AN ABSTRACT OF THE DISSERTATION OF

Alexander J Egan for the degree of Doctor of Philosophy in Medical Physics presented on March 31, 2014.

Development of an accurate Monte Carlo treatment plan calculation framework for the purpose of developing dose calculation error predictors for a widely implemented clinical algorithm.

Abstract approved: ______________________________________

Wolfram U Laub

Purpose Monte Carlo (MC) algorithms are widely accepted as the most accurate method to calculate dose in a patient geometry. For this work the EGSnrc MC code was used as a benchmark for the identification of dose calculation errors produced by the widely implemented analytical anisotropic algorithm (AAA). By correlating the location and magnitude of these errors with the physical conditions under which AAA is known to fail, a set of error prediction methods was developed which can help to identify clinical plans that are at high risk for AAA dose calculation errors. Once these plans are identified, they can be recalculated with a more accurate algorithm.

Methods First, in order to calculate clinical treatment plans with MC, a treatment plan calculation framework (MCTPCF) was developed and validated. The underlying beam model used in the MCTPCF was thoroughly benchmarked against a standard open field data set. Radiochromic film measurements were then used to validate the geometry of the employed MC multileaf collimator (MLC) model. Mechanical functionality of the MCTPCF was verified by calculating several highly modulated clinical treatment plans and comparing them with AAA calculations. Next, three novel error prediction algorithms were developed and validated to a limited extent. The first, designated the field size index (FSI), identifies regions in the treatment plan space where many small fields or blocks overlap, leading to a build-up of beam modeling and volume averaging errors. The second, designated the heterogeneous scatter index (HSI), identifies regions within the electron density distribution where the AAA rectilinear kernel scaling approximation is stressed. The third, designated the low-density index (LDI), identifies regions of very low electron density where AAA is known to overestimate dose.

Results An open field beam model for the 6MV Varian Clinac has been fully parameterized and is able to calculate dose to within 1.3% and 1.0 mm DTA (σ_{mean} = 0.3%). The MCTPCF has been shown to accurately calculate highly modulated, multiple field treatments. FSI calculations show excellent agreement with MC/AAA deviations in highly modulated MLC fields in water, and to a lesser extent in patient geometry RapidArc treatments. The LDI accurately predicts AAA overdosing for simple geometries, however for the lung case investigated other sources of error made identifying any correlation a challenge. The theoretical structure of the HSI has been developed, however its implementation is still underway.

Conclusion An accurate MC based treatment plan calculation tool has been developed and validated. Three novel error prediction algorithms have been developed, two of which have been validated for homogenous geometries. In particular, the FSI shows promise as both a direct predictor of AAA error, and also as a general treatment plan complexity index. With sufficient benchmarking, these methods may be developed into a clinical tool that can identify treatment plans that are at high risk for AAA dose calculation errors.

Development of an Accurate Monte Carlo Treatment Plan Calculation Framework for the Purpose of Developing Dose Calculation Error Predictors for a Widely Implemented Clinical Algorithm

by

Alexander J Egan

A DISSERTATION

submitted to

Oregon State University

and

Oregon Health and Science University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented March 31, 2014
Commencement June 2014

Doctor of Philosophy dissertation of Alexander J Egan presented on March 31, 2014.

APPROVED:

Major Professor, representing Medical Physics

Head of the Department of Nuclear Engineering and Radiation Health Physics

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University and/or Oregon Health and Science libraries. My signature below authorizes release of my dissertation to any reader upon request.

Alexander J Egan, Author

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my major advisor and mentor, Dr. Wolfram Laub, for all of his patient guidance throughout the course of this project.

I would like give special thanks to Dr. Camille Palmer for her offering me the opportunity to study here at OSU as well as her academic support while serving as my major advisor during my first year of studies.

I would like to thank Dr. James Tanyi for both serving on my doctoral committee, as well as for his comprehensive and engaging clinical mentorship throughout my tenure as a medical physics assistant.

I would like to thank Dr. Krystina Tack for generously funding my research efforts over the past year. Without it, I would not have been able to complete this degree.

I would like to extend my appreciation to Dr. Ken Krane for taking the time to serve on my doctoral committee.

I would also like to thank the rest of the OHSU medical physics staff for all of their encouragement and support throughout this process.

And most of all, I would like to thank my mother and father, two very exceptional human beings that never lost faith in me. This is for them.

TABLE OF CONTENTS Page

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction and Literature Review

1.1 Project Motivation

1.1.1 Increasing Complexity of Radiotherapy Treatments

In order to realize positive treatment outcomes in external beam radiation therapy, target volume dose coverage must be maximized while at the same time minimizing the dose to normal tissue. During the radiotherapy treatment planning process, the radiation oncologist will delineate these target and normal tissue volumes within the CT scan-generated electron density distribution, as well as assign to each of them a set of dose constraints. Taking into account these volumetric dose constraints as well as the beam shaping and motion capabilities of the delivery machine, the treatment planning software (or system, TPS) will attempt to find the optimal balance between target volume coverage and normal tissue sparing. Once this optimization stage is complete, the TPS then executes a final dose calculation of the planned delivery in order to predict the actual dose distribution within the patient. It is this predicted dose distribution that is used to evaluate whether or not a given plan will be delivered to the patient, and it is therefore imperative that this dose calculation is as accurate as possible. It is the accuracy of this final dose calculation that is the subject of this work.

As the complexity of radiotherapy treatments increases, existing clinical dose calculation algorithms may be stressed in unexpected ways. In particular, the focus of this study is the widely implemented analytical anisotropic algorithm, or AAA for short, which was developed in the mid-1990s when much simpler treatments were the standard. During this time, only static gantry treatments with relatively low beam aperture modulation were in clinical use. There was not yet any need for a clinical algorithm that could accurately calculate dose for highly modulated fields, from hundreds of different beam angles, as is the case for modern arc therapy treatments. The time line in Figure 1.1 illustrates the evolution of radiotherapy treatment delivery and planning techniques since 1980. Of particular note is that the first arc therapy treatments were not delivered until 2001[1], which is several years after the introduction of AAA in 1996[2].

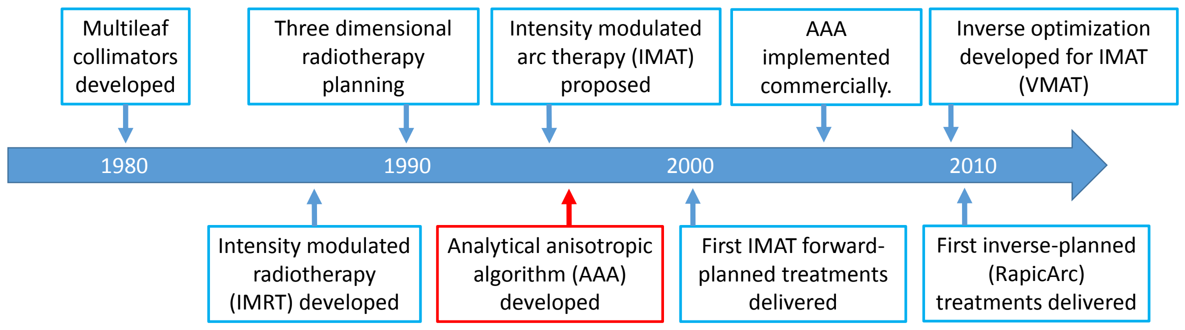

Multileaf collimators developed
Three dimensional radiotherapy planning
Intensity modulated arc therapy (IMAT) proposed
AAA implemented commercially.
Inverse optimization developed for IMAT (VMAT)
1980
1990
2000
2010
Intensity modulated radiotherapy (IMRT) developed
Analytical anisotropic algorithm (AAA) developed
First IMAT forward-planned treatments delivered
First inverse-planned (RapicArc) treatments delivered

Figure 1.1 Time line illustrating the evolution of radiotherapy treatment delivery and planning techniques.

In 2005, AAA was implemented clinically in the Eclipse [Varian Medical Systems, Palo Alto, CA] TPS. At this time patient geometry was still typically modeled as homogeneous water of constant density. Even though AAA was capable of accurately calculating dose to heterogeneous geometries, this “heterogeneity correction” can be turned off by the user such that density variations are ignored. Because clinical outcomes were at that time still associated with homogeneous dose calculations, full adoption of this more correct type of algorithm has been a gradual process. Even though the use of the heterogeneity correction has become much more widely accepted, there still exists a disconnect between its benchmarked accuracy in simple geometries[3,4], and how it performs for highly complex modern treatments.

Prior to the development of inverse arc therapy treatment optimization in 2008[5], it would have been difficult to predict the amount of increased aperture modulation that this new delivery technique would require. With the ability to now build a dose volume from hundreds of constituent beam angles, as opposed to less than ten for static gantry treatments, aperture shape complexity increased considerably. It is well understood that more aperture modulation will lead to more small field dose calculation errors[6,7]; however when many individual field dose calculations are superimposed upon one another, it is not obvious how clinically relevant the combined build-up of these errors will be.

This kind of error build-up is inherent in all algorithms that superimpose multiple constituent dose calculations. For static gantry treatments with simple beam apertures, a rudimentary understanding of how and why the dose calculation algorithm fails, coupled with the knowledge of where a failure is clinically relevant, can be used to draw useful qualitative conclusions regarding the acceptability of a specific treatment planning result. For static gantry, intensity-modulated treatments which typically utilize 7-9 beams, such conclusions are more difficult to deduce due the number and complexity of the overlapping fields. For Eclipse RapidArc treatments, up to 177 individual dose calculations, each of which can be quite complex[7–10], are overlapped in order to generate the final dose distribution.

1.1.2 Identification of Dose Calculation Errors

In clinical practice, the accuracy of the final dose calculation for a given treatment plan is often verified by comparing it to a measured dose under simplified conditions. This typically involves delivering the planned treatment to a two-dimensional detector array housed within a homogenous, water-equivalent phantom. The measured dose plane will then be compared to one calculated with the same delivery parameters but onto the simple phantom geometry. Because the patient geometry is three-dimensional and not made solely of water equivalent material, a great deal of information is never obtained. Furthermore, because of the coarse resolution of these measurement arrays, spatially confined errors may not be detected. Another distinct limitation is the lack of an exact, or even intuitive, correlation between the location of a measured deviation in the phantom, and any specific location within the patient geometry. Without this information, it is often difficult to assess the clinical significance of a given deviation.

It is therefore a useful endeavor to develop other, more robust, treatment plan evaluation tools. In particular, methods which can identify the location and magnitude of deviations between the predicted and true dose distribution will be most valuable. The inherent problem in obtaining this information is that the true dose distribution within a patient cannot be measured, at least not in any arbitrary three-dimensional way. This leaves estimating the truth as the remaining alternative. While this is exactly what the TPS seeks to accomplish during the initial planning process, it is well understood that dosimetric errors can still occur. Monte Carlo algorithms, on the other hand, are known to not suffer from the same shortcomings as AAA, and can therefore be used to identify where, and to what degree, AAA is calculating dose incorrectly.

1.2 Project Aims

1.2.1 Dose Calculation Error Detection

Monte Carlo (MC) algorithms are widely accepted as the most accurate method currently available to calculate dose in a patient geometry[11–13]. For this work the highly benchmarked[14,15] EGSnrc MC dose calculation engine was used to identify calculation errors in the widely implemented analytical anisotropic algorithm which is available for use as the final dose calculation in the Eclipse TPS.

To accomplish this, a MC based treatment plan calculation framework (MCTPCF) has been developed. This calculation framework is able to execute an EGSnrc based final dose calculation on any external beam treatment plan that was optimized in the Eclipse for delivery on a 6MV Varian Clinac. The MCTPCF inputs DICOM encoded treatment plan and patient geometry files, converts them into MC code input files, executes the dose calculation in parallel on a remote computing cluster, and then compares the results with the original AAA calculation. The core component of this calculation tool is a highly accurate MC model of the Varian Clinac 6MV beam, the development and parameterization of which is described in detail below. First, an optimal set of bremsstrahlung transport settings, as well as electron source parameters, are identified. This is followed by a careful validation of the mechanical functionality of the employed multileaf collimator (MLC) model. Mechanical validation of the MCTPCF as a whole is then accomplished through direct comparison with AAA dose calculations.

1.2.2 Dose Calculation Error Prediction

Using the knowledge of how, and under what physical conditions the AAA fails[4,7,9,10,16–18], a set of three error prediction algorithms has been developed which can identify the location and magnitude of these conditions based only on the treatment plan parameters and the patient's electron density distribution. The first metric, designated the field size index (FSI), identifies regions in the plan space where many small field errors overlap. The second, designated the heterogeneous scatter index (HSI), identifies regions within the electron density distribution where the AAA rectilinear scatter approximation is known to fail. The third, designated the low-density index (LDI) will identifies regions of very low density which are known cause AAA to overestimate dose. These prediction methods are then validated in simple geometries and ultimately applied to clinical treatment plans.

1.3 Linac Model

1.3.1 Monte Carlo Medical Accelerator Modeling

Monte Carlo (MC) modeling of radiotherapy beams has become an important tool in many areas of medical physics including beam characterization[19–24], dose-deposition studies[3,25–27], treatment planning[28–31], treatment plan evaluation[9,10], as well as routine patient quality assurance[32,33]. For a detailed historical review of the subject see Verhaegen and Seuntjens[34]. As the cost of parallel computing continues to decrease, the development of these models has become feasible to an increasingly wider range of institutions. For this reason, a comprehensive understanding of how to best parameterize a MC model of a medical linear accelerator (linac) is of continued, and likely increasing, importance to the medical physics community.

For this study, special care was taken to model the linac geometry exactly as specified by the manufacturer. This, in conjunction with the choice of a standard data set as a comparison benchmark, will allow these results to be reproduced by other researchers. In particular, this work seeks to provide a standard set of source parameters and physics settings that can aid other EGSnrc users wishing to model the 6MV Clinac beam, one the most common beams in current use. Indeed, in the early stages of this modeling process, it would have been invaluable to know exactly what geometry specifications, transport settings, and source parameters would have provided good agreement with a standard data set.

Several groups have outlined their approaches to linac photon beam modeling with MC codes[20–24,29,35–45]. Typically the components modeled will be the electron beam source, bremsstrahlung target, primary collimator, vacuum window, flattening filter, monitor chamber, lower collimating jaws, and MLC. The geometry specifications for these head components are often obtainable from the linac manufacturer; however, the electron beam source parameters are typically not provided and must be identified independently. In addition to geometry and source parameters, the transport settings specific to the chosen MC code will also have a significant effect on the model output and must be tuned accordingly.

1.3.2 Bremsstrahlung Cross Sections

Bremsstrahlung production in the target is the dominant first-order process in a linac simulation[22,23,46]. For the MC code chosen for this study, EGSnrc[15] , there are three user-selectable options for the total/energy bremsstrahlung cross section (BCS). For each option, both the total interaction probability and photon energy sampling processes are handled by the same analytic distribution or numerical data. The default BCS is a corrected form of the Bethe-Heitler (BH, analytic) differential cross section as described by Koch and Motz[47]. The user may also select the NIST (numeric) [National Institute of Standards in Technology, Gaithersburg, MD, US] cross section which was calculated by Seltzer and Berger[48]. At least two groups have studied the effects of varying between NIST and BH BCS. Fragoso *et al.*[49] reported a consistent deviation of approximately 2% in the mean photon energy at the phantom surface for a Siemens beam. Faddegon *et al.*[50] also reported significant differences in simple bremsstrahlung target simulations, finding that the NIST option yielded better agreement with experiment, though the lowest energy studied was 10 MeV. The third BCS option (NRC, numeric) was developed more recently by Tessier and Kawrakaw[51], and includes a more accurate description of the electron-electron bremsstrahlung process. They report that radiative stopping powers calculated with this new cross section can differ by up to 50% from ICRU values[52] at a few hundred keV for heavier elements. When tempered with the much larger nuclear cross section this error is reduced to 0.3% for incident energies in the 10 keV to 10 MeV range; however, it is not immediately clear whether this will have a significant effect on the MC simulation of entire linac geometries. A study of this nature has not yet been published; and indeed, to date only one MC study[53] exists in which this newer bremsstrahlung cross section was reported to be implemented.

EGSnrc offers two options for the handling of bremsstrahlung angular sampling (BAS). Both options sample the polar scattering angle from a modified version of the Koch and Motz (KM) differential cross section[47] via rejection. The default BAS option, designated “simple” samples from the first term of the distribution and the KM option from the entire modified distribution. Ali *et al.*[54] reported that MC simulations for their bremsstrahlung transmission study were significantly sensitive to the BAS setting for Pb targets and that the full KM distribution was necessary to achieve good agreement with measurement. Here again however, the lowest electron energy investigated was 10 MeV. In this study, the sensitivity of this 6MV model to all six combinations of the BCS and BAS options has been investigated.

1.3.3 Electron Source Parameterization

At least two groups have simulated the electron source by modeling the entire accelerator cavity[55,56]. However, these methods required the use of ancillary deterministic codes as well as proprietary machine specifications that may not be available to the customer. Sawkey and Faddegon[57] and De Smedt *et al.*[58] both measured linac output with some beam modifying components removed in order to infer source parameters from less complex models. These types of studies provide valuable information regarding the range of electron source parameters for a specific linac model; however, reproducing these methods on a machine in clinical use may not be feasible for many institutions. The majority of the published electron source characterization methodologies involve an iterative, trial-and-error procedure whereby a full simulation is performed for each set of parameters, the results of which are then compared to some measured data set. This methodology requires only the MC code itself, the expertise to use that code, geometry specifications, and a measured data set with which simulation results can be compared.

Electron source parameters that are typically varied are mean energy, energy spectrum, spatial intensity, and angular distribution. These parameters can vary considerably among manufacturers[31], though broadly speaking water phantom depth-dose and profile data are most sensitive to variations in the mean electron beam energy and intensity distribution[24,29,31,36,38,42,44,59].

The energy spectrum of the electron beam has been cited as having a negligible[39,46,58], to weak[38], and even significant[42] effect on the resultant dose distributions. For Varian machines at least, the energy spread is most often cited as 3% FWHM of a Gaussian distribution[29,38], though at least one group has used a spread as large as 17% and obtained good results[23]. Table 1.1 lists several Varian beam studies and their respective electron beam parameter sets. Preliminary investigations of this beam model showed a relatively small response to energy spread when compared to variation of beam divergence or bremsstrahlung transport settings; therefore this parameter was not varied.

In some early Varian beam modeling studies, pencil[22] and circular[20,21,28,35,37] electron sources were used. Most researchers, however, have made use of a Gaussian intensity distribution as prescribed by the manufacturer. Bush *et al.*[10] through an inverse iterative method, and Aubin *et al.*[55] through direct forward modeling of the accelerator cavity, both report that the most optimal distribution is non-Gaussian; however, in both cases a Gaussian model was also found to yield useful agreement with experiment. This introduces the question of how model accuracy is defined. Ideally, the most accurate representation of the geometry, combined with the most accurate representation of the transport physics, should lead to the best agreement with experiment. However, as demonstrated by the aforementioned cases, and as will be shown below, this is not the case. A choice between simulation physicality and agreement with experiment must sometimes be made.

The exclusion of beam divergence from the source model is one example of the aforementioned trade-off. There is no question that the electron beam leaving the bending magnet possesses some degree of divergence, however in at least two Varian beam model studies, a zero divergence was found to be optimal[38,42]. In contrast to these results however, preliminary work with this model showed a relatively strong dependence on this parameter and it was therefore included in the optimality search.

For this study, "the most accurate model" is identified as that which yields the best agreement with the specified data set. In accordance with preliminary investigations into which parameters have the strongest effect on depth dose and profile distributions, bremsstrahlung transport settings, as well as electron source energy, Gaussian intensity distribution FWHM, and beam divergence have been optimized. Also, so that these modeling results can be clearly understood, well-defined accuracy, uncertainty, and distribution normalization specifications are introduced and discussed.

Table 1.1 Monte Carlo 6MV Varian Clinac beam modeling studies and their respective electron source parameterizations, code type, and employed bremsstrahlung model. Parameters that were not discussed are labeled n/d.

Year	Author(s)	Clinac Model or Data Source	Energy	Energy FWHM	Intensity Dist. Model	Intensity Dist. Diam. or FWHM	Diverg.	Monte Carlo Code	Brems. Cross-section	Brems. Ang. Sampling
1995	Lovelock *et al.*[20]	600 C	5.8 MeV	2.1%*	circular	n/d	n/d	EGS4[6]	n/d	n/d
1997	Liu *et al.*[2]	2100 C	6.5 MeV	0%*	circular	2-4 mm	n/d	EGS4	n/d	n/d
2001	Fix *et al.*[4]	2300 C/D	6.05 MeV	0%*	pencil beam	-	n/d	GEANT 3.21	n/d	n/d
2001	Hartmann-Siantar *et al.*[3]	2100 C	6.2 MeV	0%*	circular	2 mm*	0°*	PERE-GRINE[17]	EEDL[57]	simple KM
2002	Ding[19]	21 EX	6.02 MeV	17%*	Gaussian	1.2 mm	n/d	EGS4	n/d	n/d
2002	Sheik-Bagheri and Rogers[23]	AAPM TG-46 data set	5.7 MeV	3%*	Gaussian	2.0 mm	0°*	EGS4	n/d	full KM†
2003	Keall *et al.*[12]	21 EX	6.2 MeV	3%*	Gaussian	1.3 mm	0°*	EGS4	n/d	n/d
2005	Cho *et al.*[6]	21 C/D	6.2 MeV	3%*	Gaussian	1.0 mm	n/d	EGSnrc[7]	n/d	n/d
2011	Chibani *et al.*[27]	21 C/D	6.3 MeV	0%	Gaussian	1.4 mm	0°	GEPTS[58]	n/d	simple KM
2014	This study	Eclipse Beam Data	5.90 MeV	0%*	Gaussian	0.15 mm	1.0°	EGSnrc	NRC	full KM

*Value was not optimized.
†Private communication.

1.4 MLC Model Validation

To reliably calculate the dose distribution produced by a radiation field modulated by a multileaf collimation system, two distinct benchmarking criterion must be met. First, the shape of the modulated dose distribution must be geometrically correct, and second, the transmission through the leaves must match that of dosimetric measurements. The first criteria requires a measurement method of high resolution, while the second requires a method of high dosimetric accuracy.

