical structures and highly complex ones. An object is considered to be a geometry if it can provide answers to the following questions:

- (1) Given a region index  $i$ , a position  $\vec{x}$ , a direction  $\vec{u}$  and an intended transport distance  $t$ , will the particle trajectory intersect a boundary? If yes, what is the new region index and what is the distance to the boundary? The method providing the answer to this questions will be referred to as the ***howfar()*** method of a geometry and is specified by the *howfar()* pure virtual function of the EGS\_BaseGeometry class.
- (2) Given a region index  $i$  and a position  $\vec{x}$ , what is the nearest distance to a boundary in any direction? The method providing the answer to this questions will be referred to as the ***hownear()*** method of a geometry and is specified by the *hownear()* pure virtual function of the EGS\_BaseGeometry class.
- (3) Is position  $\vec{x}$  inside or outside the geometry? The method providing the answer to this questions will be referred to as the ***isInside()*** method of a geometry and is specified by the *isInside()* pure virtual function of the EGS\_BaseGeometry class.
- (4) In addition to the above, what is the region index corresponding to  $\vec{x}$  if it is inside? The method providing the answer to this questions will be referred to as the ***isWhere()*** method of a geometry and is specified by the *isWhere()* pure virtual function of the EGS\_BaseGeometry class.
- (5) What is the medium in region  $i$ ? The method providing the answer to this questions will be referred to as the ***medium()*** method of a geometry specified by the *medium()* virtual function of the EGS\_BaseGeometry class.

(6) How many regions are there in this geometry? The method providing the answer to this questions will be referred to as the ***regions()*** method of a geometry specified by the *regions()* virtual function of the EGS\_BaseGeometry class.

As a convention, all geometries numerate their regions between 0 and the number of regions minus one whereas a negative region index is considered to be outside of the geometry (i.e., if a particle would exit the geometry after crossing a boundary, the new region index returned is -1, or if the region index  $i$  is negative in questions 1 and 2, the geometry object can assume that it is known that the position  $\vec{x}$  is outside of the geometry). Questions 1 and 2 are specified by the EGSnrc geometry interface specification except that now geometry objects must be able to determine the answer to these questions also for the situation of the position being outside (i.e. region  $i$  is negative). This extension, together with 3, 4 and 6 is necessary so that one can construct more complicated geometries from simpler geometries. Question 5 is necessary to completely decouple the geometry information from simulation kernels.

To describe the various geometry objects provided by the egs++ library, we will group them in two classes: (1) Elementary or primitive geometries. These geometries are called elementary not because it is easy to implement the required methods but because these methods are implemented directly, without the use of geometry methods of other objects. (2) Composite geometries. The geometry methods of such geometries are implemented using the geometry methods of the objects from which such geometries are built using a certain type of logic to obtain ***howfar()***, ***hownear()***, etc., from the corresponding methods of the constituents. Composite geometries can be constructed from elementary geometries and/or other composite geometries.

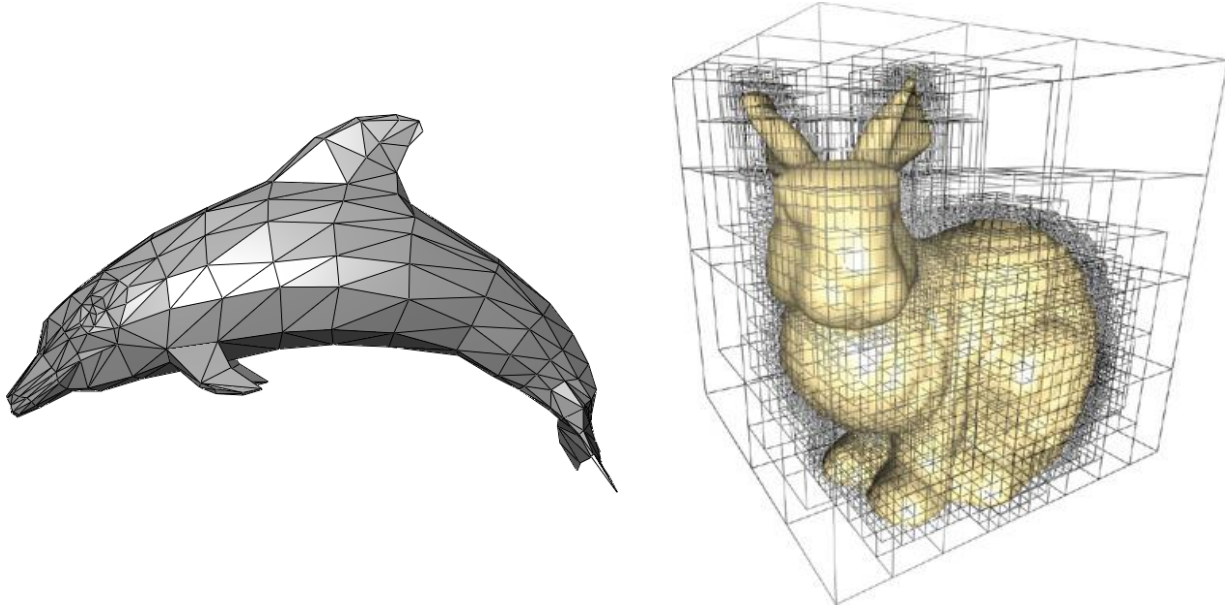
Given the above discussion, all geometry objects in the `egs++` package are derived from the `EGS_BaseGeometry` class, which is part of the main `egs++` library. Concrete geometry classes are compiled into separate shared libraries (a.k.a. dynamic shared objects, DSO, or dynamically linkable library, DLL) that can be loaded dynamically at run time as needed. Each of these geometry libraries provides a `EGS_BaseGeometry *createGeometry(EGS_Input *inp)` C-style function, the address of which is resolved when a geometry library is loaded and is used to create a geometry object from the input information stored in an `EGS_Input` object and pointed to by `inp`. The information stored in the input object is typically extracted from an input file that specifies the various aspects of a particle simulation. It is of course possible to create an `EGS_Input` object specifying one or more geometries by other means (e.g. within a GUI) and then use the geometry creation functions `EGS_BaseGeometry::createGeometry()` or `EGS_BaseGeometry::createSingleGeometry()` to obtain a pointer to the geometry object.

The motivation behind this design is twofold: (1) Most of the time simulations are performed within a geometry that only requires a single class or a limited set of classes to be modeled. It would therefore be wasteful to link against a library containing all geometry classes available in `egs++`. (2) Extendibility: it is easy to create a new geometry class by deriving from `EGS_BaseGeometry`, implementing the necessary methods and the `createGeometry` function and compiling the class into a shared library that can immediately be used with the rest of the system.

## 6.2 Arbitrary triangle-mesh model

In section 6.1, we borrowed the `egs++` module from `EGSnrc` to conveniently model objects with regular shapes. In more general scenarios, however, the objects to be modeled usually own arbitrary/irregular shapes, which is beyond `egs++`'s coverage. An easy solution is to borrow the

idea of mature 3D game character models, i.e. to approximate those complex surfaces via closed triangular meshes (e.g Figure 6.3 left).

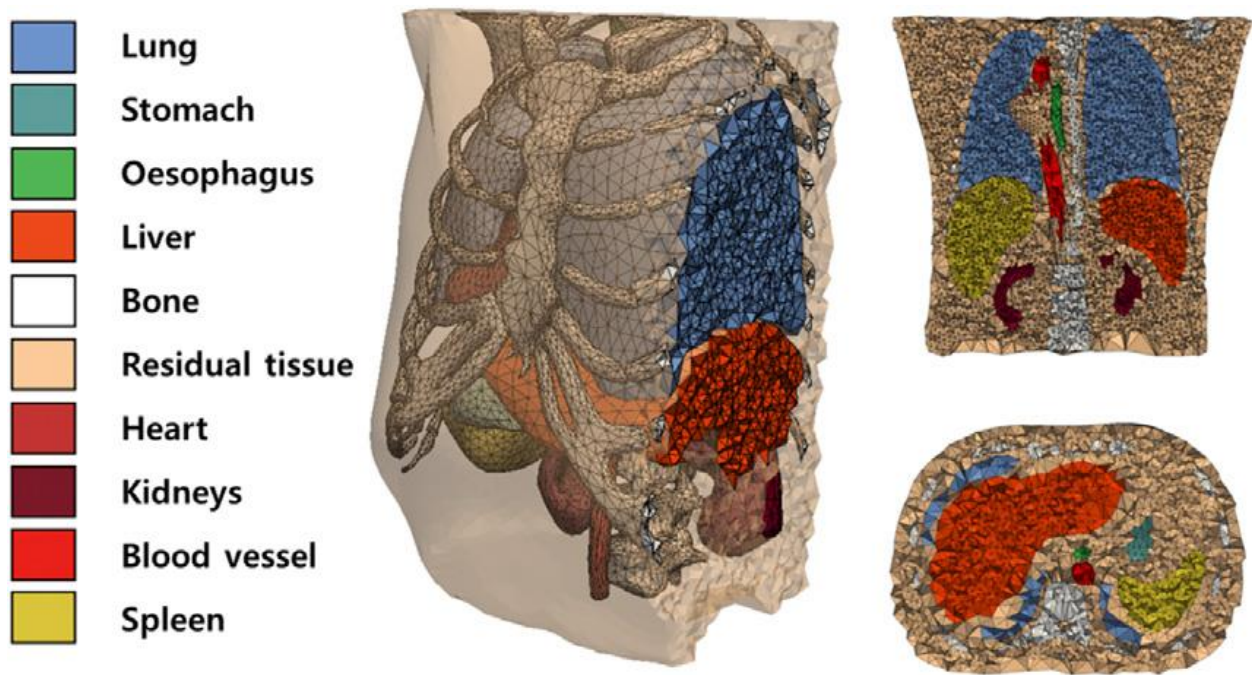


**Figure 6.3** (left) Dolphin modeled by triangular mesh. (right) demonstration of spatial octree division of triangles.

However, each object may require many (hundreds to thousands) small triangles to reach a modest description. The intersection tests with a large number of triangles become the biggest obstacle to high simulation efficiency. Luckily, there is a mature and effective ray-tracing solution existed in gaming industry. That is to recursively divide the space into an octree so that each leaf has none or only a few triangles to compare (see Figure 6.3 right). We also need to record the pointers of neighbor nodes for fast node switch. The C++ triangle-mesh module for PENELOPE has been reported by Badal *et al* [69]. Based on their work, we made some modifications and implemented the first GPU based triangle-mesh geometry library. This library can be applied with gPENELOPE to improve the accuracy of LINAC output simulations.

## 6.3 Arbitrary tetrahedron-mesh model

Though the geometry modules introduced in section 6.1 and 6.2 can conveniently model many objects, they are not able to provide the radiation dose distribution as we do for in-patient voxel grid. For objects of arbitrary shape, it is also difficult to bridge voxel grids inside. One geometry model formed by tetrahedron mesh can provides natural containers for spatial dose counting, and be can deformed corresponding to the motion of patients' organs (see Figure 6.4).