The spatial benchmarking process for a radiation beam modulating device is inherently two-dimensional. In this case, the regions of validation will be measured and simulated planes perpendicular to the direction of the beam (*z* axis). The location of the virtual device along the *z* axis is also of vital importance, however this can be verified through simple beam divergence calculations. This validation procedure is further decomposed into two perpendicular one-dimensional profile comparisons, the locations of which are carefully chosen to reveal the largest potential model errors.

To properly resolve field "edges" across the measured and simulated planes, the resolution of the employed measurement method must be sufficiently smaller than the field dimensions being resolved. Radiation measurement resolution can be defined as a length dimension of the volume over which the detector response is averaged. Ion chamber measurements are widely accepted as the most reliable, however even the smallest commonly available ion chambers have active volume dimensions on the same order as MLC leaf dimensions (projected to isocenter), making them unsuitable in this procedure. Even the ~1 mm resolution of diode or scintillation detectors will produce some level of volume averaging. Though not available for this study, the ideal choice for such high resolution measurements would be a diamond detector. The spatial resolution of radiosensitive film is effectively limited by that of the scanner used to read the film[60,61]. Radiochromic film has been shown to exhibit a much weaker energy dependence[61,62] than that of radiographic film, and across the spectrum of the modeled 6MV beam, this dependence is less than 1%[63]. For these reasons Gafchromic EBT3 [Ashland Inc. Covington, KY] film has been selected for the spatial validation of the MLC model.

Multichannel film scan analysis methods have recently been developed for absolute dosimetry with Gafchromic film[64,65]. These improved scan analysis methods utilize all three color channels of the RGB scanner to reduce the effects of thickness variations and low-frequency lateral inhomogeneities. Dosimetric accuracy of these methods will depend heavily on the care taken in calibration and scanning processes.

1.5 MC Treatment Plan Calculation

In order to evaluate the developed error prediction methods, a MC treatment plan calculation framework (MCTPCF) has been developed and validated. The development and dosimetric validation of the core linac model is discussed in other sections of this work. This section deals specifically with the mechanical operation of the calculation framework only. Meaning that, because the open field beam model has been thoroughly benchmarked, and the dosimetric accuracy of the transport code itself has been benchmarked elsewhere[14,15,66–72], MC simulation of MLC modulated fields is assumed accurate to within the accuracy threshold discussed below.

Briefly, the MCTPCF is essentially a set of MATLAB [MathWorks Inc., Natick, MA] scripts which are able to input clinical treatment plan and CT geometry DICOM files, anonymize them if necessary, and then generate the requisite BEAMnrc and DOSXYZnrc input files. Next, a set of shell scripts is used to sync these input files to the remote parallel processing environment for calculation in batch form. After the individual jobs are complete, they are summed into MC dose-distribution files and returned to the local machine for analysis. Because of the large amount of computation required, the MCTPCF has been implemented in a parallel computing environment, namely the Oregon State University [Corvallis, OR] College of Engineering High Capacity Computing Cluster

Four clinical plans, one head-and-neck, one prostate, one lung, and one esophageal, were selected for MCTPCF validation. These plans were first calculated within Eclipse [Varian Medical Systems, Palo Alto, CA] with the AAA algorithm, the results of which were used as a mechanical benchmark for the MC calculated plans. That is to say, at this stage no special emphasis was placed on small dose deviations as this benchmarking process is meant to demonstrate the MCTPCF is delivering the correct MLC apertures, in the correct coordinate system, with correct absolute calibration, etc. Indeed this was no small task as none of the code documentation specifically addressed this particular implementation of the various code components. A similar MC treatment plan calculation system, entitled the Victoria Island Monte Carlo (VIMC)[73] system, was developed by the medical physics group at the University of Victoria [Victoria, BC, CA]. This was the first such system capable of calculating intensity modulated arc therapy (IMAT) treatments (e.g. RapidArc) with the precision of the EGSnrc code. This work would not have been possible without that group's initial development of the new DOSXYZnrc dynamic arc sources[74], nor without the literature describing the VIMC implementation[9,10,33], which has served as a roadmap for the MCTPCF.

1.6 Error Prediction

1.6.1 Analytical Anisotropic Algorithm

The analytical anisotropic algorithm, commonly referred to as the AAA algorithm, is a semi-analytic pencil-beam kernel, superposition type algorithm[16,75,76]. This algorithm is currently implemented in the Eclipse treatment planning system and is seeing widespread use for many types of radiotherapy modalities including 3DC, IMRT, IMAT and SBRT.

In AAA, the treatment beam is modeled by a triple-source model consisting of a primary/focal photon source, a scatter/extra-focal photon source, and a contaminant electron source. The primary bremsstrahlung source is modeled as a point-like source on the central axis at the target plane, the spectrum of which is predetermined by MC simulations for each nominal energy. The extra-focal photon source is modeled as a Gaussian distribution of ray, or beamlet, sources at the plane of the flattening filter. This source essentially hardens the energy spectrum of the primary source according to the flattening filter thickness at each beamlet's off-axis radius. The contaminant electron source is located at the surface of the calculation volume and is modeled as a sum of two Gaussians in the lateral direction, and a sum of empirically fit exponentials along the divergent beamlet grid direction. When the AAA algorithm is commissioned for a specific linac, the free parameters of the model are optimized by minimizing a gamma index based objective function which inputs either the EBD or user measured data. A complete description of the AAA dose calculation and source modeling algorithms, as they are implemented in the Eclipse TPS, is provided in two papers by Tillikainen *et al.*[16,76].

AAA beamlets diverge from the primary source, and are arranged in a rectilinear grid whose spacing is constant at each depth plane. The beam is modulated by the collimator jaws and MLC in a binary fashion; meaning that if a beamlet is blocked, the energy transported is either completely impeded, or attenuated by a user selected global transmission factor. Edge effects are not modeled. Also excluded is the modeling of non-target generated photons that do not travel along a given beamlet, e.g. low-angle Compton photons scattered in the flattening filter. This shortcoming is to some degree corrected through the fitting of the source model parameters to measured data which imposes an energy conservation condition on the beamlet weighting. However for highly modulated, heterogeneous geometries, it is not clear whether or not these off-beamlet-axis photons require more accurate modeling.

Once a beamlet enters the dose calculation geometry, its energy is then dispersed within geometry according to a MC pre-calculated, pencil dose-deposition kernel. These kernels were calculated in water for a set of mono-energetic pencil photon beams whose energies are preselected depending on the nominal energy of the modeled beam. In homogeneous media, the dose calculation is just a convolution of each energy component of the beamlet with the corresponding pencil beam kernel. In heterogeneous media, the MC kernel must be appropriately scaled depending on the electron density, and convolution is no longer possible.

In order to transport energy accurately through heterogeneous media, the MC kernel is fit with a set of exponential models in both the lateral and beam-directed directions. Along the central axis of the beamlet, the amount of energy available for lateral transport is determined by the electron density of the geometry already traversed. At each lateral plane (perpendicular to the central axis of the beamlet) energy is then dispersed according the fitted model, but scaled by the relative (with water) density at the central axis value. This is lateral scaling is correct for a perfect slab heterogeneity, but in any other case it is an approximation. It is on this approximation that the heterogeneous scatter index (HSI) is based.

1.6.2 EGSnrc as the Dosimetric Benchmark

For this work, EGSnrc based dose calculations are considered more accurate than that of AAA in three specific contexts: a) in regions of high modulation, b) regions of lateral (to the beam direction) heterogeneities, and c) in very low density media. For this reason, the MCTPCF introduced in section 1.5 can be used as a comparison benchmark for the detection of AAA dosimetric errors.

The ability of the MC calculation to accurately transport energy through an MLC shaped field will depend primarily on the accuracy of the virtual MLC geometry and positioning. Because AAA models MLC leaves as two-dimensional binary blocks, and does not account for interleaf leakage or leaf scatter, it is clear that the MC code more accurately model MLC modulation. AAA does, to some degree, account for leaf end shape with a leaf-tip offset parameter which is applied globally to each leaf; however, this first-order approximation likely contributes significantly to modulation errors along the direction parallel to the leaf. AAA accounts for leaf transmission with a global fluence attenuation factor. While this can be a very accurate method for solid blocking, it will of course fail to model any leaf edge effects in the cross-leaf direction. Many regions of a highly modulated beam aperture will see these effects compounded depending on the leaf positions. For example, modulation error across a single leaf gap is influenced by both leaves due to their close proximity. This error near the leaf end (in the leaf-parallel direction) will also include a leaf-tip contribution, and so on.

As discussed in section 1.6.1, AAA uses a MC kernel scaling approximation that will limit its ability to properly account for heterogeneities near, but not crossed by, the central axis of the kernel. An EGSnrc pencil beam dose distribution is illustrated in Figure 1.2. Here the central axis of the beam passes near, but not across, the lower density heterogeneity. MC correctly calculates the path of every scattered particle, however AAA uses an all-water version of this kernel and rectilinearly scales it according the density along the central axis only. This results in accurate heterogeneous scaling in the beam direction, but incorrect scaling in the lateral direction for cases like the one displayed in Figure 1.2. As illustrated, scaling the kernel according the central axis value, overestimates the amount of scatter attenuation, and will therefore underestimate the amount of energy available for deposition at the calculation point.

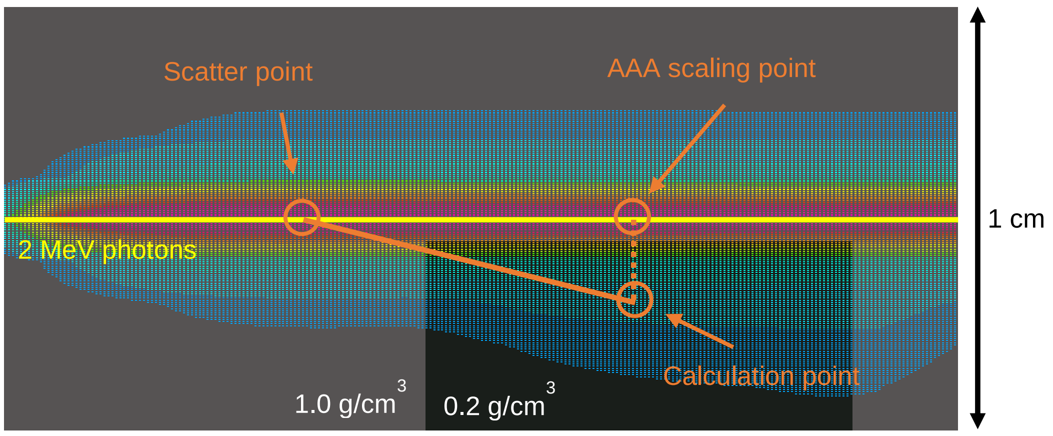


Figure 1.2 Illustration of the AAA lateral MC kernel scaling approximation.

It has been demonstrated in several studies that AAA incorrectly calculates dose in very low-density media[3,18]. This is likely due to the limited lateral extent of the MC kernel or perhaps in the lateral fitting of the kernel to a finite sum of exponentials. Whatever the reason for this shortcoming, it is almost certainly a byproduct of AAA attempting to approximate much longer path lengths in low density media. MC can accurately simulate any non-infinite path length that remains in the calculation geometry.

1.6.3 Plan Complexity and Error Prediction in the Literature

Over the past decade there have been several studies on how to quantify the complexity of intensity modulated treatments. These methods typically focus on either the heterogeneity of the geometry[77–79] of the level of aperture modulation[80–82]. For example, in a recent study by Disher *et al.*[79], the conditions under which charged particle disequilibrium is created for low density media was investigated through MC simulations. They developed a scaled percent depth-dose metric to quantify this effect, however their "relative depth-dose factor" is a parameter of an entire clinical setup and is not meant to predict local dose calculation errors. A more comprehensive approach of detecting heterogeneity induced dose calculation errors was first introduced by Pflugfelder *et al.*[78] and further developed by Bueno *et al.*[77]. Their method is similar to that presented in this work in that they developed a heterogeneity index for treatment plan evaluation; however, their index is applicable only to a proton transport algorithm. Also, the resultant aperture-specific error maps for that method are two-dimensional and therefore cannot be easily correlated with specific locations with the patient geometry. Similarly, the existent methods to assess aperture complexity are also aggregate in nature in that they also assign a single parameter to an entire beam aperture. Recently Masi *et al*[81] expanded the "modulation complexity score”, developed by McNiven *et al.*[82], for application to arc therapy treatments. This metric can quantify the amount of modulation in a given plan, however it cannot predict if actual dosimetric errors caused by that modulation are being either washed out or compounded in individual voxels. Herein lies one of the most important differences between these existing methods, and those proposed in this work. The FSI, HSI and LDI all retain a fully three-dimensional mapping to individual voxels and can therefore answer the important question of whether or not errors are indeed building up.

Chapter 2 Materials and Methods

2.1 Linac Model

2.1.1 Monte Carlo Calculations

The BEAMnrc[71] code was used to simulate a 6MV Varian Clinac treatment beam which was directed onto a DOSXYZnrc simulated water phantom. Preliminary investigations showed that the variation of total photon cross section, as well as the inclusion of electron binding energy in the Compton process, had a negligible < 1.0% effect on simulated dose-distributions, therefore these settings were left at their default values (Storm-Israel, off). Second-order processes including atomic relaxations and electron impact ionization were assumed to be negligible and were also turned off. The variation of condensed history and boundary crossing settings were considered beyond the scope of this study and were all left at their default values as listed in Table 2.1. Only bremsstrahlung settings in the accelerator head as well as electron source parameters were optimized. A complete list of the employed transport settings for both sections of the model is provided in Table 2.1.

Table 2.1 EGSnrc transport setting used for the BEAMnrc and DOSXYZnrc sections of the simulation geometry.

EGSnrc Transport Setting	DOSXYZnrc	BEAMnrc
ECUT	0.7 MeV	0.7 MeV
PCUT	0.01 MeV	0.01 MeV
Global SMAX	10^{10}	10^{10}
ESTEPE	0.25	0.25
XIMAX	0.5	0.5
Boundary crossing algorithm (BCA)	exact	exact
Skin depth for BCA	3	3
Electron step algorithm	PRESTA-II	PRESTA-II
Spin effects	on	on
Bremsstrahlung cross section	Bethe-Heitler	optimized
Bremsstrahlung angular sampling	simple	optimized
Bound Compton scattering	off	off
Compton cross sections	Klein-Nishina	Klein-Nishina
Pair production cross sections	Bethe-Heitler	Bethe-Heitler
Pair production angular sampling	simple	simple
Photoelectron angular sampling	off	off
Rayleigh scattering	off	off
Atomic relaxations	off	off
Electron impact ionization	off	off
Photon cross sections	Storm-Israel	Storm-Israel

The water phantom simulation in DOSXYZnrc used a full BEAMnrc simulation as its source. A voxel size of $0.25\times0.25\times0.5$ cm^3 inside a $72\times72\times40$ cm^3 phantom was chosen in order to match the comparison data set grid. As only a half-plane is needed for depth-dose and profile simulations, only the region $y = 0$, $x \geq 0$, and $z \geq 0$, used this higher resolution with the rest of the phantom merged into as few, much larger, rectilinear regions. This, combined with the utilization of the “HOWFARLESS” feature significantly reduced computation time when compared to a fully voxelized geometry.

In the BEAMnrc simulation, directional bremsstrahlung was used with a splitting number of 1000. For each field size the splitting radius was set to 5 cm larger than the half-field width. Radially-symmetric electron redistribution was turned on. Global range rejection was used with a set cut off value (ECUTRR=2) with the maximum allowable rejection energy (ESAVE_GLOBAL) set to 1 MeV. Range rejection was turned off in the target by setting the local ESAVE parameter equal to the electron cutoff energy of 0.7 MeV. The effect of the listed variance reduction methods on these MC results was found to be negligible (< 0.05%) through the recalculation of several fields with all variance reduction switched off.

Simulations were executed on a Linux computing cluster utilizing an open source batch-queuing system. A typical 40×40 cm^2 field size simulation required approximately 3×10^9 histories to reach the required statistical uncertainty limit of $\sigma \leq 0.5\%$. This took approximately 3 hours when distributed among a standard allotment of 120 processors, with average clock speeds of 2.7 GHz.

2.1.2 Model Geometry

Accelerator head geometry specifications were obtained from the Varian Monte Carlo Data Package (MCDP) [Varian Medical Systems, Palo Alto, CA] which is applicable to Varian Clinac 2100/2300, iX, DX, C/D, EX, and cX accelerators. Aside from some primary collimator and *x* jaw specifications, the MC geometry was modeled exactly as indicated in that document. The MLC was not modeled for the purposes of source parameter optimization.

The exact position and thickness of the primary collimator was somewhat ambiguous in MCDP drawings so a more detailed drawing was obtained from the manufacturer. Because this geometry was too complex to be modeled by the BEAMnrc component module CONS3R, the sensitivity to somewhat extreme approximations of the true geometry was investigated and found to be negligible. It is therefore unlikely that any reasonable CONS3R approximation to the true geometry will produce significant deviations in model output from that of this study.

The *x* jaws were initially modeled with face angle parallel to beam divergence and their position assigned such that the 50% penumbra width matched that of the EBD. Preliminary investigations suggested these face angles were incorrect so the exact motion of the *x* jaw was carefully modeled according to a separately obtained mechanical drawing. While these drawings are freely available to any Clinac user, it is worth noting that these minor adjustments were only important for sub-millimeter penumbra region matching, and are likely unimportant for any model intended for use with MLC modulation.

To better understand potential geometry shortcomings, a fairly rigorous sensitivity study of several component specifications was conducted. It is well understood that the three most significant contributors to surface fluence are the target, flattening filter, and primary collimator[22]; for this reason the variation of the position and thickness of each of these three components was investigated. It was found that millimeter variations of target and flattening filter location and thickness could produce significant (> 1%) variations in model output. It is also worth noting that when varying the target position within the MCDP specified tolerance, variations of more than 1% were observed in depth-dose and profile distributions. For this reason the position of the target was re-derived from a more detailed, separately obtained drawing of the full target block. The small deviation (<0.01 cm) in the 6MV target position between the two documents was found to be insignificant so the nominal MCDP value was used.

2.1.3 Data Set

To ensure the reproducibility of these results, the MC model output was compared to the widely available Eclipse Beam data (EBD) [Varian Medical Systems, Palo Alto, CA]. This data set, which is applicable to the same set of Clinac models as the MCDP, consists of depth-dose and profile data for 12 field sizes ranging from 3×3 to 40×40 cm^2, and 5 depths ranging from $D_{\max}$ to 30 cm. Cho *et al.*[24] report this data set as falling within a single standard deviation (< 1%) of the mean data of more than 50 Radiological Physics Center (RPC) [MD Anderson Cancer Center, Houston, TX] measured 6MV Clinac data sets, and is therefore endorsed by that body as legitimate reference photon dosimetry data. Here this data set has been implemented for the purpose of producing a standard set of electron source parameters and bremsstrahlung transport settings. The data were measured with a 0.6 cm diameter ion-chamber[29] and are assumed to be corrected for the chamber's effective point of measurement. For the purposes of this study, a normally distributed uncertainty of 1% is assigned to the measured data as indicated by the aforementioned RPC measurements. It is worth noting that the data set contains both full and half-field profiles, presumably due to limited water tank dimensions. In particular, the 30×30 and 35×35 cm^2 field sizes each contained both types indicating water tank motion between segments of data collection.

2.1.4 Electron Source Parameterization

As discussed, open field simulations were performed for varied electron energy E_0 = [5.6, 6.4] MeV (σ_E = 0), Gaussian intensity distribution FWHM I_0 = [0.00, 0.40] cm, and beam divergence θ_0 = [0.0, 2.2]. The definition of the first two parameters is straightforward, however the divergence parameter is not explicitly defined in the latest version of the BEAMnrc user manual[71]. From inspection of the source code, at each source position the polar angle is directly sampled from the Gaussian probability

$$p(\theta) = \frac{1}{\theta_0^2} \exp\left[-\frac{\theta^2}{\theta_0^2}\right] \tag{2-1}$$

where the resultant polar angle is assumed to be small so that the employed polar cosine can be approximated by $\sqrt{1-\theta^2}$.

To reduce computation time, source parameters were initially stepped by the coarser resolution of 0.2 MeV, 0.1 cm, and 0.4°. This resolution was then narrowed to 0.01 MeV, 0.05 cm, and 0.1° for the more optimal bremsstrahlung cross section combinations. Also, during this initial parameter characterization phase, only 3×3, 20×20, and 40×40 cm^2 field sizes were simulated. Preliminary testing showed that the combined agreement with measurement for these three field sizes was strongly representative of the entire field size set. Only for the most optimal parameterizations were the remaining 20×20, and 40×40 cm^2 field sizes then simulated.

All raw MC data were first smoothed by a one-dimensional second order adaptive Savitzey-Golay (SG) filter[83] with a maximum smoothing window width of 5 cm and a χ^2 threshold of 2. The effect of this filter was to not only remove unphysical high frequency variations in the MC distributions but to also reduce the statistical uncertainty of the smoothed points. Prior to smoothing, half-field profiles were mirrored so that the smoothing operation could be usefully performed on the central axis region. The SG smoothed distributions were then filtered further by a circularly shaped averaging filter set to the width of the ion-chamber used for the EBD measurements (6 mm). This second smoothing process simulates the "measurement" of the simulation data so that they can be properly compared the ion chamber measured data.

2.1.5 Dose Distribution Normalization

Every point in the smoothed MC profile and depth-dose distributions was normalized to the 10 cm depth value of a polynomial fit of the depth-dose curve between the values of 5 and 15 cm. There is an important distinction to be made between this kind of normalization, and that where each profile curve is normalized to its central axis value. Here, the first case is designated as type A, and the second as type B. When comparing a simulated and measured profile curve, a type B normalization scheme will not account for any deviation of its normalization value and that of the measured value. In other words, any deviation in the depth-dose curve value at the level of a selected profile must be included in the profile deviation in order to obtain the optimal parameter set. Granted this bias can be small, but almost by definition it can be on the same order as the reported accuracy of the model, as will be shown below. The normalization type for several Varian Clinac beam modeling studies is listed in Table 2.2. The EBD is distributed with a type B normalization, however since it is the benchmark set, profiles can simply be rescaled by their respective depth-dose intersections. There is the possibility that this process is masking a setup error due to water tank motion or the scanning arm not returning to the exact same central axis point, however the alternative is to use a type B normalized comparison which will almost certainly introduce systematic errors.