**Figure 6.4** Human body modeled by tetrahedron mesh. It provides natural containers for spatial dose counting, and can be easily deformed to simulate the motion of patients' organs.

As the tetrahedrons are naturally adjacent, it is easier to perform ray-tracing as the node switch can be very fast. The transport algorithm has been reported by Qianqian Fang (2010) [70]. Though his work only simulated photons, it's easy to add electron transport extensions. One of his optimizations is to do Ray-polygon intersection test using Plücker coordinates instead of the

regular coordinates. This algorithm basically trades the computation cost with the memory consumption.

Those tetrahedron meshes for simulation can be constructed from CAD models or from a set of CT images. The famous physics simulation software COMSOL provides a complete toolkit to build tetrahedron-mesh models. The approach of generating tetrahedron-mesh models from CT scans has been reported by Qianqian Fang (2009) [71], who also released a set of easy-to-use MATLAB scripts called “iso2mesh” on GitHub.



# Chapter 7: Graphic user interface

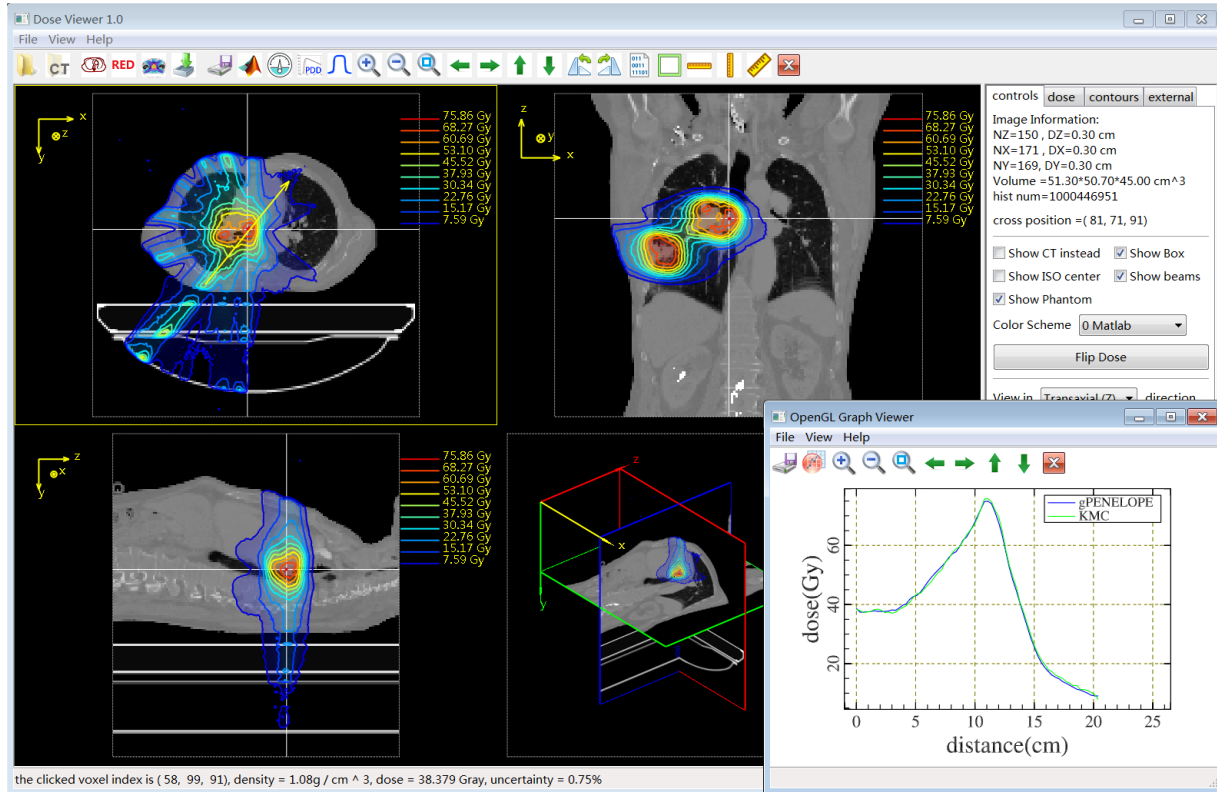
A good designed toolkit should be friendly to all users. We integrate all functionalities for inpatient dose QA into a graphic user interface (GUI) called DoseViewer to deliver best user experience. Similarly, we integrate all modules for tetrahedron-mesh based simulations into a GUI called TetViewer. Users can performance complicated simulations and view the results at the same time without having to edit tedious configurations or write any code.

## 7.1 DoseViewer

As the name indicates, DoseViewer should be able to display the dose distribution comprehensively to users. Although several software packages exist for visualizing 3D dose distributions, such as CERR (MATLAB) [72] and 3DSlicer (C++) [73] among others, they were either developed in script language which has relatively low executing efficiency, or rely on cumbersome runtime libraries which complicate software distribution. More importantly, it is difficult to integrate customized functionality into existing packages like those mentioned above. We therefore decided to develop our own lightweight C++ program – called DoseViewer – which enables DICOM phantom creation/conversion, dose calculation, multiple views, 3D observation, dose profiling, gamma analysis (GPU-accelerated), etc. in a compact size. The highly responsive and user-friendly GUI for DoseViewer is shown in **Figure 7.1**.

Since all was designed as a cross-platform package, we decided to make DoseViewer cross-platform as well using the library wxWidgets to display the GUI framework and OpenGL to render the phantom and dose in 3D space. We utilized the DCMTK library developed by OFFIS to load DICOM data (including CT images, dose, and contours). We also used OpenCV to simplify image manipulation. Besides standard DICOM files, DoseViewer can also load dose and phantom data in MRIdian formats for dose comparison and gamma analysis. Moreover, we

applied a lightweight MathGL library to enable quick visualization of dose profiles. To accelerate 3D gamma analysis for large dose matrices, we developed a DLL module using GPU computation which is at least four times faster than CPU code (NVidia GT650M GPU vs Intel i7 3630QM CPU). To further save time, we also ported many MATLAB functions (e.g. 3D interpolation) to C++ code through MATLAB coder.

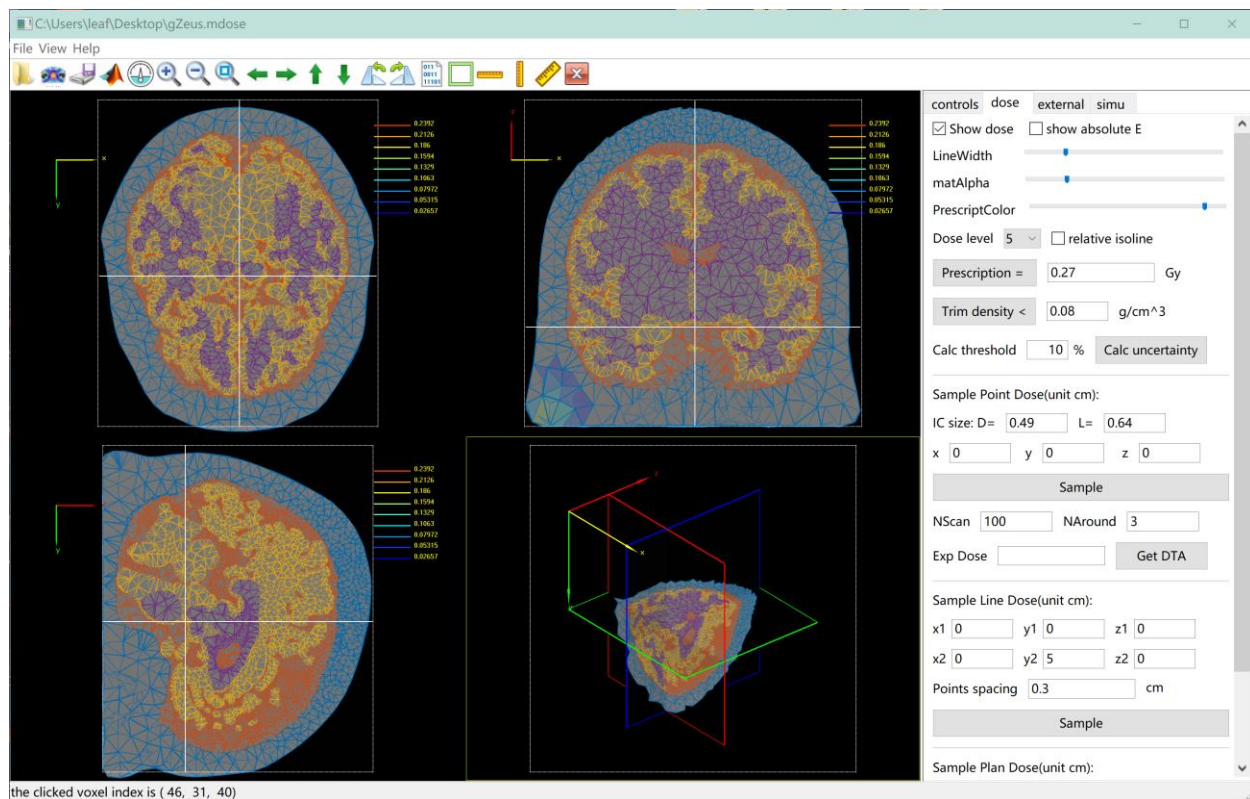


**Figure 7.1** GUI of DoseViewer that enables DICOM phantom creation/conversion, dose calculation, multiple views, 3D observation, dose profiling and gamma analysis (GPU accelerated), etc. in a compact size. The smaller window, as an example, shows an arbitrary dose profile comparison between gPENELOPE and KMC (engine of MRIdian’s treatment planning system) in a lung case.

## 7.2 TetViewer

The implantation of TetView (see **Figure 7.2**) is very similar to DoseViewer except its main window is to display tetrahedron meshes and the dose recorded in those tetrahedrons. Since the

boundary of the tetrahedron mesh is clear, we can improve the visualization by assigning parent weight outside the boundary in each slice, and then we can view elements behind the slides.



**Figure 7.2** GUI of TetViewer that provides 3D display of tetrahedron meshes and radiation dose.

# Chapter 8: Conclusion and outlook

Through the efforts introduced in previous chapters, we have successfully built a complete toolkit for fast Monte Carlo dose calculation in radiation therapy. Here, we will briefly recall the methodology and produced results, and have an outlook of the future of fast Monte Carlo simulation and its applications in radiation therapy.