Table 2.2 Monte Carlo 6 MV Varian Clinac beam modeling studies and their reported model accuracy, normalization method, and Monte Carlo statistical uncertainty. Values not specifically specified are labeled n/s. Hyphenated values are meant to represent depth-dose fall-off and profile-plateau regions respectively.

Year	Author(s)	Depth-dose Fall-off	Profile Plateau	Penumbra	Norm. Type	MC Stat. Unc.
1995	Lovelock *et al.*[30]	1% of D_{10}	2-3% of D_{CAX}	n/s	B	n/s
1999	Libby *et al.*[32]	1% of $D_{6\text{-}16}$*	n/s	1 mm*	n/s	n/s
1999	Ma *et al.*[23]	2% of $D_{\max}$	n/s	n/s	n/s	≤1%
2002	Sheik-Bagheri and Rogers[33]	1.5% of D_{local}	1.4% of D_{CAX}	n/s	B	~0.7%
2003	Keall *et al.*[24]	1%† of $D_{\max}$	1%† of $D_{\max}$	1 mm	A	< 2% at $D_{\max}$
2005	Cho *et al.*[18]	1.5%‡ of D_{local}	2% of D_{local}#	2 mm	B	<0.7% – <1%
2011	Chibani *et al.*[37]	1% of D_{local}	1% of D_{local}#	0.5 mm	n/s	~0.3%
2014	This study	1.0% of D_{local}	1.3% of D_{local}	1 mm	A	<0.5%

*The percent deviation was weighted with the volume under the depth-dose curve from 6 to 16 cm. Field width < 1 mm.
†The slope of a linear fit of the $D_{\max}$ weighted percent deviation was used.
‡95% of points were within 1% of local dose, 100% of points were within 1.5%.
#Whether or not profile percent deviation is weighted by central axis (CAX) fractional dose is not specified.

2.1.6 Comparison Methods

When assigning a percent deviation between the simulated and measured data sets, the deviations in the normalized doses were weighted by the local dose as opposed to the maximum dose. Explicitly this is

$$\delta_{MC(x)} = \frac{D_{MC}(x) - D_{meas}(x)}{D_{meas}(x)} \tag{2-2}$$

where the dose values $D_{\{MC,meas\}}$ are defined at some position x. Indeed this is a distinction that is not always made clear in the literature. Table 2.2 lists several Clinac beam modeling studies and their reported deviation weighting. As is well understood, a simple difference in percent depth-dose far from the normalization point can translate to a much larger locally-weighted percent deviation in regions of lower dose. Seemingly less obvious is how significantly overall source parameter optimality can be affected by a type B profile normalization. Figure 2.1 illustrates a 30 cm depth profile of the 40×40 cm^2 field size for the optimal parameter set normalized with both methods. In the upper frame, both profiles are weighted by local dose, however a shift of more than 0.3% is clearly visible. The lower frame illustrates how locally weighting the deviation can have a much more significant effect for a type A normalization. In conclusion, type B normalizations can not only mask significant shifts in profile comparisons, they can also mask the significance of using a non-weighted profile deviation. For these reasons a type A normalization has been employed for this work.

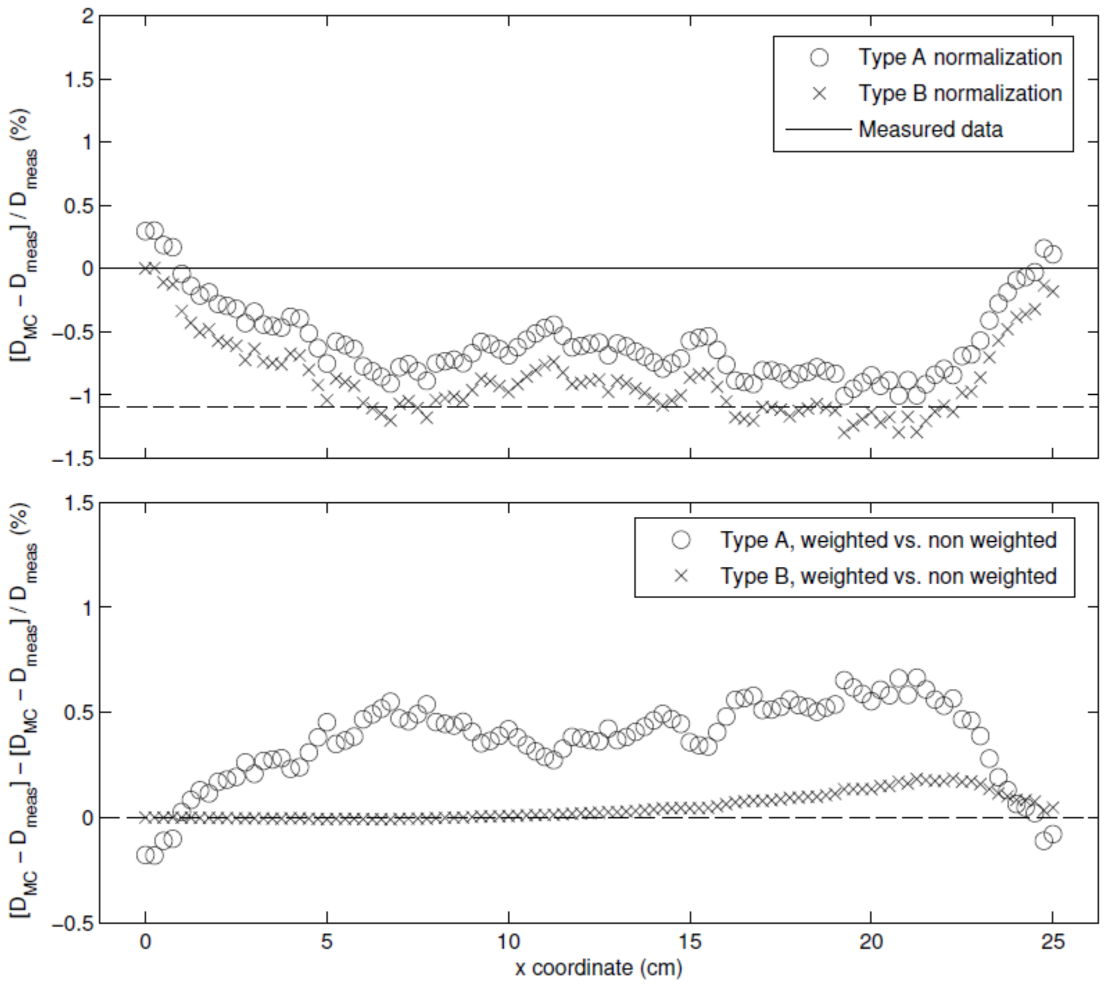


Figure 2.1 The upper frame illustrates how a type B normalization can introduce systematic errors into the comparison process. The lower frame illustrates how when using a type B normalization, significant errors may go unnoticed when not weighting deviations by local dose.

To report model quality as concisely as possible, a single accuracy threshold δ_{thresh} is assigned for the entire region-of-interest. This region was defined as all depths greater than the depth of maximum dose to avoid the introduction of surface measurement errors into the comparison analysis. Simulation results were compared with measurement in terms of percent deviation (PD) for depth-dose fall-off (FO) and profile plateau (PL) regions, and in terms of distance-to-agreement (DTA) for penumbral (PN) regions. Penumbra regions were separated from the plateau regions by applying a slope threshold to the measured data. This insured that every point in the distribution was utilized for comparison. For each of the six bremsstrahlung setting combinations, parameterization quality was mapped with the following objective function

$$\Omega_{PD,DTA}(E_{mono}, I_{FWHM}, \theta_0) = \frac{1}{S}\sum_{s=1}^{S}\left[\frac{w_{PD}}{P+1}\left[f_{FO,s} + \sum_{p=1}^{P} f_{PL,p,s}\right] + \frac{w_{DTA}}{P}\left[\sum_{p=1}^{P} f_{PN,p,s}\right]\right] \tag{2-3}$$

which sums the fraction *f* of points that meet a specific accuracy threshold. These fractions are summed first across each region $f_{\{FO,PL,PN\}}$ with weights $w_{\{PD,DTA\}}$, and then across the field size set. Here there are *P* profile depths and *S* field sizes.

2.1.7 Selected Bremsstrahlung Settings

As discussed, for a coarser parameter resolution, simulations were performed for all six bremsstrahlung transport setting combinations. This was done to demonstrate general trends in the parameters as well as to characterize the effects of the varying the bremsstrahlung settings. In order to find the most optimal parameter set for a given bremsstrahlung setting choice this resolution must be stepped down considerably. Ideally this would be done for all six settings, however even with the ample computing resources available (120 processors at any given time), this was not feasible from a time standpoint. Two of the six combinations were chosen based on a) the completeness of the physics modeled, b) previous investigations in the literature, and c) the utility of distinguishing between the two, which produced a better MC model of the beam in question.

As pointed out, the fully termed Koch and Motz bremsstrahlung angular sampling option has been found to produce simulated results which agree better with experiment than that of the lower termed formula[54]. Of the three available total cross sections, the NRC and NIST were selected a) because their physical models are more comprehensive than that of the BH cross section[15] and b) because it would be useful to determine whether the additional physics included in the NRC cross section produces any noticeable difference in a 6MV beam simulation.

2.2 MLC Model Validation

2.2.1 Varian Millennium 120

The MLC that is being modeled for this work is the Varian Millennium 120 [Varian Medical Systems, Palo Alto, CA], which is the standard out-of-the-box device for many of the Clinac models for which the open field beam model was developed. The nomenclature “120” indicates the total number of MLC leaves in the device, 60 per leaf bank. Each leaf bank has 40 0.5 cm wide leaves (isocenter projection) from y = [-20, 20], with the remaining 20 1.0 cm wide leaves on either end. Exact dimensions of the leaves themselves, as well as their location along the z axis are considered confidential by the manufacturer and cannot be discussed here.

2.2.2 The DYNVMLC Component Module

In an effort to obtain the highest level of agreement between Monte Carlo simulations and the true dose distribution, the most accurate MLC component module available in the BEAMnrc code was employed. Less complex models[84,85] have been successfully implemented, however the model developed by Heath and Seuntjens[86], labeled DYNVMLC in BEAMnrc, allows for the most accurate representation of the true MLC geometry. This model takes into account complex features such as the tongue-and-groove geometry, rounded ends, and the driving screw cavity. As part of the geometric specification process a MATLAB [MathWorks Inc., Natick, MA] script MLC_dimensions.m was written to properly calculate the MLC dimensions for input into the divergent coordinated system of the DYNVMLV module. Of particular importance is centering the virtual leaves on the central axis of the beam. Within the BEAMnrc input file, the lateral positioning of the entire MLC module is defined by the “beginning” of the first leaf, and not the center point for the whole bank. This means that to align the center two leaves on x = 0, the 30 leaf thickness, and leaf spacing thickness must be correct within ± 0.01 mm in order to prevent a buildup of misalignments at the other end of the leaf bank.

2.2.3 Film Measurements

Four MLC apertures were designed to represent the most extreme cases of beam modulation for clinically implemented IMRT. Schematics of these apertures are illustrated in Figure 2.2. Fields one and three are designed to induce errors from modulation in a single dimension, and fields two and four are designed to compound these effects by further limiting the aperture or block in the perpendicular direction. Measurements were conducted in a homogeneous solid water phantom with film placed perpendicular to the central axis of the beam, at 5 and 10 cm depth. A cross section of the phantom geometry is illustrated in Figure 2.3. As each piece of film was placed onto the solid water stack, the location of the crosshair projections were marked so that film could later be registered to the beam. 400 monitor units were delivered to each of the four fields. Calibration film measurements were taken from 50 to 450 cGy at increments of 50 cGy in order to capture the entire range of exposure.

The software package FilmQA Pro [Ashland Inc. Covington, KY] was used to scan and analyze the eight exposed films. An Epson Expression 10000XL scanner was used to scan each exposed film at 75 dpi as prescribed the film manufacturer. Note that this scan resolution is thus the effective resolution of this measurement benchmark. The triple-channel uniformity correction which was developed by Micke *et al.*[64], and has been incorporated into this software, was used to remove thickness and lateral scan heterogeneities. Scanned and processed film images were then exported from the commercial software and imported into a MATLAB analysis script profile_analyize.m.

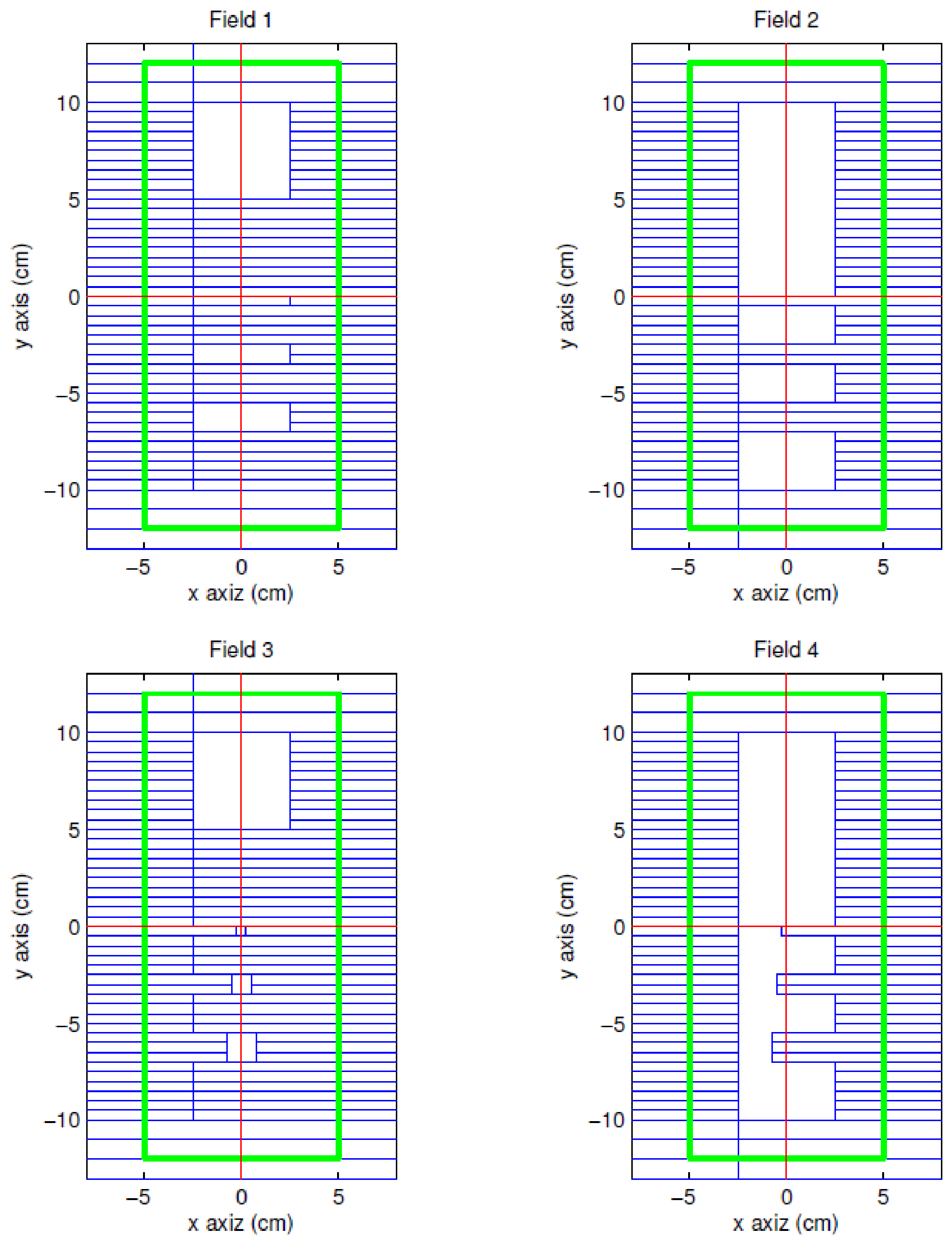


Figure 2.2 MLC apertures at isocenter used for film measurements. Green lines indicate jaw positions at isocenter. Red lines indicate the linac crosshairs.

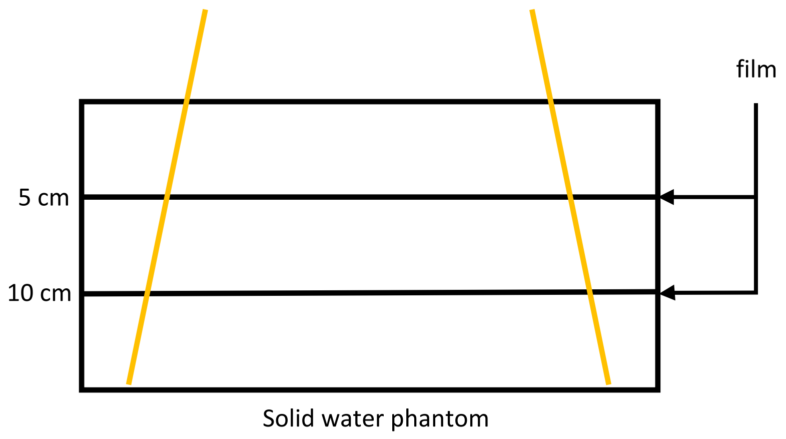


Figure 2.3 Schematic of the solid water slab phantom used for film measurements as well as MC calculations of the selected planes.

2.2.4 Monte Carlo Calculations

The solid blocks used for the film measurements were scanned in a Philips Brilliance BigBore 16-slice CT scanner in order to derive the correct Hounsfield Units (HU) for the MC simulation. The average value was found to be 90 HU which mapped to an electron density (ED) of 1.0664 g/cm^3. Because the results of the MC calculations and film measurements were also compared to a AAA calculation in Eclipse, the same HU to ED map implemented in Eclipse was used here. A DOSXYZnrc phantom geometry file (.egsphant file) was built with 0.1×0.1×0.1 cm^3 voxels centered on the xy plane at the same depths where film was placed for the measurements. Voxels in regions between these fine resolution planes were made as large as possible and the HOWFARLESS option was turned on in order to reduce computation time. Briefly, this option takes advantage of the homogeneous nature of the virtual water phantom by not calculating the distances to voxel boundaries as it is already known that no media change, and hence a path length change, will occur. The beam model developed as part of this study was directed onto the DOSXYZnrc film phantom. All transport setting were set as listed in Table 2.1. For this geometry, 10^9 histories were required in order to reach $\sigma_{max} \leq 0.5\%$. Resultant dose-per-particle distributions were scaled to absolute dose in the same manner as that discussed in

2.2.5 Comparison Tools

The MATLAB script profile_analyize.m loads the film images (exported from FilmQA Pro in .tif format) then calls film_register.m to register the images to the beam coordinate system. The registration script searches the film edges for the fiducial marks mentioned above, and used them to locate the isocenter. The film image is then shifted and rotated accordingly. An additional trial-and-error optimization search can also be called to improve the registration. Once the registration script returns a registered film image to the main analysis script, user selected MC, AAA, and film plane and profiles can be extracted from the various data stores and compared. Profiles for MC simulations and film measurement are both SG smoothed as discussed in section 2.1.4. In order for SG smoothing to be applied to the film profiles, a random uncertainty of 1% was assigned based on the evaluation of raw film responses. Both profile and planar distributions are evaluated with the common gamma[87] metric. The script gamma_calc.m implements the gamma metric employed for this work. Note that in order to reduce computation time, especially for finer resolution planes, this version only searches within the distant-to-agreement limit. This causes Gamma values greater than one to be overly biased by the percent deviation component and therefore not useful for interpretation.

2.3 MC Treatment Plan Calculation

2.3.1 RapidArc Clinical Treatment Plans

Four clinical IMAT plans, one head-and-neck, one prostate, one lung, and one esophageal, were selected for the MCTPCF validation. Each plan was calculated with the MCTPCF and then compared to the original clinical AAA calculation using the two-dimensional gamma analysis tool discussed in section 2.2.5. Simpler plan structures were implemented in the initial development stages of the MCTPCF, however one eventual use for this tool will be to reveal the superposition of dosimetric errors from plans with many component fields, thus IMAT plans have been selected for this proof-of-principle procedure. RapidArc, the Varian version of IMAT, was selected due to its wide clinical implementation, and also for its use of AAA for the final dose calculation algorithm which is the algorithm for which the error prediction metrics discussed in section 2.4 were developed.

Unlike discrete delivery angle modalities like three-dimensional conformal (3DC) or intensity modulated radiotherapy (IMRT), in IMAT the treatment beam remains on while the gantry is in motion. RapidArc treatments in particular are comprised of 177 component fields, or control points, which are developed by the TPS during plan optimization. Each control point consists of an MLC aperture, gantry angle, collimator angle, dose rate and monitor unit (MU) index. During delivery, between each control point the MLC leaves, gantry, collimator all vary at constant speeds. The AAA final dose calculation is the simple sum of 177 static-field calculations, one per control point, and weighted by its MU index. This sampling of the treatment is an obvious approximation, but as discussed by Teke *et al.*[33], it is a small one. Because a MC simulation of a particle history is independent of any other, each particle can be assigned a source angle at any sampling resolution, without any additional computational cost.

2.3.2 DICOM File Decoding

The dose calculation of a treatment plan requires the treatment plan parameters as well as the patient's electron density (ED) distribution. The first is generated by the TPS optimization engine and the second from a CT scan of the patient. In both cases, information is encoded in the Digital Imaging and Communication in Medicine (DICOM) file format. The MATLAB computing environment is packaged with some simple DICOM encoding/decoding tools which are utilized by the MCTPCF scripts anonymize.m and plan_pars.m. The first inputs radiotherapy plan, dose, structure, and CT DICOM files, and then removes all protected health information. The second is a treatment parameter extraction script for generating BEAMnrc and DOSXYZnrc input files.

2.3.3 BEAMnrc and DOSXYZnrc Dynamic Syncing

The main functionality of the MCTPCF is derived from the newly developed[74] dynamic library sharing between the SYNCVMLC component module in BEAMnrc code and source 21 in the DOSXYZnrc code. This allows particles generated by a full linac head geometry in BEAMnrc to the available on-the-fly for sourcing by the phantom code. Note that SYNCVMLC is an adaptation of the DYNVMLC component module that is equivalent geometrically, but has the ability to sync MLC modulation in the BEAMnrc code with DOSXYZnrc source parameters, such as gantry angle. From here on, the term SYNCVMLC will be used to describe the implemented module. The syncing is handled essentially by attaching an MU index to the phase space information for each particle produced by the BEAMnrc simulation which DOSXYZnrc then uses to select a control point from which to extract the required source information.