## 8.1 Summary of the results

The key results are three fast Monte Carlo dose calculation packages suitable for different application scenarios. First, we applied GPU parallelization on the famous Monte Carlo code PENELOPE and build a new code system gPENELOPE that runs 5 times faster and remains the same accuracy. It works well for applications requiring high accuracy and fast speed. Second, we further took two steps, i.e. transport simplification and variance reduction to build an even faster code system named gDPMvr. Benchmarks show that the transport simplification and variance reduction can accelerate the simulation by about 7 and 6 times respectively. gDPMvr runs 43 times faster than gPENELOPE but it only compromises 1% accuracy, so it's well suited for in-patient dose calculation purpose. Third, we applied "DVH constraint" on gDPMvr and build another code system named gDVH, which finally double the simulation efficiency. It is spherically designed for DVH oriented QA. With all these efforts, the simulation time can be significantly reduced from 5 hours to 1.2 minutes for a typical treatment plan.

This dissertation also covers 3 applications of this toolkit other than the primary QA purpose. These are: (1) Validate the vender provided Co60 radiation head model by comparing the dose calculated by gPENELOPE to experiment data; (2) Quantitatively study the effect of magnetic field to dose distribution and proposed a strategy to improve treatment planning efficiency; (3) Evaluate the accuracy of the build-in MC algorithm of MRIdian's treatment

planning system. Many other time-sensitive applications (e.g. motional dose accumulation) will also benefit a lot from our fast MC infrastructure.

To extend our code systems to more general scenarios, we also integrated a geometry module that can easily handle regularly shaped model, arbitrary triangle-mesh model and arbitrary tetrahedron-mesh model. Moreover, we also developed two graphic user interfaces (GUIs) called “DoseViewer” and “TetViewer” to lower the level of difficulties to average users.

In summary, this dissertation presents a complete toolkit for fast Monte Carlo simulations and can be widely applied in radiation therapy.

## **8.2 Outlook**

The Monte Carlo dose calculation method has been used in radiation therapy for over half a century, and but it is not applied as widely as its competitor -- convolution-superposition (CS) method due to speed issue. The recent rise of MRI guided radiation therapy draws new attention to the relative mature Monte Carlo algorithm because the introduced magnetic field may cause significant error in CS algorithm. Monte Carlo algorithm fundamentally correct the error since it simulates millions of particles in a microscopic sense. The challenge lies at how to improve the simulation efficiency to clinically acceptable level.

As the single core in CPU has almost reached its physical limitation for calculating speed, the multi-core structure becomes more feasible to improve overall performance. This strategy propels the debut of general-purpose GPU programming language, and then we enter the new era of parallel computation on GPU. The conventional Monte Carlo packages, however, are not readily applicable to GPU architecture. In the future, those Monte Carlo developers may spend efforts to adapt their codes to GPU for better performance, and they have to overcome the two major GPU programming difficulties mentioned in section 3.2.4. However, the GPU hardware

may be redesigned to address the two issues instead of making painful effort in programming level. Maybe programming on next-generation GPU platform would be as easily as we do now on CPU platform.

One drawback of current Monte Carlo algorithm is the multiple scattering theory of a large electron step in magnetic field hasn't been established. The introduced magnetic field makes the original theory unfixable. The algorithm we use today is based on a simple unjustified fix. It will cause big error if the advancing step becomes large. Usually the efficiency is much restricted. Finding a close and efficiency approximation remains a big challenge. We expect a satisfactory solution in the near future.

One competitor of the Monte Carlo algorithm is so-called “deterministic linear Boltzmann transport equation (D-LBTE) solver” [74], which establishes a complicated Boltzmann transport equation to describe the statistical status of the particles and then obtains the dose distribution by solving the D-LBTE in a deterministic way (by iterations for example). The algorithm can produce results much faster than conventional Monte Carlo packages and yield similar level of accuracy. However, the solver may not give a converged solution after many iterations. Its speed cannot beat the recent fast Monte Carlo packages. The “D-LBTE solver” algorithm is still under active research and development.

# Bibliography

- [1] R.R. Wilson, The Range and Straggling of High Energy Electrons, *Physical Review*, 84 (1951) 100-103.
- [2] R.R. Wilson, Monte Carlo Study of Shower Production, *Physical Review*, 86 (1952) 261-269.
- [3] E.C. Halperin, C.A. Perez, L.W. Brady, Perez and Brady's principles and practice of radiation oncology, 5th ed., Wolters Kluwer Health/Lippincott Williams & Wilkins, Philadelphia, 2008.
- [4] I.J. Chetty, B. Curran, J.E. Cygler, J.J. DeMarco, G. Ezzell, B.A. Faddegon, I. Kawrakow, P.J. Keall, H. Liu, C.M. Ma, D.W. Rogers, J. Seuntjens, D. Sheikh-Bagheri, J.V. Siebers, Report of the AAPM Task Group No. 105: Issues associated with clinical implementation of Monte Carlo-based photon and electron external beam treatment planning, *Med Phys*, 34 (2007) 4818-4853.
- [5] D.W. Rogers, Fifty years of Monte Carlo simulations for medical physics, *Phys Med Biol*, 51 (2006) R287-301.
- [6] S. Mutic, J.F. Dempsey, The ViewRay system: magnetic resonance-guided and controlled radiotherapy, *Seminars in radiation oncology*, 24 (2014) 196-199.
- [7] R.L. Harrison, Introduction To Monte Carlo Simulation, AIP conference proceedings, 1204 (2010) 17-21.
- [8] J.C. Butcher, H. Messel, ELECTRON NUMBER DISTRIBUTION IN ELECTRON-PHOTON SHOWERS IN AIR AND ALUMINIUM ABSORBERS, *Nuclear Phys.*, (1960) Medium: X; Size: Pages: 15-128.
- [9] I. Kawrakow, Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version, *Med Phys*, 27 (2000) 485-498.
- [10] J.F. Briesmeister, MCNP-a general Monte Carlo N-particle transport code. Los Alamos National Laboratory Report LA-12625-M 1993.
- [11] 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, H. Burkhardt, S. Chauvie, J. Chuma, R. Chytrcek, G. Cooperman, G. Cosmo, P. Degtyarenko, A. Dell'Acqua, G. Depaola, D. Dietrich, R. Enami, A. Feliciello, C. Ferguson, H. Fesefeldt, G. Folger, F. Foppiano, A. Forti, S. Garelli, S. Giani, R. Giannitrapani, D. Gibin, J.J.G. Cadenas, I. Gonzalez, G.G. Abril, G. Greeniaus, W. Greiner, V. Grichine, A. Grossheim, S. Guatelli, P. Gumplinger, R. Hamatsu, K. Hashimoto, H. Hasui, A. Heikkinen, A. Howard, V. Ivanchenko, A. Johnson, F.W. Jones, J. Kallenbach, N. Kanaya, M. Kawabata, Y. Kawabata, M. Kawaguti, S. Kelner, P. Kent, A. Kimura, T. Kodama, R. Kokoulin, M. Kossov, H. Kurashige, E. Lamanna, T. Lampen, V. Lara, V. Lefebure, F. Lei, M. Liendl, W. Lockman, F. Longo, S. Magni, M. Maire, E. Medernach, K. Minamimoto, P.M. de Freitas, Y. Morita, K. Murakami, M. Nagamatsu, R. Nartallo, P. Nieminen, T. Nishimura, K. Ohtsubo, M. Okamura, S. O'Neale, Y. Oohata, K. Paech, J. Perl, A. Pfeiffer, M.G. Pia, F. Ranjard, A. Rybin, S. Sadilov, E. Di Salvo, G. Santin, T. Sasaki, N. Savvas, Y. Sawada, S. Scherer, S. Seil, V. Sirotenko, D. Smith, N. Starkov, H. Stoecker, J. Sulkimo, M. Takahata, S. Tanaka, E. Tcherniaev, E.S. Tehrani, M. Tropeano, P. Truscott, H. Uno, L. Urban, P. Urban, M. Verderi, A. Walkden, W. Wander, H. Weber, J.P. Wellisch, T. Wenaus, D.C. Williams, D. Wright, T. Yamada, H. Yoshida, D. Zschiesche, GEANT4-a simulation toolkit, *Nuclear Instruments & Methods in Physics Research Section a-Accelerators Spectrometers Detectors and Associated Equipment*, 506 (2003) 250-303.
- [12] J. Baro, J. Sempau, J.M. Fernandezvarea, F. Salvat, Penelope - an Algorithm for Monte-Carlo Simulation of the Penetration and Energy-Loss of Electrons and Positrons in Matter, *Nucl Instrum Meth B*, 100 (1995) 31-46.

- [13] I. Kawrakow, D.W.O. Rogers, The EGSnrc code system: Monte Carlo simulation of electron and photon transport. Ionizing radiation standards,” Report No. PIRS-701 (NRC, Ottawa, Ontario, 2003).
- [14] F. Salvat, The PENELOPE code system. Specific features and recent improvements, *Ann Nucl Energy*, 82 (2015) 98-109.
- [15] G. Pratx, L. Xing, GPU computing in medical physics: a review, *Med Phys*, 38 (2011) 2685-2697.
- [16] I. Kawrakow, M. Fippel, K. Friedrich, 3D electron dose calculation using a Voxel based Monte Carlo algorithm (VMC), *Med Phys*, 23 (1996) 445-457.
- [17] M. Fippel, Fast Monte Carlo dose calculation for photon beams based on the VMC electron algorithm, *Med Phys*, 26 (1999) 1466-1475.
- [18] J. Gardner, J. Siebers, I. Kawrakow, Dose calculation validation of Vmc++ for photon beams, *Med Phys*, 34 (2007) 1809-1818.
- [19] J. Sempau, S.J. Wilderman, A.F. Bielajew, DPM, a fast, accurate Monte Carlo code optimized for photon and electron radiotherapy treatment planning dose calculations, *Physics in medicine and biology*, 45 (2000) 2263-2291.
- [20] X. Jia, X. Gu, J. Sempau, D. Choi, A. Majumdar, S.B. Jiang, Development of a GPU-based Monte Carlo dose calculation code for coupled electron-photon transport, *Physics in medicine and biology*, 55 (2010) 3077-3086.
- [21] X. Jia, X. Gu, Y.J. Graves, M. Folkerts, S.B. Jiang, GPU-based fast Monte Carlo simulation for radiotherapy dose calculation, *Physics in medicine and biology*, 56 (2011) 7017-7031.
- [22] S. Hissoiny, B. Ozell, H. Bouchard, P. Despres, GPUMCD: A new GPU-oriented Monte Carlo dose calculation platform, *Medical physics*, 38 (2011) 754-764.
- [23] L. Jahnke, J. Fleckenstein, F. Wenz, J. Hesser, GMC: a GPU implementation of a Monte Carlo dose calculation based on Geant4, *Physics in medicine and biology*, 57 (2012) 1217-1229.
- [24] W.H. Payne, J.R. Rabung, T.P. Bogyo, Coding the Lehmer pseudo-random number generator, *Commun. ACM*, 12 (1969) 85-86.
- [25] P. L'Ecuyer, Efficient and portable combined random number generators, *Commun. ACM*, 31 (1988) 742-751.
- [26] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Trans. Model. Comput. Simul.*, 8 (1998) 3-30.
- [27] M. Manssen, M. Weigel, A. K. Hartmann, Random number generators for massively parallel simulations on GPU, 2012.
- [28] A.J. Walker, An Efficient Method for Generating Discrete Random Variables with General Distributions, *ACM Trans. Math. Softw.*, 3 (1977) 253-256.
- [29] F. Salvat, Algorithms for random sampling from single-variate distributions, *Computer Physics Communications*, 46 (1987) 427-436.
- [30] J.H. Hubbell, H.A. Gimm, I. O'verbo/, Pair, Triplet, and Total Atomic Cross Sections (and Mass Attenuation Coefficients) for 1 MeV-100 GeV Photons in Elements Z=1 to 100, *Journal of Physical and Chemical Reference Data*, 9 (1980) 1023-1148.
- [31] M. Born, R.J. Blin-Stoyle, J.M. Radcliffe, *Atomic physics*, 8th ed., Blackie, London,, 1969.
- [32] D. Brusa, G. Stutz, J.A. Riveros, J.M. Fernández-Varea, F. Salvat, Fast sampling algorithm for the simulation of photon Compton scattering, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 379 (1996) 167-175.
- [33] Y.-S. Tsai, Pair production and bremsstrahlung of charged leptons, *Reviews of Modern*