2.3.4 BEAMnrc and DOSXYZnrc Input File Generation

This whole process requires four main user adaptable ASCII file inputs, a BEAMnrc input file (.egsinp) with an accompanying MLC position file (.mlc), and a DOSXYZnrc input file (.egsinp) and an accompanying geometry file (.egsphant). The parameterization of the main BEAMnrc input file is essentially the final product of the work described in the linac modeling sections, with addition of the MLC specifications discussed in section 2.2.2. Within the SYNCVMLC specifications (within the BEAMnrc input file), the MLC position file is referenced. This MLC file is basically an ASCII lookup table for each leaf position, for each control point. The DOSXYZnrc input file list the location of the .egsphant geometry file as well as positioning information for the linac source for each control point. These parameters are isocenter position, distance from isocenter to the source plane, gantry angle, table angle, and collimator angle. Note that this file also contains its own set of transport physics settings which only apply to the transport in the phantom geometry.

The phantom geometry file is based on the CT scanned ED of the patient. Because EGSnrc employs atomic number dependent physical models, material type, in addition to ED, is required for each voxel. Because the only information available is HU per voxel, each must be calculated from empirically developed ED and media definition threshold maps. For this work the script CT_ramp.m converts HU to ED and has been parameterized with the HU to ED mapping utilized by the instance of AAA used for this study. Media mapping is handled in the phantom building script CT_prep.m (called by plan_prep.m) which has been parameterized with the ED to media definition listed in the DOSXYZnrc reference manual[72]. For the simulations in this work four types of media are mapped onto the HU grid: air, ICRP bone, ICRP tissue, and ICRP lung.

Template BEAMnrc and DOSXYZnrc input files are modified by the RP_prep.m script which extracts the required treatment parameters from the DICOM plan file via the plan_pars.m script. This script also handles the writing of the .mlc file. A schematic of the input file creation and subsequent cluster submission process is illustrated in Figure 2.4. Once a set of input files is generated, they are then transmitted to the remote cluster network. In the current implementation this transfer is handled with secure-shell based “rsync” program.

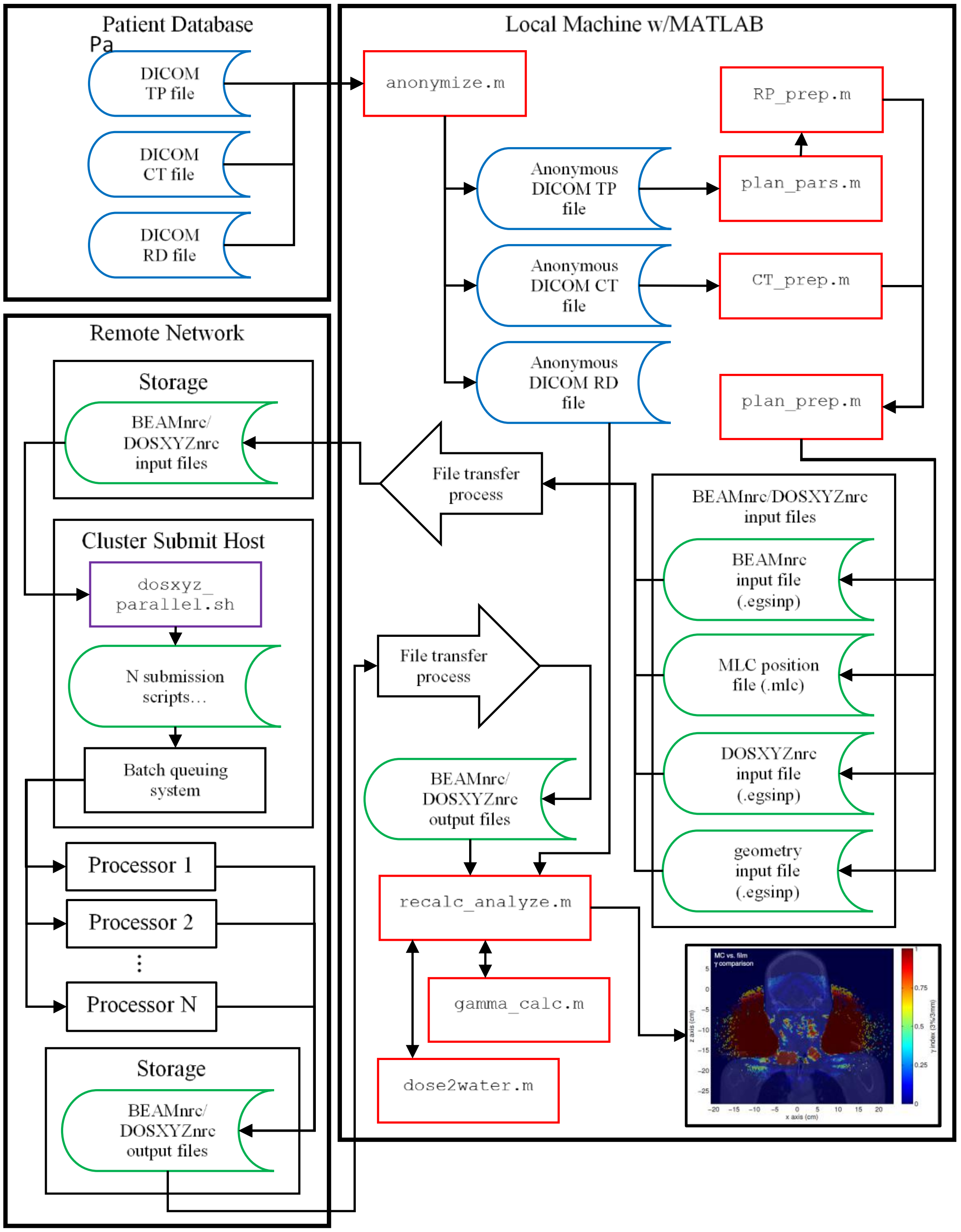


Figure 2.4 Data and code work flow chart for the MC treatment plan calculation framework. Thick black lines delineate network independent locations, and thin black lines delineate separate machines within those separate locations. DICOM encoded data objects are outlined in blue, and ASCII in green. MATLAB code is outlined in red and shell scripts in purple.

2.3.5 Cluster Submission and Extraction

Calculating a single RapidArc arc can take as long as 800 CPU hours to reach 0.5% statistics in the target volume. For this reason, all clinical plan calculations are submitted to the OSU computing cluster. For this work, 120 to 300 processors were typically available making the average computation time for a 2 arc plan in neighborhood of 6 to 14 hours. BEAMnrc and DOSXYZnrc handle parallel distribution of the simulation histories in a pseudo-batch like manner, meaning there is some cross-talk among the running jobs; however this is not encoded into the executables themselves. When a series of jobs is sent to the queuing system, the execution of the first job spawns an ASCII .lock file that acts as a history ledger for the entire batch of jobs. As each instance of the executable, in this case the DOSXYZnrc core binary, completes its sub-batch of histories, it “checks-in” with the .lock file to see if there are any particles left to run. This maximizes efficiency as simply dividing the entire number of histories evenly among the jobs may lead to a) some jobs finishing early and thus no longer participating in the computation, or more importantly b) a failed job removing a statistically significant number of particles from the overall simulation tally.

In order to efficiently execute EGSnrc codes in parallel on a networked file system, a complete installation must be placed on a machine as close as possible, network-wise, to the submit and execution hosts. When submitting hundreds of jobs, with each needing access to the same .lock, network lag can cause many jobs to fail. The OSU cluster uses a “scratch” volume designed especially for this purpose. In fact the first function of the dosxyz_parallel.sh job submission shell script is to sync an entire bare-bones EGSnrc installation over the scratch volume where the submission process will actually take place.

In the process of syncing all input, binary, and transport data files to the scratch volume, the submission shell script dosxyz_parallel.sh updates the .mlc and .egsphant file paths to match their new locations. It then executes its main functions which are to a) build a submission script for each parallel job, b) submit each script to the queuing system, and c) build a cleanup script for collecting important outputs once the jobs have finished running. One important feature of these cleanup scripts is that they will detect if the last parallel job was able to sum all the individual dose-distribution output files. If not, it will execute resum.sh which will “manually” sum them. As this is a common occurrence, and otherwise the simulation results would be discarded, this is a valuable functionality.

2.3.6 Absolute Dose Calibration

The monitor chamber charge collection in a physical linac is affected by radiation backscattered from the collimating jaws causing the amount of charge collected per MU to vary with field size. A virtual linac on the other hand, is metered by the electron source fluence which cannot be properly scaled to monitor units unless this backscatter is taken into account. To accomplish this, the ratio between the dose-per-particle (*dpp*) to the air inside the virtual monitor chamber for the delivery jaw setting $D_{ch,dpp}(X \times Y)$, and that for a calibration field $D_{ch,dpp}(10 \times 10)$, is used to scale the number of particles-per-MU N_{MU}. N_{MU} can be calculated by taking the ratio of the dose at the calibration point under calibration conditions $D_{abs}(x_{cal}, y_{cal}, z_{cal})$, and the simulated *dpp* at the calibration point under calibration conditions $D_{dpp}(x_{cal}, y_{cal}, z_{cal})$. In terms of simulated absolute dose-per-MU at a given point $D_{dpu}(x, y, z)$, this is

$$D_{dpu}(x,y,z) = D_{dpp}(x,y,z)\frac{D_{ch,dpp}(10 \times 10)}{D_{ch,dpp}(X \times Y)}\frac{D_{abs}(x_{cal}, y_{cal}, z_{cal})}{D_{dpp}(x_{cal}, y_{cal}, z_{cal})} \tag{2-4}$$

where $D_{dpp}(x, y, z)$ is the *dpp* at that point. A complete formalism is laid out by Popescu *et al.*[88]. In the MCTPCF, the output analysis script output_analyze.m automatically extracts the monitor chamber dose from BEAMnrc output files and calculates the proper number of particles using this method.

2.3.7 Dose to Water Considerations

It has been pointed out by several authors[9,10,89,90] that it is necessary to properly scale the MC dose-to-medium D_{med} to dose-to-water D_{wat} in situations where the value it is being compared to is expressed in terms of D_{wat}. Many TPS dose calculation algorithms, including AAA, express their calculation results in terms of D_{wat}, meaning they do not take into account the differences in stopping powers for non-water materials the way that a MC calculation does naturally. Siebers *et al.*[89] describe a protocol in which a single correction factor which depends only on nominal beam energy and material type can be applied to convert MC D_{med} to D_{wat}. In their study, spectrum averaged stopping power ratios were calculated for a range of radiotherapy energies. Their results show a weak very energy dependence which makes a linear scaling of dose for different materials a reasonable method for calculating D_{wat} from D_{med}. This linear scaling method is implemented in the script dose2water.m with scaling values taken from ref. [89]. For MC/AAA comparisons in this work, MC D_{wat} is calculated in this manner. This adds a degree of consistency in the comparison process as it removes the media-type dependency from the energy deposition process, thus isolating transport-only induced deviations. Alternatively, the AAA dose could also be scaled to D_{med} in the same manner.

2.3.8 Comparison Tools

The primary output analysis tool is the script output_analyze.m which extracts the user selected MC and AAA dose distributions for the selected patient, makes sure they are aligned to the same grid, calibrates the MC dose, permutes the dose distributions to the desired viewing coordinate system, applies a gamma calculation for profile and planar comparisons, and finally displays the results in the user selected imaging mode. All MC phantoms used for this work were built to match the calculation grid used for the AAA comparison calculations, namely 0.25 cm cubic voxels, in order to minimize potential comparison interpolation errors. Three comparison/imaging modes are available: profile, planar, and 3D. A gamma comparison is applied to the profiles and planar comparisons, however 3D gamma functionality has not yet been developed.

2.4 Error Prediction

2.4.1 Field Size Index

It has been shown that AAA can fail to adequately model fields with at least one dimension smaller than 3 cm[3,7,90,91]. This is understandable since AAA is often configured with only the EBD set whose smallest field size is 3×3 cm^2. Even if the AAA beam configuration extrapolates output factors to smaller field sizes, the model would have to be fairly complex as output becomes increasingly nonlinear in this range[92–94]. Indeed, for homogeneous media, both Ong *et al.*[7] and Gagne *et al.*[9] have shown that that for MLC leaf gaps ≤5 mm AAA can underestimate output by up to 10%.

In the context of beam modeling, there are typically two major sources for small field discrepancies: a) lack of lateral charged particle equilibrium[6,95], and b) source occlusion. Lack of charged particle equilibrium in homogeneous media is not an issue for AAA as the MC generated dose-deposition kernel inherently takes these effects into account. Source occlusion on the other hand can be, if the AAA spot size is left at its default setting of zero width, as it is for this work. For example, in a Varian Clinac where the MLC is located at approximately 50 cm from the target, a 2.5 mm target/isocenter (Millennium 120 MLC) leaf gap is meant to produce a 5 mm wide field at isocenter. Projecting this field/aperture back onto the target yields an effective point source; however, it is well understood that a megavoltage bremsstrahlung source has real area[43,96].

Volume averaging effects are also significant for small fields and blocks. Because the most common AAA calculation resolution (0.25×0.25×0.25 cm^3) is a significant fraction of a leaf width, the peak and trough doses can be averaged off[7,9]. Indeed a simple experiment that illustrates this effect can be carried out in any clinic with the Eclipse TPS simply by changing the calculation resolution from 0.25×0.25×0.25 cm^3 to 0.1×0.1×0.1 cm^3, and comparing dose profiles across single and double leaf gaps and blocks. Figure 2.5 displays a cutout of Figure 3.8 and illustrates this effect. Note that this kind of volume averaging is algorithm independent. Here the MC profile was calculated at a resolution of 0.1×0.1×0.1 cm^3, but when the extracted profile is linearly interpolated to the 0.25 cm *y* axis grid of the AAA profile, some of the peak is lost.

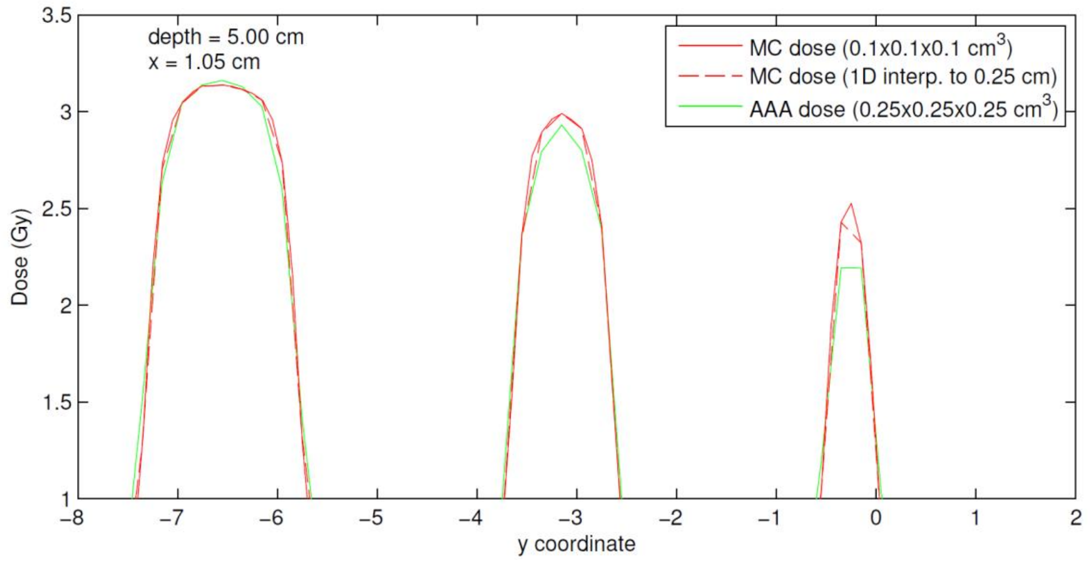


Figure 2.5 Cutout of Figure 3.8 showing the volume averaging effects of AAA for (left-to-right) triple, double, and single leaf gaps at a calculation resolution of 0.25×0.25×0.25 cm^3. The original MC calculation resolution is 0.1×0.1×0.1 cm^3. A 1D linear interpolation of the MC profile onto the 0.25 cm *y* axis grid of the AAA profile has been added to display simple geometric volume averaging.

The fact the AAA dose is still lower would seem contrary to the discussion of source occlusion; however, it may simply mean that volume averaging, or perhaps some artifact of beamlet discretization (also set at 0.25×0.25 cm^2 at isocenter), is the dominant effect. There is also the possibility that the Gaussian scatter source, which does have considerable width, is softening the penumbra. In any case, the discrepancy is clearly dependent on the overlapping of field edge penumbra and it is on this phenomena that the field size index is based.

The FSI uses the MLC aperture for a given field or control point in order to locate within the patient geometry which regions are exposed to small blocks or openings. The FSI for a given pixel of the MLC aperture at the isocenter plane is

$$FSI_{i,j} = \begin{cases} f_{FSI}\left\|\nabla_{i,j}M\right\|, & and M_{i,j} = 1 \\ -f_{FSI}\left\|\nabla_{i,j}M\right\|, & and M_{i,j} = 0 \end{cases} \tag{2-5}$$

where *M* is a binary mask of the MLC aperture, i.e. open pixels are set to one and blocked pixels are set to zero. The magnitude of the gradient is used to capture the “amount” of penumbra contained in pixel, and the binary mask is used to assign either a positive or negative FSI scaling factor f_{FSI} to account for the predicted AAA deviation. An example binary map and two-dimensional FSI map for a control point aperture from a prostate

RapidArc plan are displayed in Figure 2.6. For the FSI map, background gray identifies zero field size induced error, regions inside (unblocked) the field edges show a positive value, and regions outside (blocked) the field edges show a negative value. Note that a heavier weight is applied to inside corner regions due to contributions from a leaf edge and a leaf tip, and also to single leaf gaps and blocks due the close vicinity of aperture gradients. Once the two-dimensional map is calculated, it is then projected along the divergent beamlet grid. Any rotations due to gantry or collimator angle are then applied. This index calculation is currently implemented in the FSI_calc.m script which is called by the parent error prediction script error_predict.m.

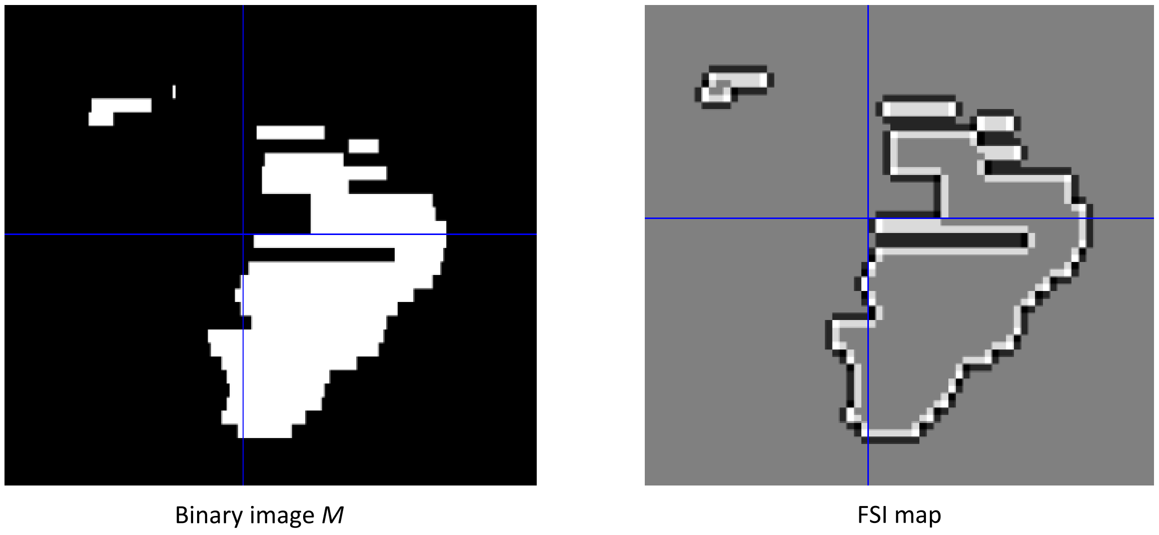


Figure 2.6 Example binary map and two-dimensional FSI map for a control point aperture from a prostate RapidArc plan.

2.4.2 Heterogeneous Scatter Index

In AAA, heterogeneities are handled by first density scaling the kernel along its axis and then rectilinearly, along the scatter profile. As discussed in section 1.6.2, the degree to which the kernel is laterally scaled depends on the media density along the central axis of the kernel only, and not on that of the surrounding regions. From a physical perspective, this is clearly not the way energy is transported at radiotherapy energies. In reality photons are scattered at primarily small angles at some significant distance upstream from the energy deposition point. If these scatter rays pass through a different density path than that of the central ray, the AAA MC kernel scaling is not properly modeling the energy transport.

Deviations from either MC or measurement, under the conditions described here, have been identified in by several authors. Esch *et al.*[4] compared film measurements with AAA calculated beam profiles near cork/solid-water interfaces. They found significant deviations within a centimeter of the vertical interfaces as well as for homogeneous profiles just below the end of the interface. In another study, Fogliata *et al.*[3] compared AAA to MC calculations, and also showed significant deviations within a centimeter of vertical lung/water interfaces. Preliminary studies for this work showed similar results.

The HSI seeks to quantify the difference in density scaled path length among the scattered energy rays at any given scatter point along the path of the AAA beamlet. Because the linac beam diverges from a point-like source, the HSI calculation will be performed in beamlet space, meaning that the entire electron density (ED) distribution must first be interpolated onto the divergent beamlet grid such that indexing along grid columns is equivalent to moving along a divergent beamlet. This mapping

$$\rho_{i,j,k} \longmapsto \rho'_{i,j,k} \quad \Longrightarrow \quad \left(x_j \frac{SAD + y_i - y_{iso}}{SAD}, y_i, z_k \frac{SAD + y_i - y_{iso}}{SAD}\right) \longmapsto \left(x'_{i,j}, y_k{}', z'_{i,k}\right) \qquad \text{(2-6)}$$

where the y axis is parallel with central axis of the linac beam, will stretch the voxel lengths $v_{j.k}$ along the beamlets depending on their off-axis polar angle. The HSI is calculated point-wise along each beamlet $\beta_{j,k}$. For each primary scatter point $p_{i,j,k}$ in $\beta_{j,k}$, a subset set of voxels

$$b_{i,j,k,l,m,n} \equiv \rho'_{i+l,j+m,k+n} \text{ for } \begin{cases} i = [1 \ldots L] \\ m = [-M \ldots M] \\ n = [-N \ldots N] \end{cases} \qquad \text{(2-7)}$$

designated the "scatter box", is extracted from the ED distribution. This scatter box is interpolated in the same way as $\rho'_{i,j,k}$ but onto a grid divergent from the scatter point. This mapping

$$b_{i,j,k,l,m,n} \longmapsto b'_{i,j,k,l,m,n} \quad \Longrightarrow$$

$$\left(x'_{i+l,j} + \left(x'_{i+L,j+m} - x'_{i+L,j}\right)\frac{y'_{i+l} - y'_i}{y'_{i+L} - y'_i}, y''_i, z''_{i+l,k} + \left(z'_{i+L,k+n} - z'_{i+L,k}\right)\frac{y'_{i+l} - y'_i}{y'_{i+L} - y'_i}\right) \longmapsto \qquad \text{(2-8)}$$

$$\left(x''_{i+l,j+m}, y'', z''_{i+l,k+n}\right)$$

allows the ED along a scatter ray to be indexed along the l dimension of $b'_{i,j,k,l,m,n}$. The attenuation factor at each point in $b'_{i,j,k,l,m,n}$ is then

$$A_{i+l,j+m,k+n} = \exp\left[-\mu_{i,j,k} v_{j.k} \sum_{l'=0}^{l} \rho_{i+l',j+m,k+n} - \rho_{i+l',j,k}\right] \tag{2-9}$$

where $\mu_{i,j,k}$ is a depth dependent attenuation parameter. The lateral weighting of the rays is modeled with a Gaussian distribution off-axis distance. The width parameter $\sigma_{i,j.k}$ is also depth dependent as will be discussed below. The final form of the HSI is

$$HSI_{i+l,j+m,k+n} = \left[\tau_{i,j,k} \exp\left[\frac{-\left(x''_{i+l,j+m} - x''_{i+l,j}\right)^2}{2\sigma_{i,j,k}^2} + \frac{-\left(z''_{i+l,k+n} - z''_{i+l,k}\right)^2}{2\sigma_{i,j,k}^2}\right] \left[A_{i+l,j+m,k+n} - 1\right]\right] \tag{2-10}$$

where the scaling factor $\tau_{i,j,k}$ is primary photon energy fluence, or terma, dependent. A diagram of the HSI scatter kernel is illustrated in Figure 2.7. By subtracting one from the attenuation factor, the HSI takes the form of a signed fractional correction to the scatter energy released from the scatter point and also zeroes the index in homogeneous media. Note that there is a small approximation made in using a displacement distributed Gaussian as opposed to an angular one. As the off-axis polar angle of the beamlet is increased, constant increments in off-axis distance will lead to a reduction in “scatter” angle increment. This computationally expensive trigonometric relationship scales with the square of the cosine of the beamlet polar angle so it is thus taken as negligible. The depth dependent terma factor $\tau_{i,j,k}$ weights the HSI along the beamlet and is derived from fitting a sum of exponentials to MC simulated pencil beam terma in water. Fitting methods for $\mu_{i,j,k}$ and $\sigma_{i,j.k}$ are still being developed. This index calculation is currently implemented in the HSI_calc.m script which is called by the parent error prediction script error_predict.m.

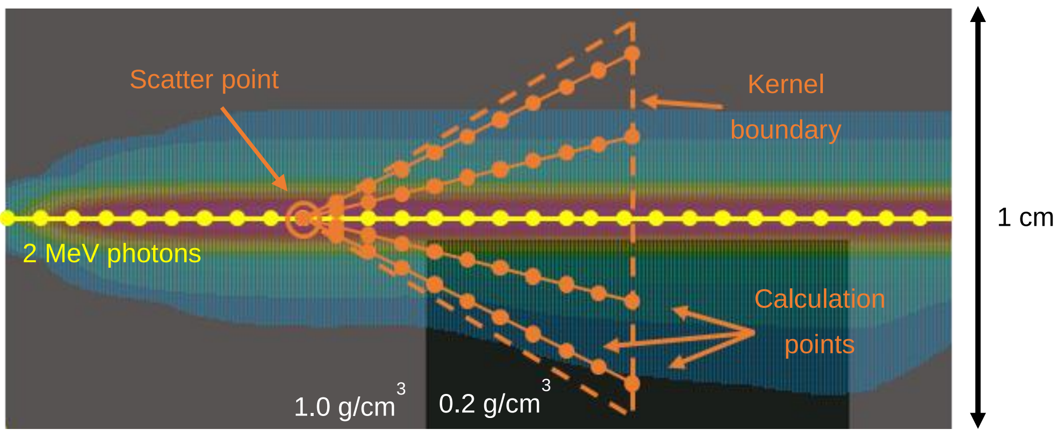


Figure 2.7 Diagram of the HSI kernel.

2.4.3 Low-density Index

It has been shown that although AAA transports energy effectively through very low density media, its deposition in such media is overestimated[3,18], seemingly due to forced energy conservation combined with the limited lateral extent of the pencil beam kernel. The LDI thus maps the fractional deviation between MC and AAA for simple open fields for densities less than 1.0 g/cm^3. To develop this mapping, open field simulations (100 cm SSD, 10×10 cm^2 field size) for both algorithms were carried out on a water phantom with the electron density varied from zero to 1.0 g/cm^3. For each algorithm, the mean doses in a 2×2×2 cm^3 volume centered at 10 cm depth on the central axis were fit with a linear plus exponential model, the ratio of which is the low-density index

$$LDI_{i,j,k}(\rho_{i,j,k}) = LDI_0 + LDI_1\,\rho_{i,j,k} + LDI_\alpha \exp[LDI_\beta\,\rho_{i,j,k}] - 1 \qquad (2\text{-}11)$$

A diagram of the simulation geometry is illustrated in

Figure 2.8. Due to its simple nature and fast calculation, this index is currently implemented in the beginning of the parent error prediction script error_predict.m.

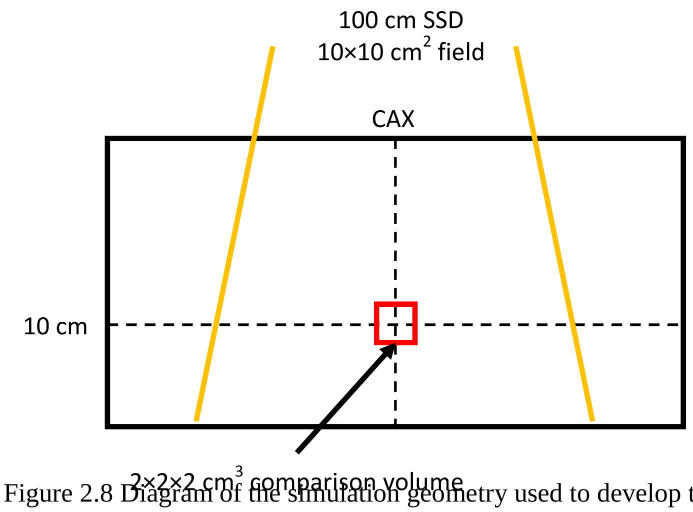


Figure 2.8 Diagram of the simulation geometry used to develop the LDI.

Chapter 3 Results and Discussion

3.1 Linac Modeling

3.1.1 Trends in Parameter Optimality

Figure 3.1 shows the variation of electron source parameters for the six possible bremsstrahlung transport settings. For these open field simulations, matching criteria was set to 1.5%/1.0 mm and the aforementioned objective function was summed over 3×3, 20×20, and 40×40 cm^2 field sizes. Regions of interest were weighted w_{PD} = 0.75, w_{DTA} = 0.25, such that objective function values approach a value of one. Heavier emphasis was placed on the depth-dose fall-off and profile plateau regions as it was found that penumbra matching becomes noisy for poorly matching parameterizations, which in turn produces noisy trends in figures similar to Figure 3.1. From inspection of Figure 3.1 it is clear that source parameter optimality depends significantly on both the divergence parameter as well as the choice of bremsstrahlung transport settings. Also of note is the loose convexity of the objective function values. This is suggestive that the most optimal set of parameters do indeed lie within the ranges simulated.

3.1.2 Electron Source Parameterization

To identify the most optimal source parameterization for the selected bremsstrahlung combinations, the search resolution was increased to 0.01 MeV, 0.05 cm, and 0.1°. Figure 3.2 and 3.3 illustrate the variation of source parameter optimality for the NRC/KM and NIST/KM combinations respectively. The maximum accuracy attainable at a resolution of 0.1% was 1.3%/1.0 mm for the NRC/KM and 1.5%/1.0 mm for the NIST/KM case. The circled points in each figure indicate the single parameterization that passed for all 5 field sizes, namely 5.90 MeV, 0.15 cm, and 1.0° for the NRC/KM and 5.90 MeV, 0.15 cm, and 0.8°. Further reduction of the accuracy threshold at an increment of 0.1% (DTA was not varied) did not yield a fully passing parameterization. Figure 3.4 displays the depth-dose and profile curves for the most optimal parameter set for the 40×40 cm^2 field size, and Figure 3.5 displays their percent deviation from measurement.

Within the existing literature there is a general consensus on the strong effect of electron energy and intensity distribution, however no published Clinac studies report employing a beam divergence for the 6MV Clinac beam (see Table 1.1). The strong dependence of this model output on this parameter is clearly illustrated in Figure 3.1, 3.2, and 3.3. Sawkey and Faddegon[97] make sound arguments as the to the unphysicality of divergences on the order of 1.0°, however here the concern is only with the physicality of the resultant dose distributions. In addition, more than one author[55,98] has suggested unphysical, or at least non-Gaussian, intensity distributions yield the best agreement with experiment. In conjunction with the results of this study, this suggests that either through failures in geometry modeling, or perhaps in the transport code itself, the link between a physical electron source and the measured data set is to some degree lost. Again however, for the purpose of a dosimetrically accurate MC beam model, the physicality of the source is not of primary concern.

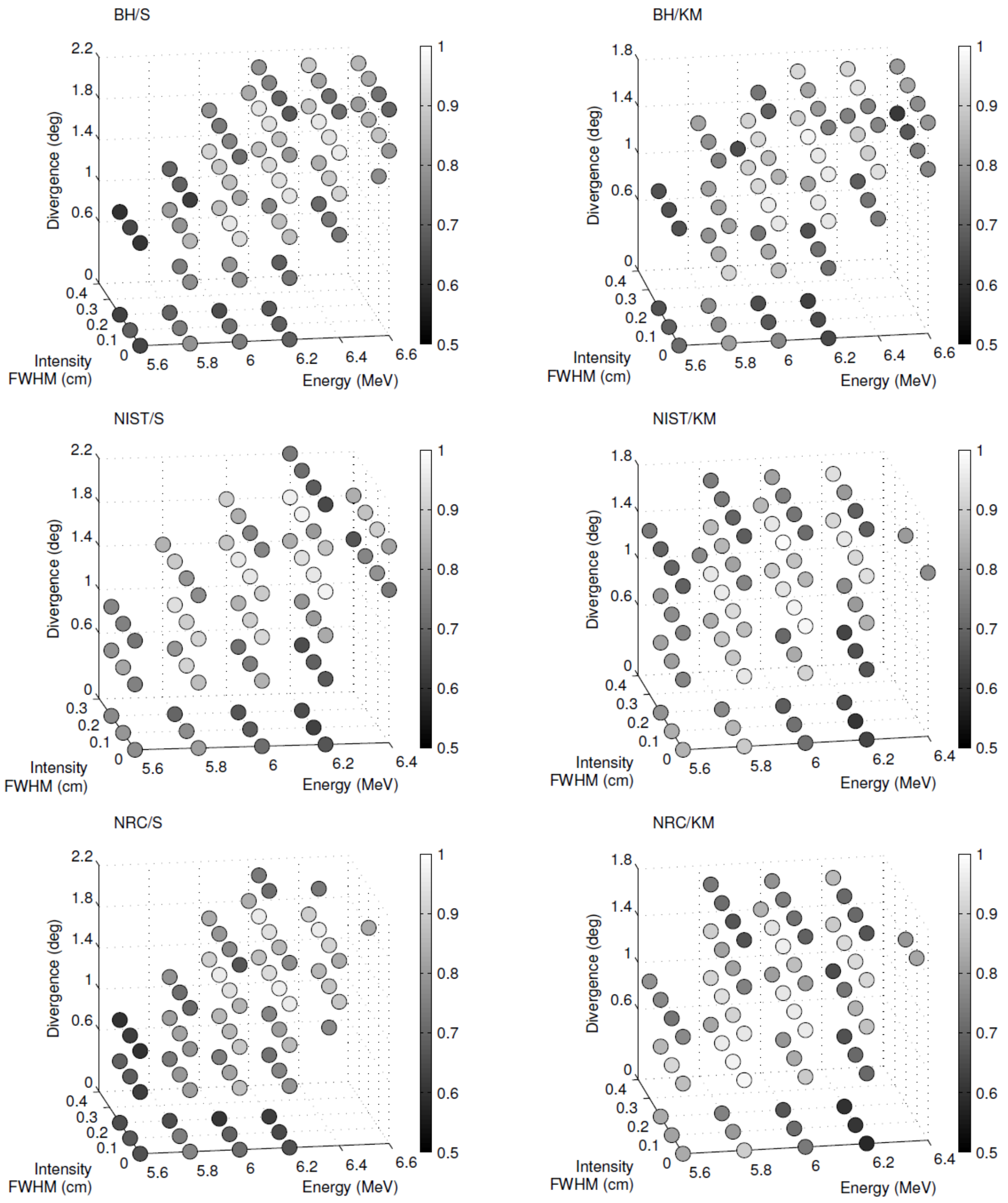


Figure 3.1 This set of figures illustrates trends in the optimality of source parameterizations for each of the six bremsstrahlung cross section combinations. Here the objective function accuracy threshold is set to 1.5%/1.0 mm with weights w_{PD} = 0.75, w_{DTA} = 0.25, and w_{χ^2} = 0.

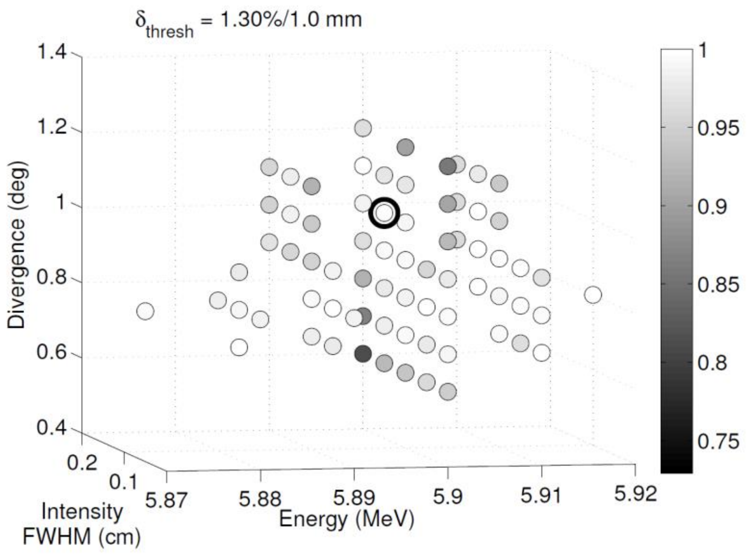


Figure 3.2 Variation of electron source optimality for the NRC/KM bremsstrahlung setting. The circled point indicates the most optimal parameterization (5.90 MeV, 0.15 cm, and 1.0°).

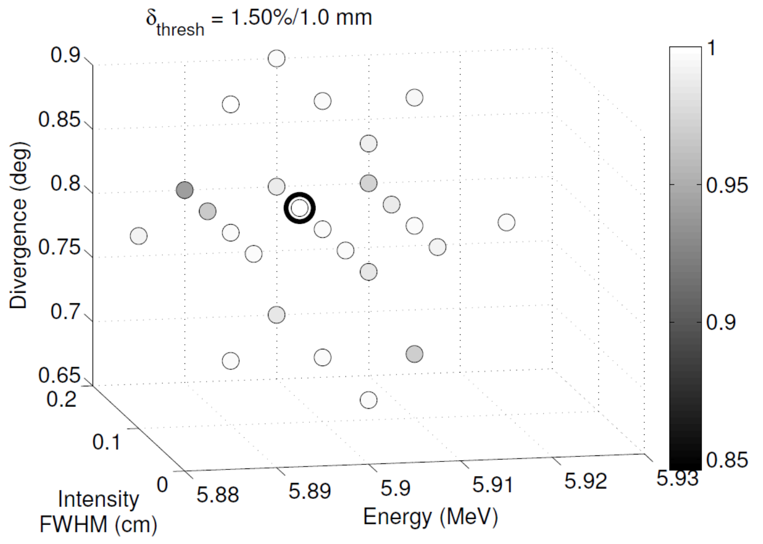
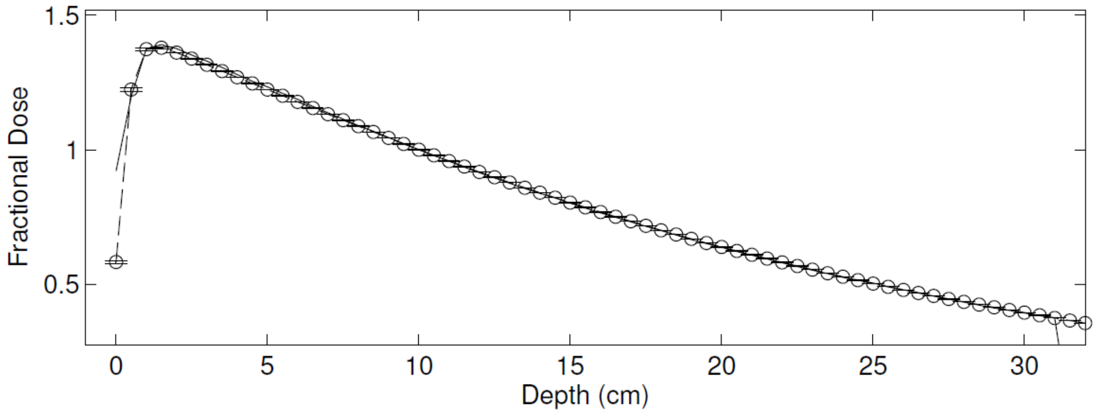


Figure 3.3 Variation of electron source optimality for the NIST/KM bremsstrahlung setting. The circled point indicates the most optimal parameterization (5.90 MeV, 0.15 cm, and 0.8°).



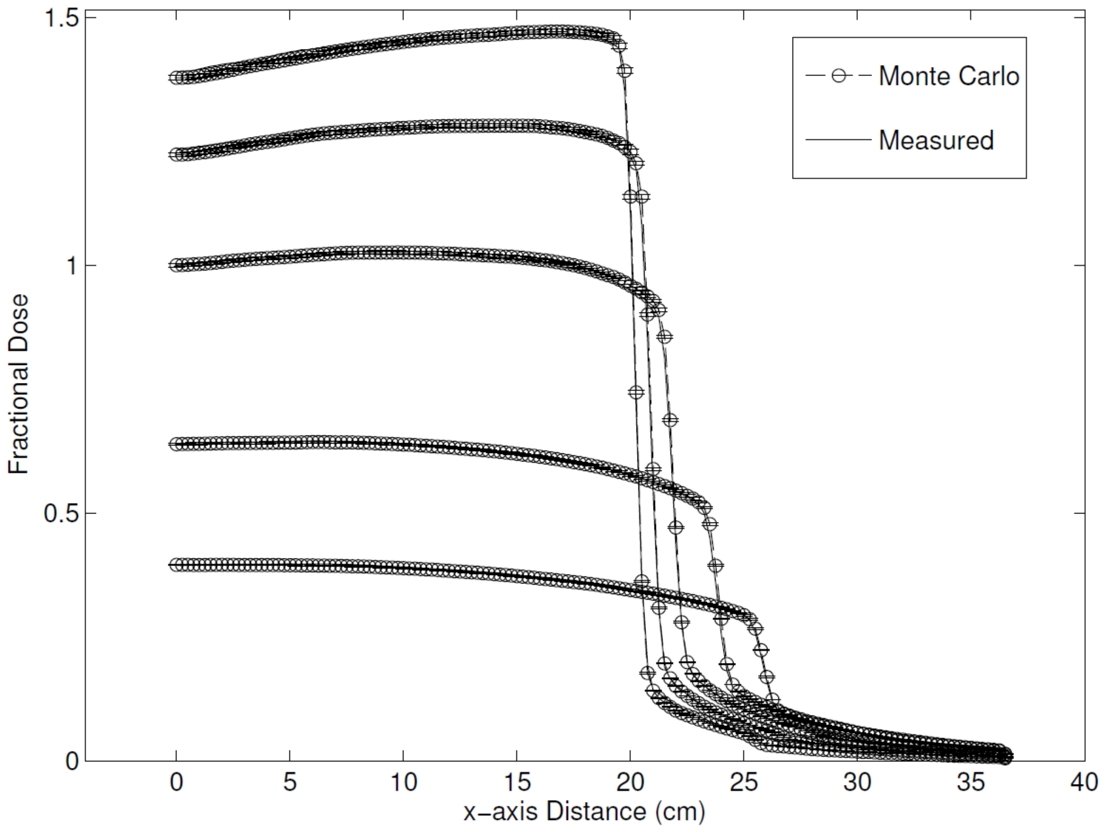


Figure 3.4 Depth-dose and profile curves for the most optimal parameter set, NRC/KM bremsstrahlung combinations, 40×40 cm^2 field size.