Physics, 46 (1974) 815-851.

[34] W. Heitler, The quantum theory of radiation, 3d ed., Clarendon Press, Oxford,, 1954.

[35] J.W. Motz, H.A. Olsen, H.W. Koch, Pair Production by Photons, Reviews of Modern Physics, 41 (1969) 581-639.

[36] P. Andreo, A. Brahme, Restricted Energy-Loss Straggling and Multiple Scattering of Electrons in Mixed Monte Carlo Procedures, Radiation research, 100 (1984) 16-29.

[37] L. Eyges, Multiple Scattering with Energy Loss, Physical Review, 74 (1948) 1534-1535.

[38] S. Goudsmit, J.L. Saunderson, Multiple Scattering of Electrons, Physical Review, 57 (1940) 24-29.

[39] I. Kawrakow, M. Fippel, Investigation of variance reduction techniques for Monte Carlo photon dose calculation using XVMC, Physics in medicine and biology, 45 (2000) 2163-2183.

[40] E. Woodcock, T. Murphy, P. Hemmings, S. Longworth, Techniques used in the GEM code for Monte Carlo neutronics calculation. Proc. Conf. Applications of Computing Methods to Reactors ANL-7050, (1965).

[41] K. Kowari, Validity of the continuous-slowing-down approximation in electron degradation, with numerical results for argon, Physical review. A, Atomic, molecular, and optical physics, 41 (1990) 2500-2505.

[42] D.A. Low, W.B. Harms, S. Mutic, J.A. Purdy, A technique for the quantitative evaluation of dose distributions, Med Phys, 25 (1998) 656-661.

[43] A.J. Raaijmakers, B.W. Raaymakers, J.J. Lagendijk, Magnetic-field-induced dose effects in MR-guided radiotherapy systems: dependence on the magnetic field strength, Phys Med Biol, 53 (2008) 909-923.

[44] H.O. Wooten, O. Green, H. Li, V. Rodriguez, S. Mutic, Measurements of the Electron-Return-Effect in a Commercial Magnetic Resonance Image Guided Radiation Therapy System. WE-G-17A-4, AAPM annual meeting, Austin, TX 2014.

[45] A. Molineu, N. Hernandez, T. Nguyen, G. Ibbott, D. Followill, Credentialing results from IMRT irradiations of an anthropomorphic head and neck phantom, Med Phys, 40 (2013) 022101.

[46] G.X. Ding, D.M. Duggan, C.W. Coffey, Commissioning stereotactic radiosurgery beams using both experimental and theoretical methods, Phys Med Biol, 51 (2006) 2549-2566.

[47] J.J. Lagendijk, B.W. Raaymakers, C.A. Van den Berg, M.A. Moerland, M.E. Philippen, M. van Vulpen, MR guidance in radiotherapy, Phys Med Biol, 59 (2014) R349-369.

[48] G.H. Bol, S. Hissoiny, J.J. Lagendijk, B.W. Raaymakers, Fast online Monte Carlo-based IMRT planning for the MRI linear accelerator, Phys Med Biol, 57 (2012) 1375-1385.

[49] E. Gete, C. Duzenli, M.P. Milete, A. Mestrovic, D. Hyde, A.M. Bergman, T. Teke, A Monte Carlo approach to validation of FFF VMAT treatment plans for the TrueBeam linac, Med Phys, 40 (2013) 021707.

[50] N. Dogan, J.V. Siebers, P.J. Keall, F. Lerma, Y. Wu, M. Fatyga, J.F. Williamson, R.K. Schmidt-Ullrich, Improving IMRT dose accuracy via deliverable Monte Carlo optimization for the treatment of head and neck cancer patients, Med Phys, 33 (2006) 4033-4043.

[51] M. Reynolds, B.G. Fallone, S. Rathee, Dose response of selected solid state detectors in applied homogeneous transverse and longitudinal magnetic fields, Med Phys, 41 (2014) 092103.

[52] A. Palm, A.S. Kirov, T. LoSasso, Predicting energy response of radiographic film in a 6 MV x-ray beam using Monte Carlo calculated fluence spectra and absorbed dose, Med Phys, 31 (2004) 3168-3178.