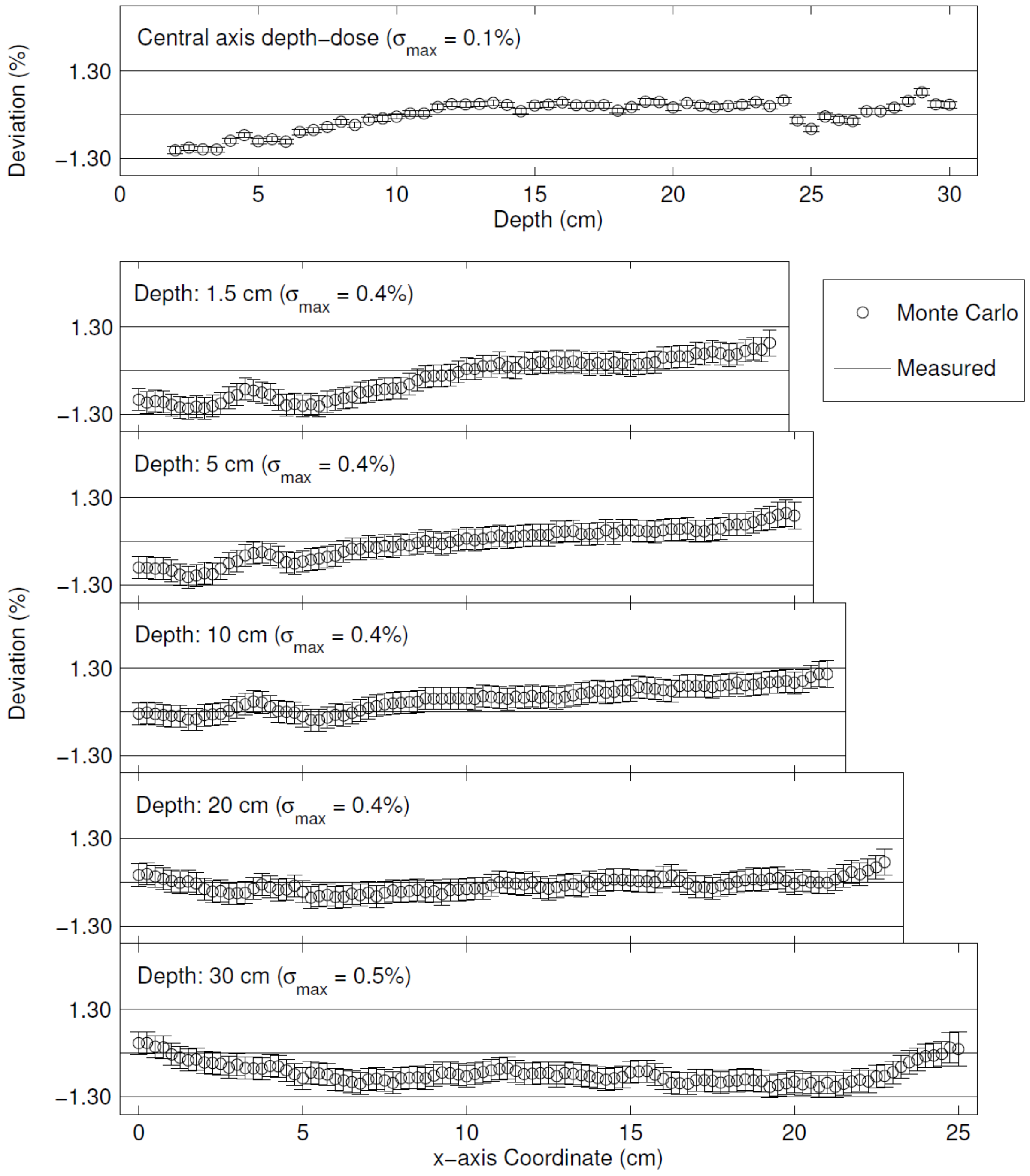


Figure 3.5 Percent deviation for curves for the most optimal parameter set, NRC/KM bremsstrahlung combinations, 40×40 cm^2 field size.

3.1.3 Model Accuracy and Sensitivity

Here, model accuracy has been defined as the percent deviation and distance-to-agreement thresholds that are met by all points within the three regions of interest (depth-dose fall-off, profile plateau, and profile penumbra). Because of the discrete nature of this thresholding however, this result is particularly sensitive to the statistical variation of the MC results. For this reason, some description of this variation must accompany the accuracy threshold. All simulations were performed until the maximum sigma of any point σ_{max} across all field sizes was less than 0.5%; however, due to nature of SG smoothing, the average sigma σ_{mean} was considerably less. This makes the accuracy threshold considerably more robust than one might expect from considering only the σ_{max} value. A much more descriptive measure of the statistical sensitivity of the result would include the standard deviation σ_{std} of the sigma distribution as well as its mean. These values for the optimal parameter set were σ_{mean} = 0.3% and σ_{std} = 0.08%.

Source parameter optimality was found to be moderately sensitive to variations in SG smoothing parameters. For example, decreasing the χ^2 threshold from 2 to 1.5 shifted the most optimal parameterization by 1.0°. The choice of smoothing parameters is a somewhat subjective process and several validation methods are discussed by Kawrakow[83]. For this study, because the general shape of depth-dose and profile distributions is well understood, smoothing parameters were chosen through visual inspection. In particular, the χ^2 threshold was increased from one until all obvious statistical variations were smoothed across an entire field size set. This increase was then limited by the introduction of unphysical lower frequency “bows” into the smoothed distributions. A 5 cm maximum smoothing window width was employed to also help limit the latter behavior.

As discussed, the type of normalization employed can have a significant effect on parameterization optimality. When rerunning the analysis that produced Figure 3.1 with a type B normalization, objective function values improve for almost every parameterization across all six bremsstrahlung combinations. This is fairly conclusive evidence that type A normalizations lead to a more sensitive measure of model accuracy.

3.2 MLC Validation

3.2.1 Planar Comparisons

Figures 3.6 and 3.7 illustrate MC and film results for two of the 8 measured/simulated planes. The MC uncertainty frames in both figures illustrate the $\sigma_{max} \leq 0.5\%$. Overall, MC dose alignment is in excellent agreement with the film measurements. This can be seen in the 1%/1 mm Gamma values being consistently low in leaf-to-aperture transition regions. In open regions, on the other hand, the general noisy behavior of the film is obvious. For film measurement taken in the same 60 minute period, with the same calibration, same setup, and same machine, significant deviations are apparent. Of the 8 films, the 5 cm depth exposure for Field 1 was relatively well behaved, while that for 10 cm depth for Field 4 illustrates a more average measurement response. It is worth noting that these measurements were completely repeated on 3 separate occasions, each with independent calibration curves. The noisy open field response exhibited here is highly representative of the general trend for this measurement medium, in this context.

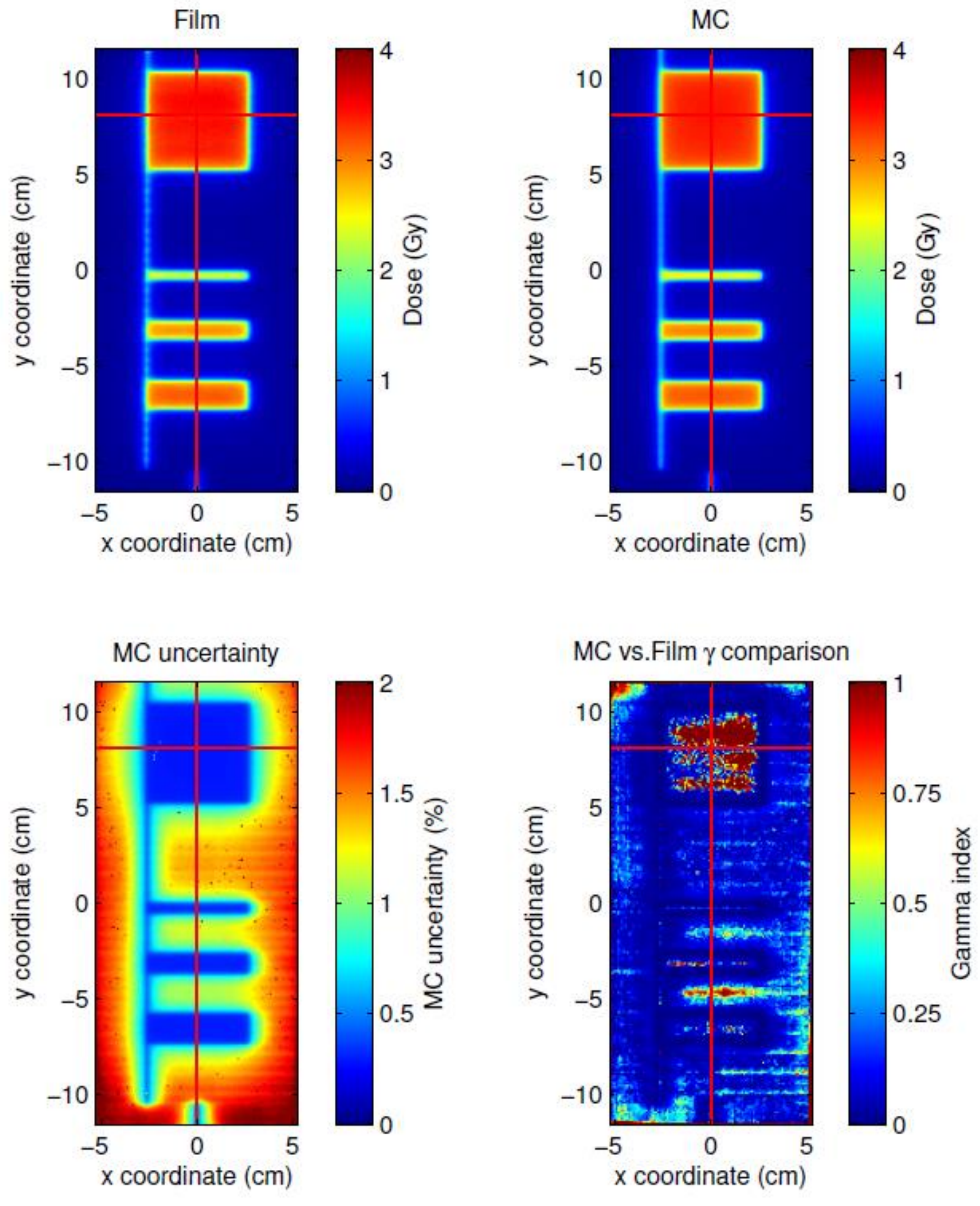


Figure 3.6 MC dose, MC uncertainty, film dose, and their gamma index (1%/1mm) comparison for field 1, at 5 cm depth. Red lines delineate where profiles were extracted for Figure 3.8.

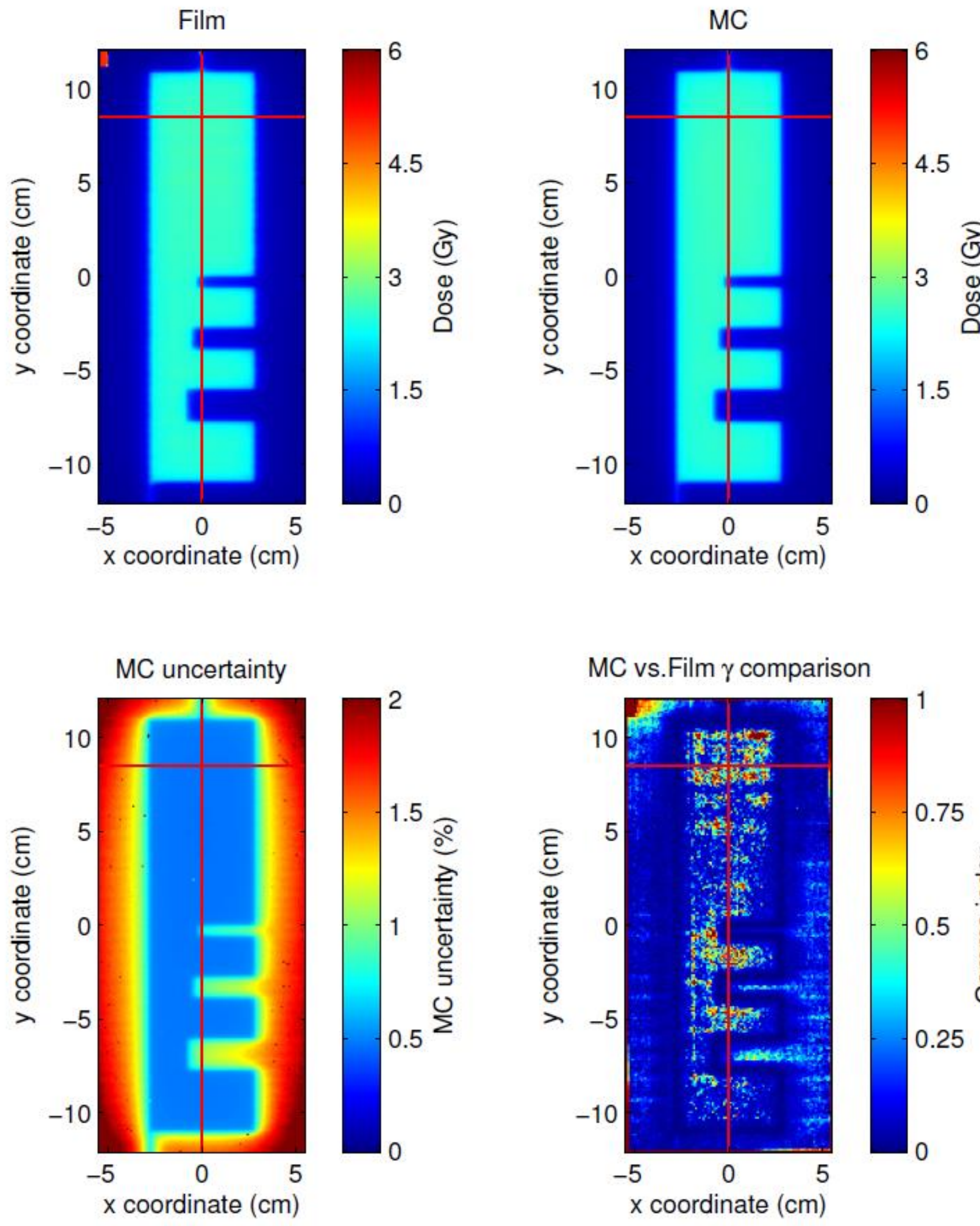


Figure 3.7 MC dose, MC uncertainty, film dose, and their gamma index (1%/1mm) comparison for field 4, at 10 cm depth. Red lines delineate where profiles were extracted for Figure 3.9.

3.2.2 Profile Comparisons

Figures 3.8 and 3.9 illustrate *x* and *y* profiles extracted from the same planar dose distributions illustrated in Figures 3.6 and 3.7. Profiles were selected based on how clearly they are able to identify the mechanical functionality of the virtual MLC. Here, the same high level of leaf-to-aperture alignment is illustrated as was for the planar comparisons. Also displayed in these two figures is the AAA calculated profiles. These consistently agree well with MC which supports the assumption that film measurements are not likely displaying the true open field dose. Some low frequency deviation between MC and AAA is apparent however this is not surprising due the limitations of the AAA model of beam modulation, i.e. beamlets in AAA are either on or off, scatter from collimator jaw ends or through the complex MLC geometry is not modeled. The overall trend for MC/film/AAA profile comparisons is similar to that for the planar distributions: Field alignment is excellent, however the dosimetric accuracy of the film is questionable.

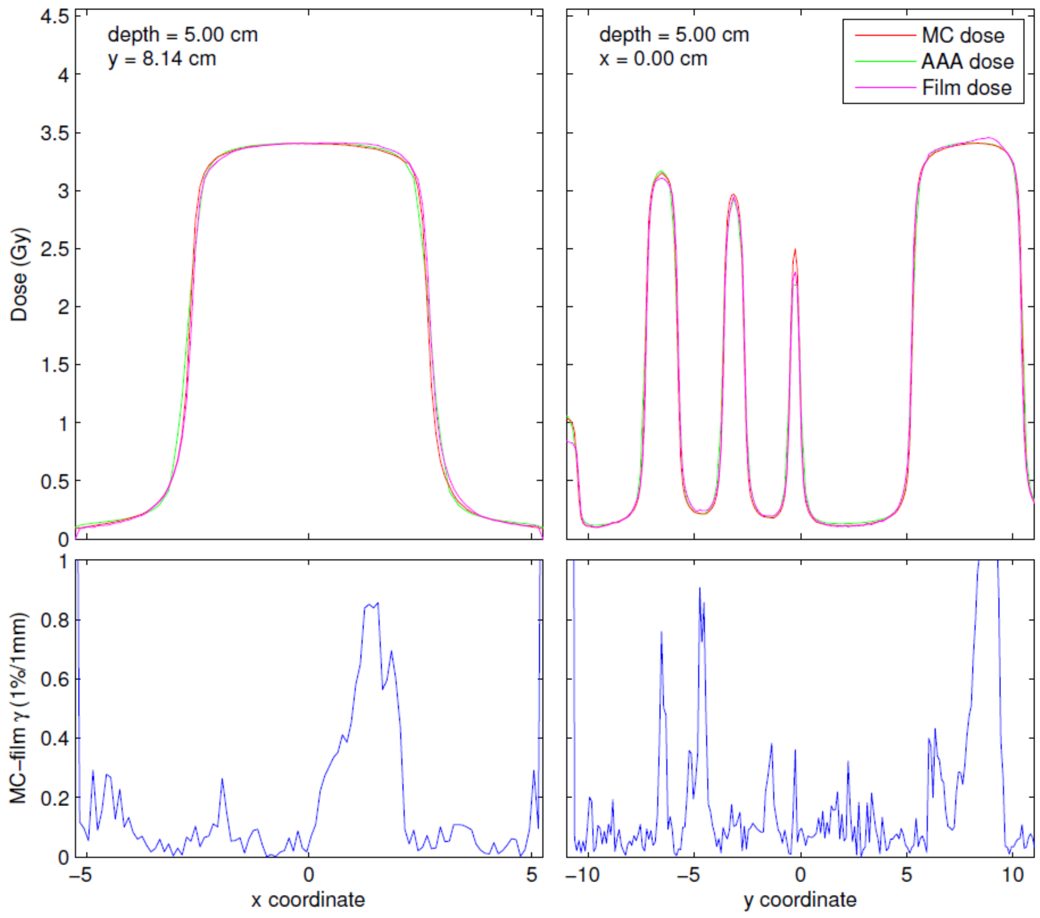


Figure 3.8 MC, film and AAA profiles for field 1 at 5 cm depth along the red lines delineated in Figure 3.6.

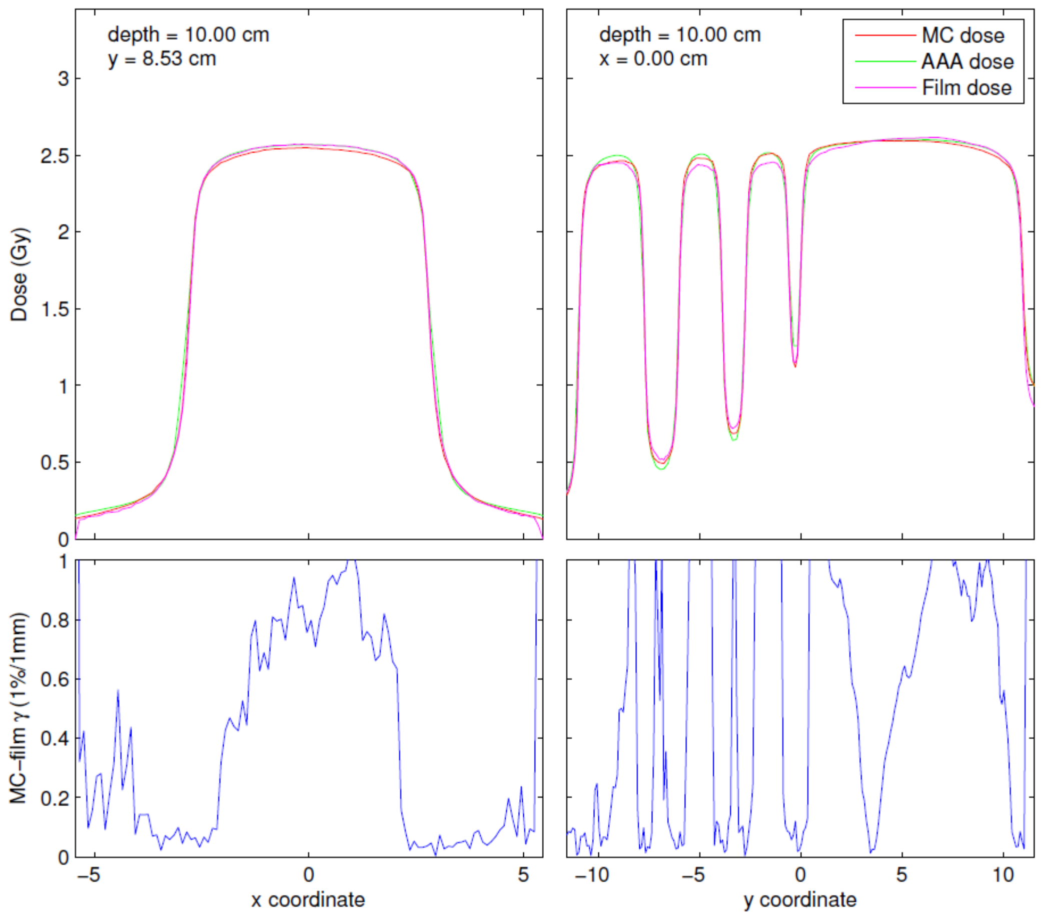


Figure 3.9 MC, film and AAA profiles for field 4 at 10 cm depth along the red lines delineated in Figure 3.7.

3.2.3 MC Model Compered to the Physical Machine Output

Because the measured output of a physical machine is being used as a comparison benchmark for the output of the EBD based MC model, it is important to understand how well their open field data match. Figure 3.10 illustrates the fractional depth-dose and profile doses for in-house measured and the EBD data for the 20×20 cm^2 field size. The corresponding locally weighted fractional deviation is illustrated in Figure 3.11. For these three depths all in-house measured points, save a few near the penumbra region in the 10 cm depth profile, fall within 1% of the EBD. This excellent agreement is also seen for the 3×3 cm^2 and 10×10 cm^2 field size as well (Figures are located in Appendix E). This high level of matching insures that the output of the physical machine is suitable for benchmarking the MC code even though it was parameterized with a different data set.

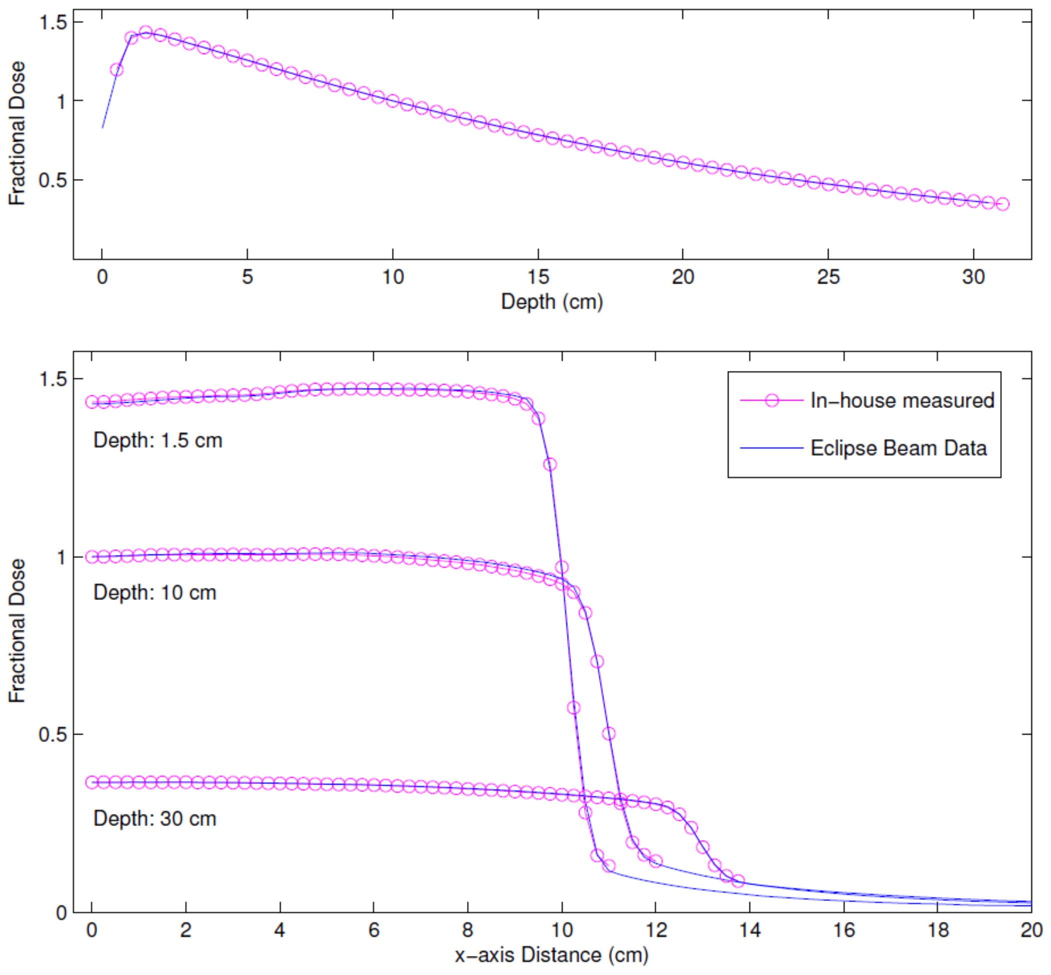


Figure 3.10 Fractional depth-dose and profile doses for in-house measured and EBD data for the 20×20 cm^2 field size.

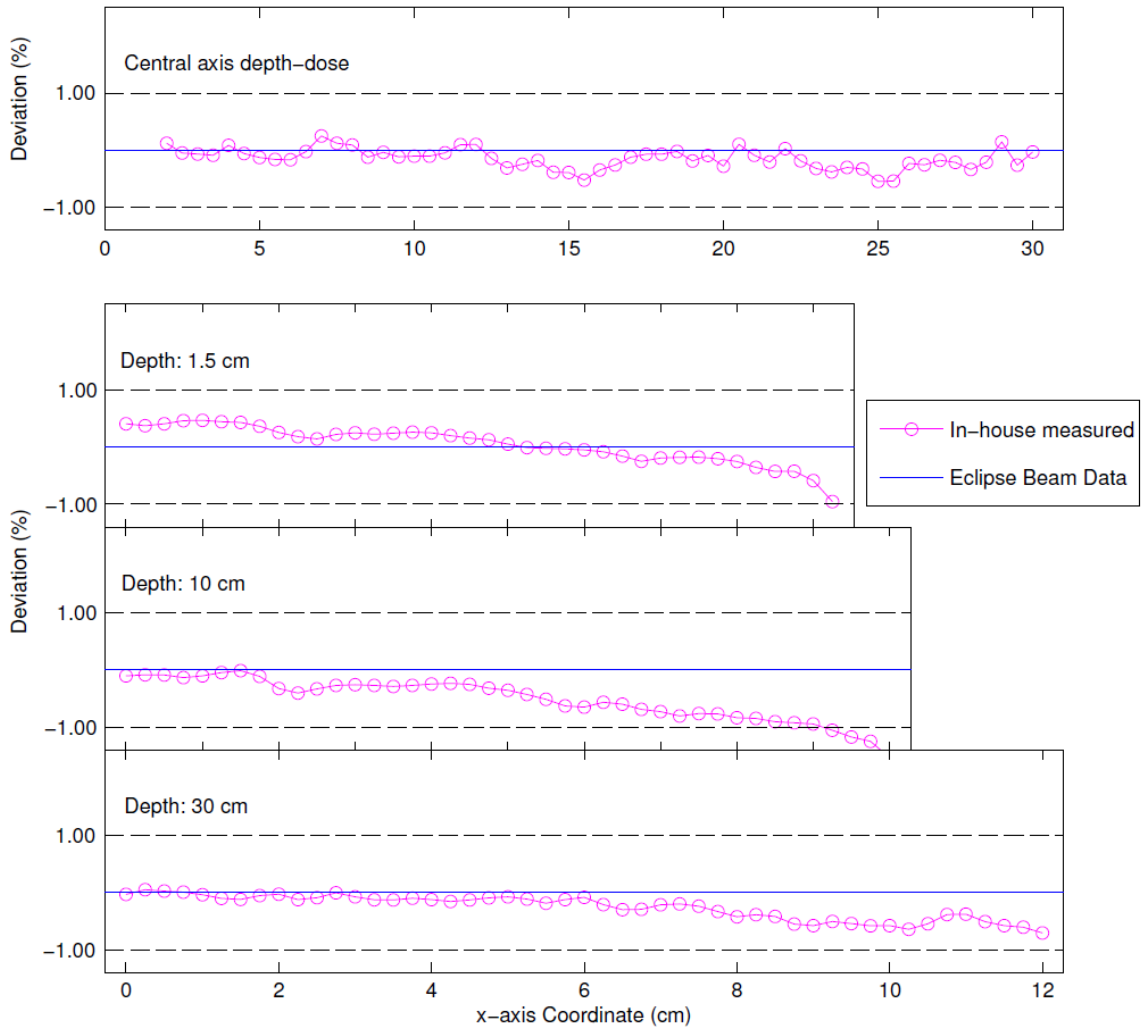


Figure 3.11 Locally weighted fractional deviation of in-house measured and EBD data for the 20×20 cm^2 field size.

It is worth noting that deviations in dose magnitude between film measured dose and either MC or AAA are not likely due to the differences in the open field data set used to configure either algorithm. For the AAA case, MLC modulation is modeled as a simple binary process as discussed in section 1.6.2 so even if the unmodulated calculated field perfectly matched that of an accurate measurement of the physical radiation field, MLC scatter effects will still not be accounted for. MC, on the other hand, will accurately model transport through the MLC, but as is clear from inspection of Figures 3.6 through 3.9 significant deviations between calculated and film measured dose-distribution still exist. This is all however, not important for the validating the mechanical functionality of the virtual MLC as it would take an extreme failure of the film measurement to induce a significant lateral shift in the nearest 50% penumbra point.

3.3 MC Treatment Plan Calculation

3.3.1 Absolute Dose Calibration

As currently implemented the absolute dose calibration as described in section 2.3.6 leads to improper dose scaling for jaw setting significantly different than the 10×10 calibration setting. This is likely due to improper modeling of how virtual monitor chamber is collecting backscatter dose. An alternative method of calibrating the MC dose is to take a simple volume dose to *dpp* ratio between the two dose-distributions

$$D_{MC,dpu,ijk} = D_{MC,dpp,ijk} \frac{\sum_{i'j'k'}^{ROI} D_{AAA,dpu,i'j'k'}}{\sum_{i'j'k'}^{ROI} D_{MC,dpp,i'j'k'}} \tag{3-1}$$

where the region of interest (ROI) is set to some high dose volume. For the calculations described below, this alternative method was employed with the ROI being defined as the set of voxels with dose greater than 80% of the maximum dose. The logic here may seem circular in that here the MCTPCF is being benchmarked with the algorithm for which it is being developed to investigate, however the current validation procedure is only designed to confirm the mechanical functionality of the MCTPCF. Its dosimetric accuracy is considered a different issue and again, the dosimetric accuracy of the EGSnrc transport engine has been benchmarked extensively elsewhere.

3.3.2 Plan Comparisons

As discussed, four test plans were calculated with the MCTPCF. Planar dose comparisons for all four cases are displayed in Figures 3.12 through 3.15. For these figures, dose less than 5% is set to zero and red lines indicate isocenter axes and green lines indicate the high-dose (>80% of max dose) center-of-mass axes. Planes are all extracted at this center-of-mass point. It is clear from inspection alone that the MCTPCF is functioning from a mechanical standpoint as there are no obvious indications of misalignment or incorrect dose shaping. Some small level of deviation is evident for the lung and head-and-neck cases, but for the more homogeneous geometries the dose distributions are nearly identical. This is a strong validation of the calculation framework as unless there were some mechanical malfunction with the code, in the absence of modulation or heterogeneities AAA is known to calculate with the same accuracy as MC.

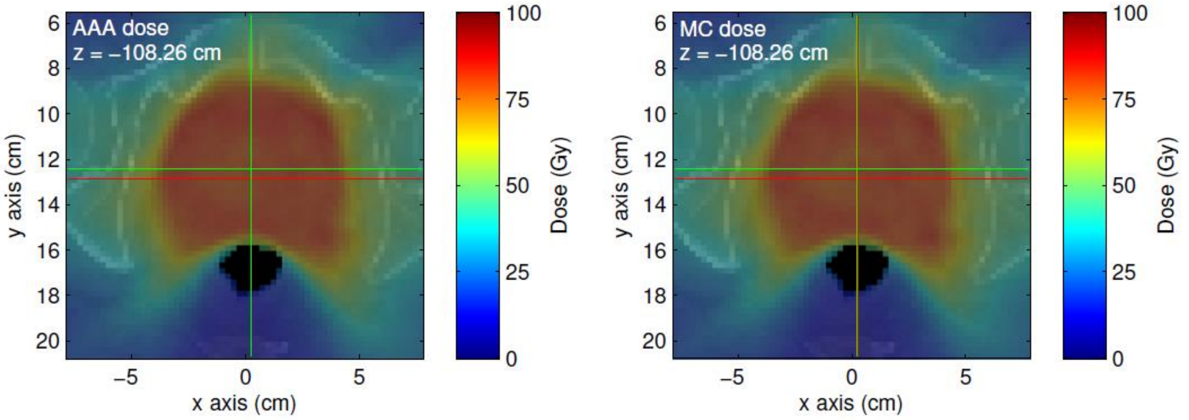


Figure 3.12 Transverse view of AAA and MC dose-to-water in the high-dose plane for the prostate case. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

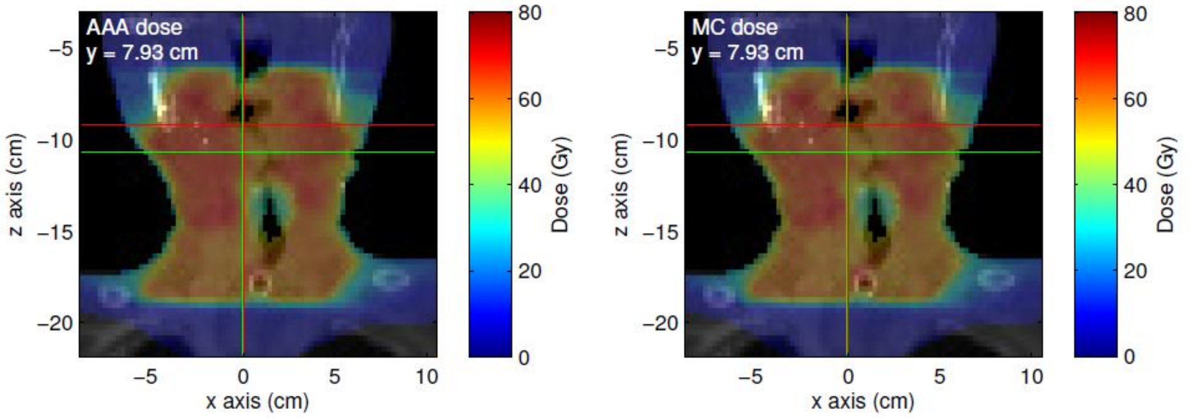


Figure 3.13 Coronal view of AAA and MC dose-to-water in the high-dose plane for the head-and-neck case. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

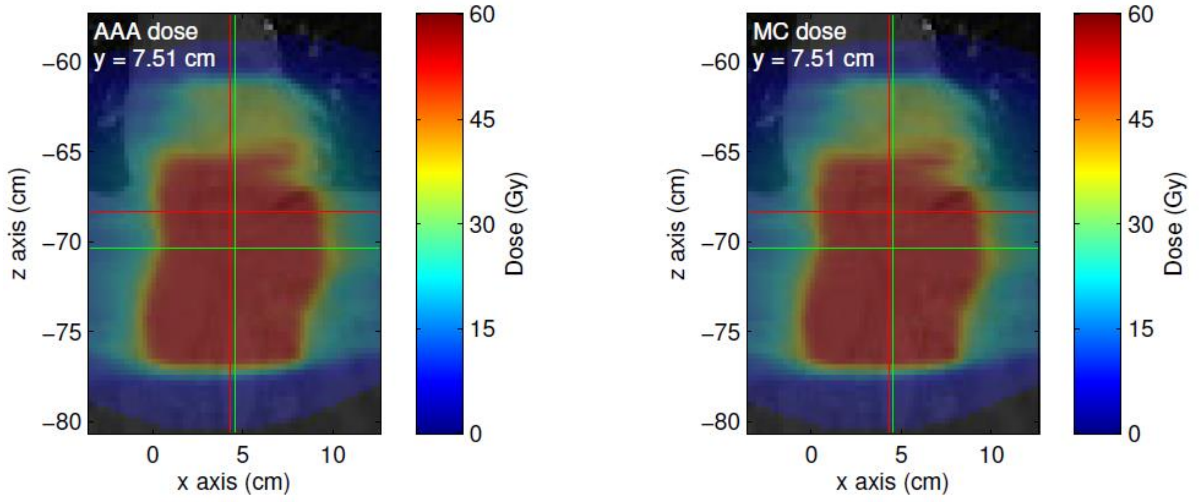


Figure 3.14 Coronal view of AAA and MC dose-to-water in the high-dose plane for the esophageal case. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

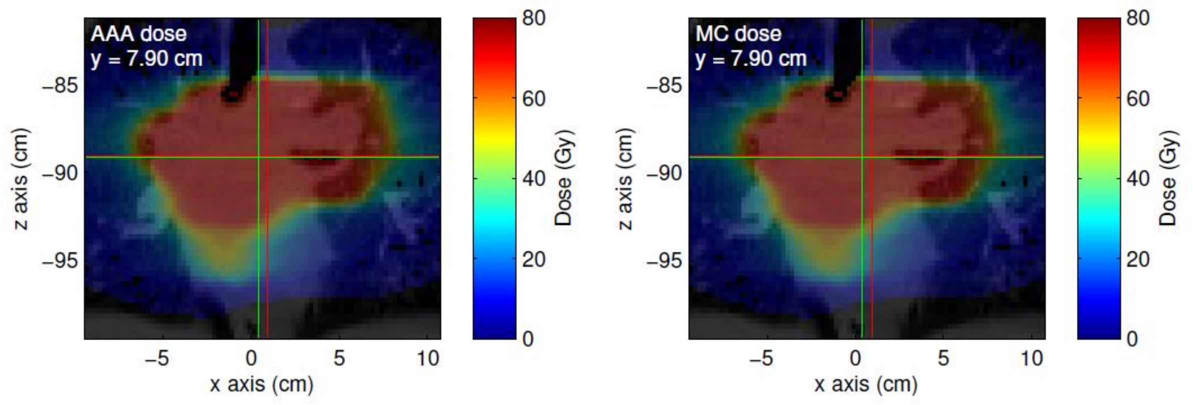


Figure 3.15 Coronal view of AAA and MC dose-to-water in the high-dose plane for the lung case. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

The deviations were quantified with a two-dimensional gamma calculation. Gamma distributions are displayed in Figures 3.16 through 3.19. For the selected high-dose planes, all plans pass with at least 97% of voxels meeting the 1%/1mm gamma criteria. As expected, the two most homogeneous geometries (prostate and esophageal) have nearly perfect passing rates. In fact reducing the threshold for the prostate case to 0.5%/0.5mm, still yields a 99% passing rate. The more heterogeneous cases (head-and-neck) have slightly lower passing rates of at least 97%. Note that the passing percentages here are based on only the voxels shown in the figures with dose values greater that 10% of the maximum dose.

It is expected that the MC calculated head-and-neck and lung dose distributions will deviate more from AAA than the more homogeneous cases due to the highly modulated treatment combined with the highly heterogeneous geometries of the oropharynx and lung regions. The level of modulation for these cases is assessed in a later section with the field-size index. The deviations due to high and low density regions are clearly visible when scrolling through these two dose-distributions. Specifically, MC calculates a higher dose near and inside bone, and also in lower density regions that closely (<1cm) border tissue density regions. Due to complexity of the treatment, attributing this behavior to a specific dosimetric effect is difficult; however the application of the FSI, HSI, and LDI does provide some useful information as to the cause of these deviaitons.

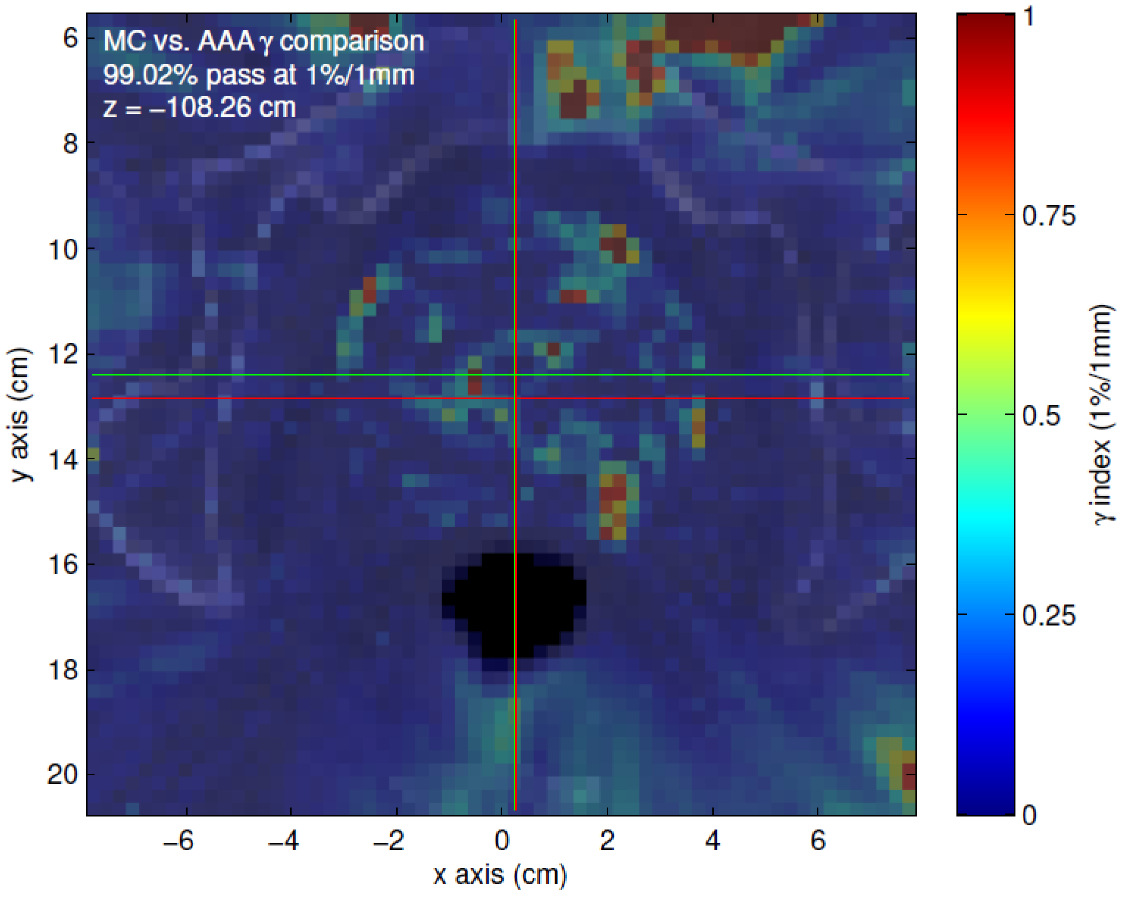


Figure 3.16 Transverse view of AAA to MC (dose-to-water) gamma comparison in the high-dose plane for the prostate case. Gamma threshold was set to 1%/1mm. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