[53] P. Piersimoni, A. Rimoldi, C. Riccardi, M. Pirola, S. Molinelli, M. Ciocca, Optimization of a general-purpose, actively scanned proton beamline for ocular treatments: Geant4 simulations, J

Appl Clin Med Phys, 16 (2015) 5227.

[54] D.A. Jaffray, P.E. Lindsay, K.K. Brock, J.O. Deasy, W.A. Tome, Accurate accumulation of dose for improved understanding of radiation effects in normal tissue, *Int J Radiat Oncol Biol Phys*, 76 (2010) S135-139.

[55] B.E. Nelms, M.F. Chan, G. Jarry, M. Lemire, J. Lowden, C. Hampton, V. Feygelman, Evaluating IMRT and VMAT dose accuracy: practical examples of failure to detect systematic errors when applying a commonly used metric and action levels, *Med Phys*, 40 (2013) 111722.

[56] S. Acharya, B.W. Fischer-Valuck, R. Kashani, P. Parikh, D. Yang, T. Zhao, O. Green, O. Wooten, H. Li, Y. Hu, V. Rodriguez, L. Olsen, C. Robinson, J. Michalski, S. Mutic, J. Olsen, Online Magnetic Resonance Image Guided Adaptive Radiation Therapy: First Clinical Applications, *Int J Radiat Oncol Biol Phys*, 94 (2016) 394-403.

[57] C.E. Noel, L. Santanam, P.J. Parikh, S. Mutic, Process-based quality management for clinical implementation of adaptive radiotherapy, *Med Phys*, 41 (2014) 081717.

[58] S.B. Ahmad, A. Sarfehnia, M.R. Paudel, A. Kim, S. Hissoiny, A. Sahgal, B. Keller, Evaluation of a commercial MRI Linac based Monte Carlo dose calculation algorithm with GEANT4, *Medical physics*, 43 (2016) 894-907.

[59] V.N. Malkov, D.W. Rogers, Charged particle transport in magnetic fields in EGSnrc, *Med Phys*, 43 (2016) 4447-4458.

[60] B.A. Faddegon, I. Kawrakow, Y. Kubyshev, J. Perl, J. Sempau, L. Urban, The accuracy of EGSnrc, Geant4 and PENELOPE Monte Carlo systems for the simulation of electron scatter in external beam radiotherapy, *Phys Med Biol*, 54 (2009) 6151-6163.

[61] Y. Wang, T.R. Mazur, O. Green, Y. Hu, H. Li, V. Rodriguez, H.O. Wooten, D. Yang, T. Zhao, S. Mutic, H. Li, A GPU-accelerated Monte Carlo dose calculation platform and its application toward validating an MRI-guided radiation therapy beam model, *Med Phys*, 43 (2016) 4040-4052.

[62] L.J. Rankine, S. Mein, B. Cai, A. Curcuru, T. Juang, D. Miles, S. Mutic, Y. Wang, M. Oldham, H. Li, Three-dimensional dosimetric validation of a magnetic resonance-guided intensity modulated radiation therapy System *Int J Radiat Oncol Biol Phys*, 97 (2017) 1095-1104.

[63] Y. Yazaki, How the Klein-Nishina formula was derived: Based on the Sangokan Nishina Source Materials, *Proceedings of the Japan Academy. Series B, Physical and biological sciences*, 93 (2017) 399-421.

[64] M. van Vulpen, L. Wang, Within the next five years, adaptive hypofractionation will become the most common form of radiotherapy, *Med Phys*, 43 (2016) 3941.

[65] H. Zhen, B.E. Nelms, W.A. Tome, Moving from gamma passing rates to patient DVH-based QA metrics in pretreatment dose QA, *Med Phys*, 38 (2011) 5477-5489.

[66] M. Goitein, *Radiation oncology : a physicist's-eye view*, Springer, New York, 2008.

[67] W.U. Shipley, J.E. Tepper, G.R. Prout, Jr., L.J. Verhey, O.A. Mendiondo, M. Goitein, A.M. Koehler, H.D. Suit, Proton radiation as boost therapy for localized prostatic carcinoma, *Jama*, 241 (1979) 1912-1915.

[68] H. Zhen, B.E. Nelms, W.A. Tomé, Moving from gamma passing rates to patient DVH-based QA metrics in pretreatment dose QA, *Medical physics*, 38 (2011) 5477-5489.

[69] A. Badal, I. Kyprianou, A. Badano, J. Sempau, Monte Carlo simulation of a realistic anatomical phantom described by triangle meshes: application to prostate brachytherapy imaging, *Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology*, 86 (2008) 99-103.

- [70] Q. Fang, Mesh-based Monte Carlo method using fast ray-tracing in Plucker coordinates, *Biomed Opt Express*, 1 (2010) 165-175.
- [71] F. Qianqian, D.A. Boas, Tetrahedral mesh generation from volumetric binary and grayscale images, in: 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2009, pp. 1142-1145.
- [72] J.O. Deasy, A.I. Blanco, V.H. Clark, CERR: a computational environment for radiotherapy research, *Medical physics*, 30 (2003) 979-985.
- [73] M. Narizzano, G. Arnulfo, S. Ricci, B. Toselli, M. Tisdall, A. Canessa, M.M. Fato, F. Cardinale, SEEG assistant: a 3DSlicer extension to support epilepsy surgery, *BMC bioinformatics*, 18 (2017) 124.
- [74] M.W.K. Kan, P.K.N. Yu, L.H.T. Leung, A Review on the Use of Grid-Based Boltzmann Equation Solvers for Dose Calculation in External Photon Beam Treatment Planning, *BioMed Research International*, 2013 (2013) 10.