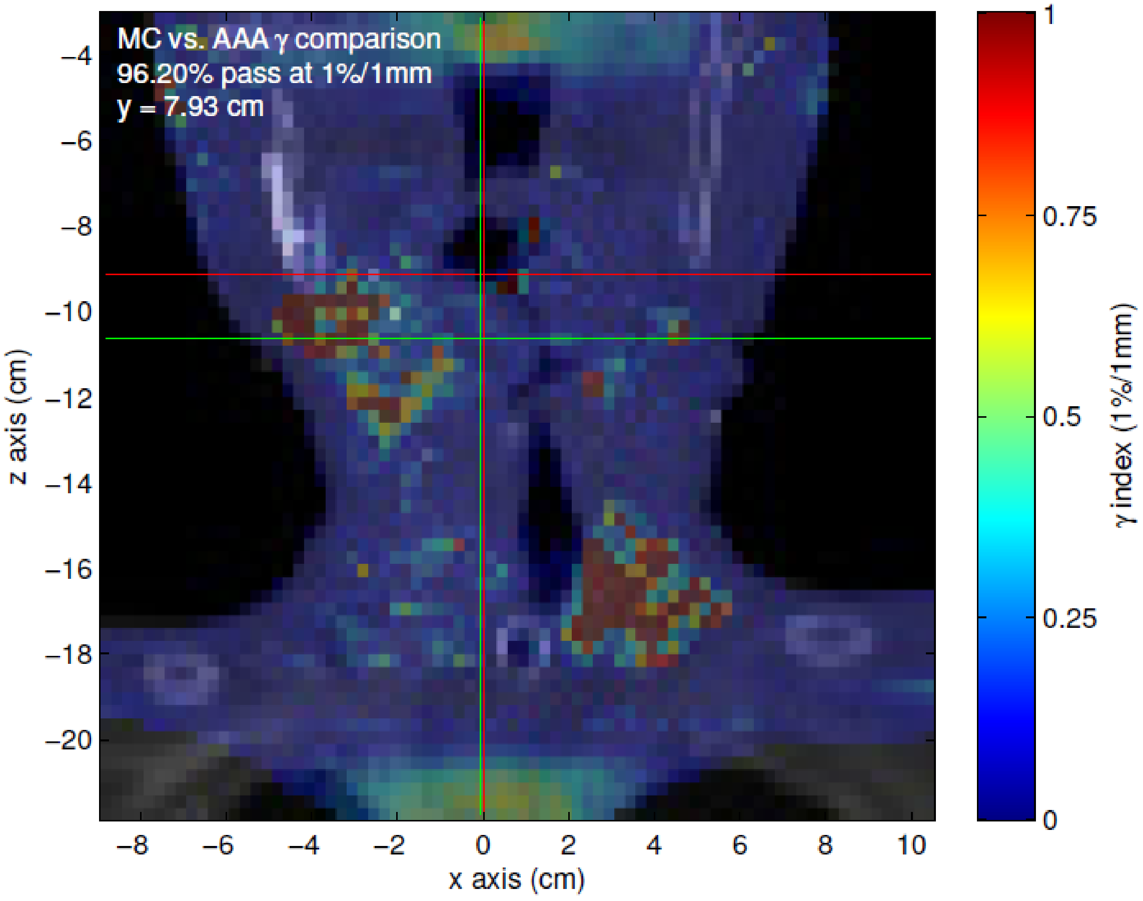


Figure 3.17 Coronal view of AAA to MC (dose-to-water) gamma comparison in the high-dose plane for the head-and-neck case. Gamma threshold was set to 2%/2mm. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

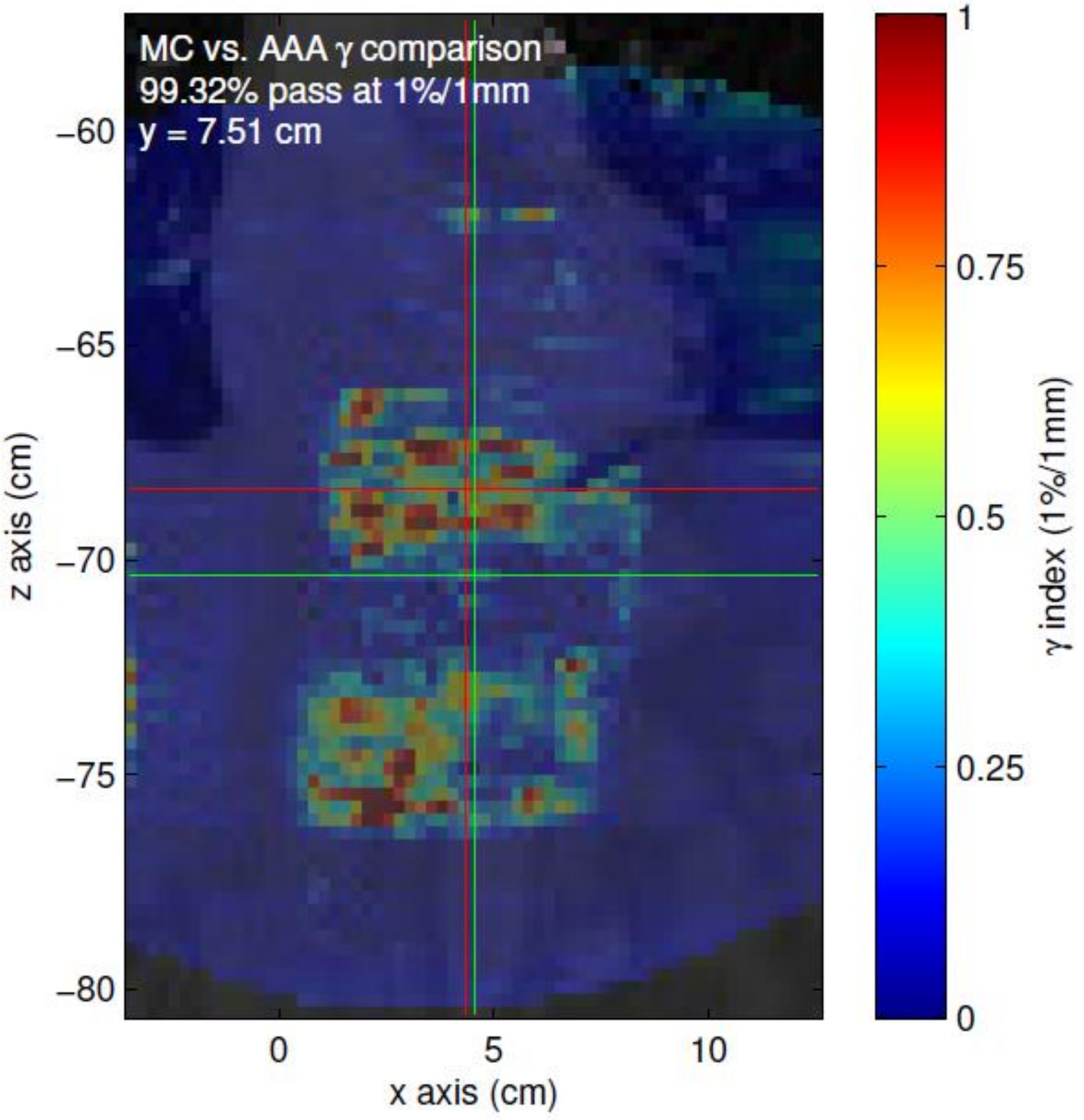


Figure 3.18 Coronal view of AAA to MC (dose-to-water) gamma comparison in the high-dose plane for the esophageal case. Gamma threshold was set to 1%/1mm. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

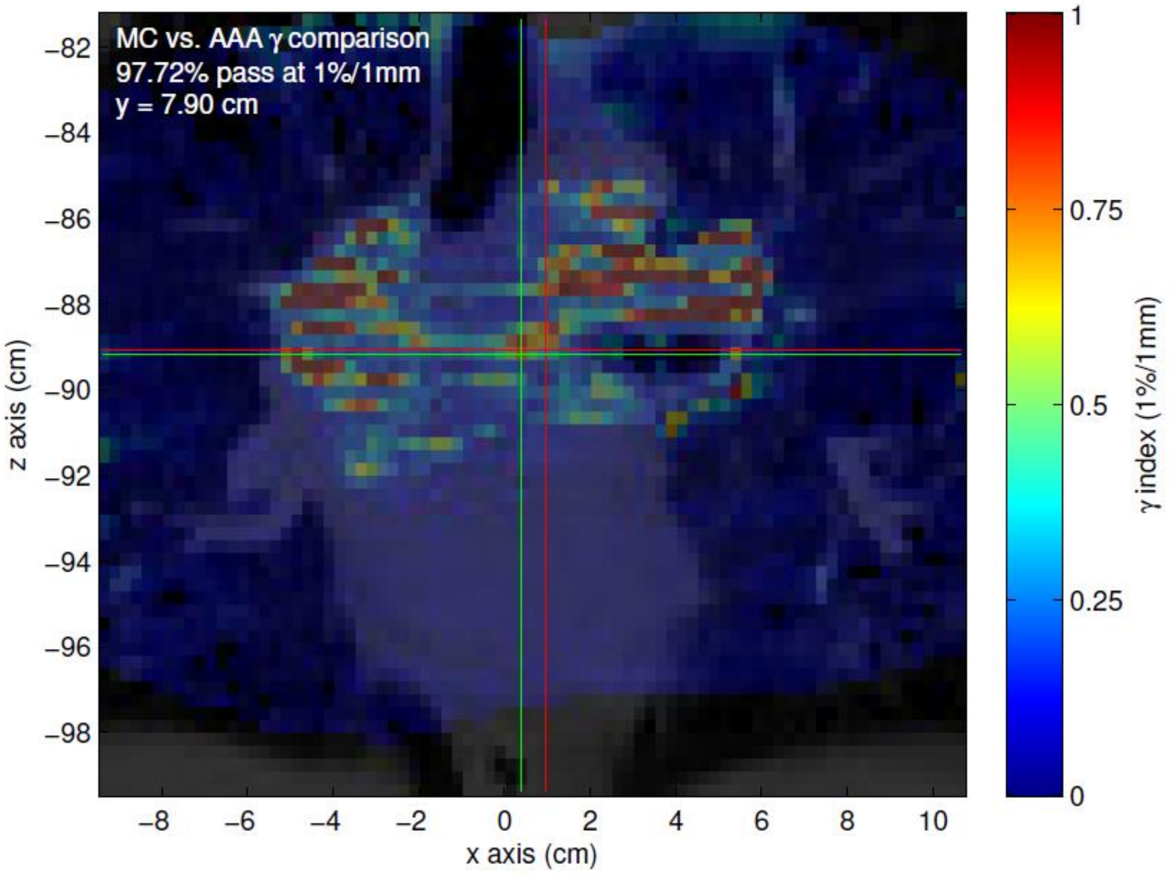


Figure 3.19 Coronal view of AAA to MC (dose-to-water) gamma comparison in the high-dose plane for the lung case. Gamma threshold was set to 1%/1mm. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

3.3.3 AAA Output Factor Calibration

Another factor affecting the deviation analysis is the way the MC dose distributions are normalized. If the normalization region discussed in the beginning of this section encompasses a region of true deviation between MC and AAA, the dose scaling may be incorrect. For this reason, at least from a theoretical standpoint, an absolute dose calibration using the formalism of equation (2-4) is preferable to the one used here. To investigate the possible calibration errors induced by the simpler method, open field AAA calculations with jaw positions set to that of a test plan were executed to find correct dose-per-MU values at the linac calibration point. The ratio of this value and a MC calculated value were then used to obtain a correct N_{MU}. This process effectively applies the same output factor used by the TPS to the MCTPCF calculation. Figures 3.20 and 3.21 illustrate coronal isocenter profiles for each type of absolute dose calibration for the test head-and-neck case. From inspection, a significant difference in absolute dose scaling is clear between the two methods. It is expected that using the AAA output factor will yield a more accurate calibration and indeed the head-and-neck gamma comparison result for the dose plane in Figure 3.17 is significantly improved with this method. This comparison yielded a 99% passing rate at 1%/1mm and is illustrated Figure 3.22.

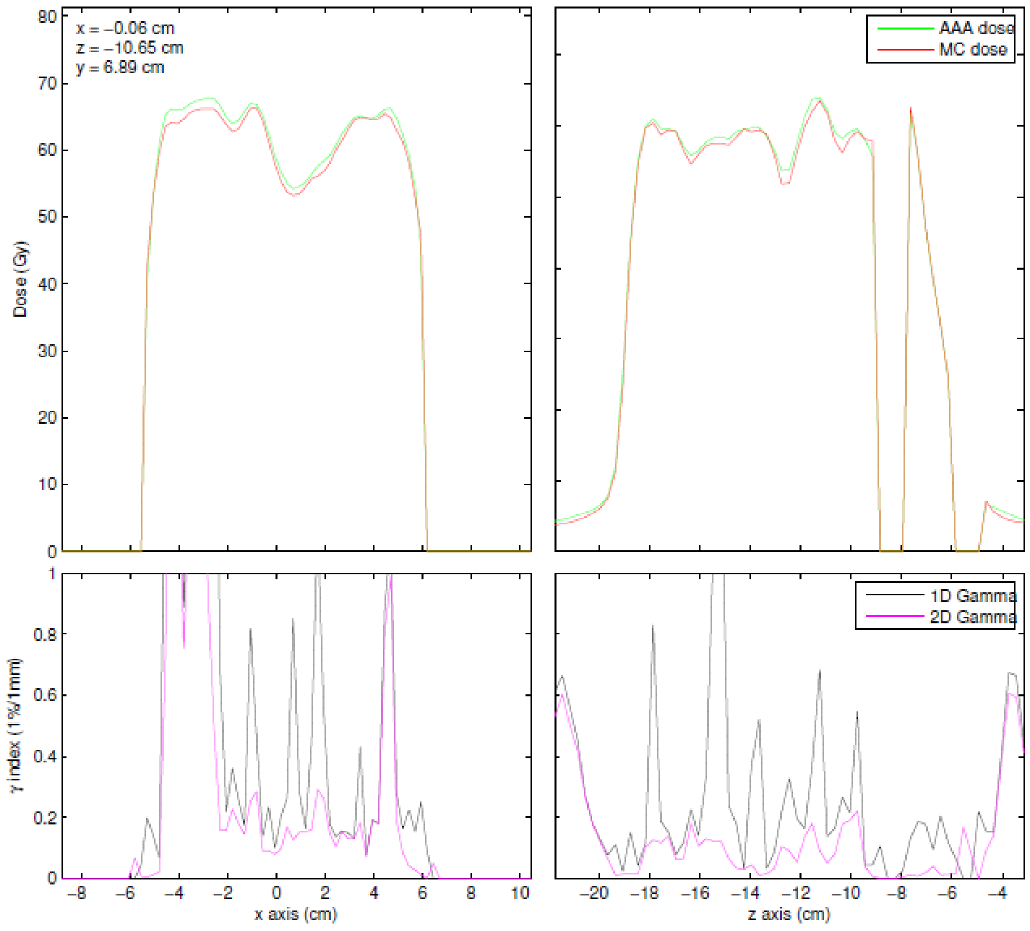


Figure 3.20 *x* and *z* dose and gamma profile comparisons of AAA to MC (dose-to-water) at isocenter for the head-and-neck case using the calibration method described in Eq. (3-1). Gamma threshold was set to 1%/1mm. Both the 1D and 2D gamma calculations are displayed.

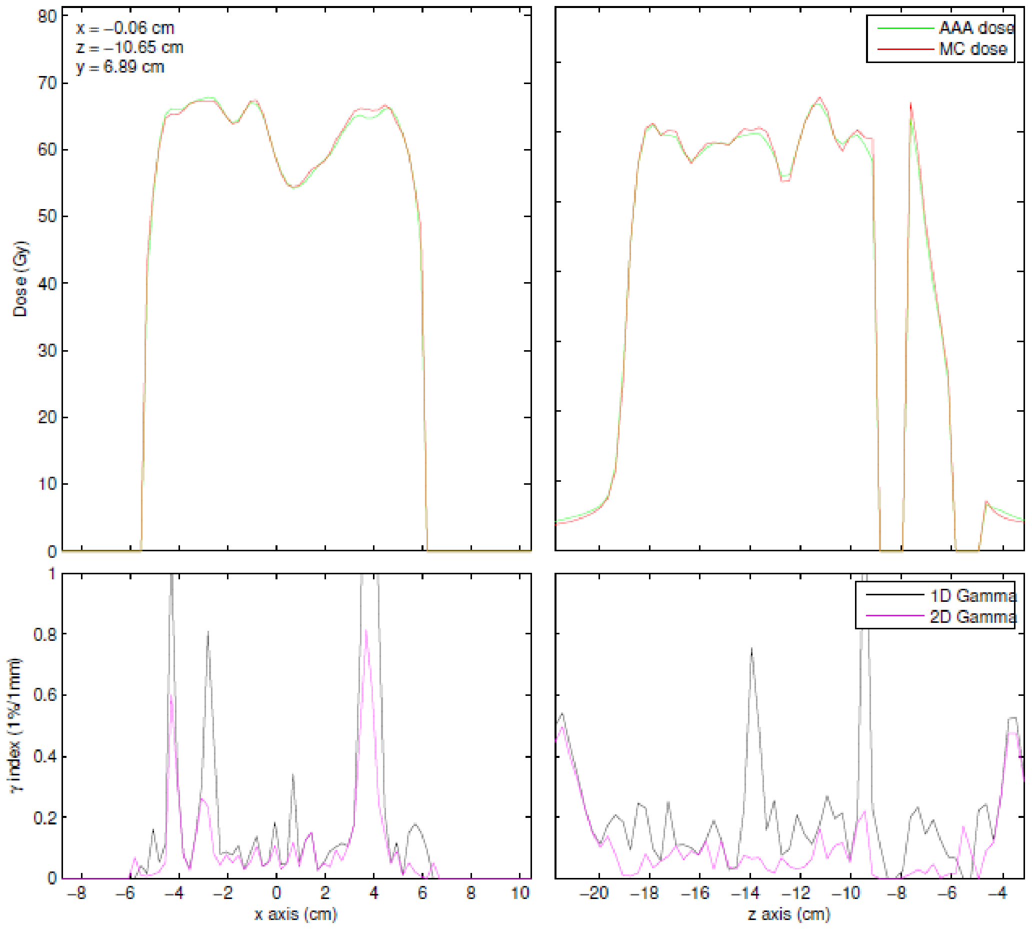


Figure 3.21 *x* and *z* dose and gamma profile comparisons of AAA to MC (dose-to-water) at isocenter for the head-and-neck case using the AAA calculated output factor calibration. Gamma threshold was set to 1%/1mm. Both the 1D and 2D gamma calculations are displayed.

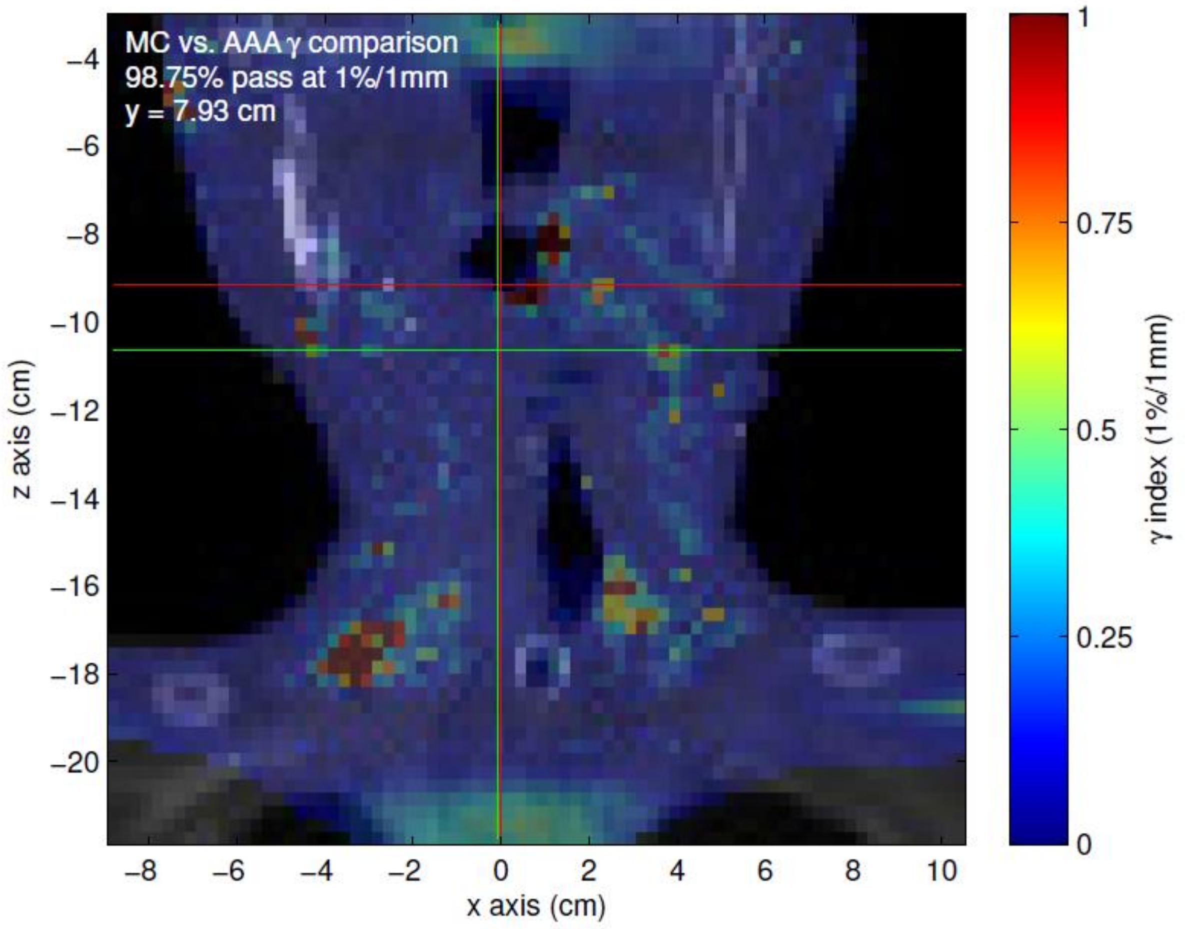


Figure 3.22 Coronal view of AAA to MC (dose-to-water) gamma comparison in the isocenter plane for the head-and-neck case using the AAA calculated output factor calibration. Gamma threshold was set to 1%/1mm. Red lines indicate isocenter axes.

Also displayed in Figures 3.20 and 3.21 is the one-dimensional gamma. This illustrates how adding a dimension to a gamma calculation can provide an a) clearly different, and b) more correct analysis of the deviation between two complex dose distributions. The gamma calculation employed here (see gamma_calc.m) currently take about 30-60 seconds to analyze a 5000 voxel dose plane, depending on the requested calculation resolution and processor speed. A three dimensional calculation would obviously be more robust, however the required computation time is considerable. The development of a computationally feasible 3D gamma calculation is currently underway for future work with the MCTPCF.

3.4 Error Prediction

3.4.1 Field Size Index

To evaluate the ability of the FSI to identify small field/block errors, it was applied to the four MLC fields introduced in section 2.2.3. Figures 3.23 and 3.24 illustrate the strong agreement between the fractional deviation (FD) of AAA/MC and the FSI map for MLC field 1. This strong agreement can be observed for all 4 fields. The profiles illustrated in Figure 3.24 showed best agreement with f_{FSI} = 0.11. This value was derived from visual inspection of profile figures and was a tradeoff between what was required by field 1 and by field 3. The difference in modulation between these two fields is the that field 3 has 2D aperture gradients and field 1 has 1D; this suggests that there is a more robust way to combine the component gradients, as opposed the norm in equation (2-5), to achieve better FSI/FD matching.

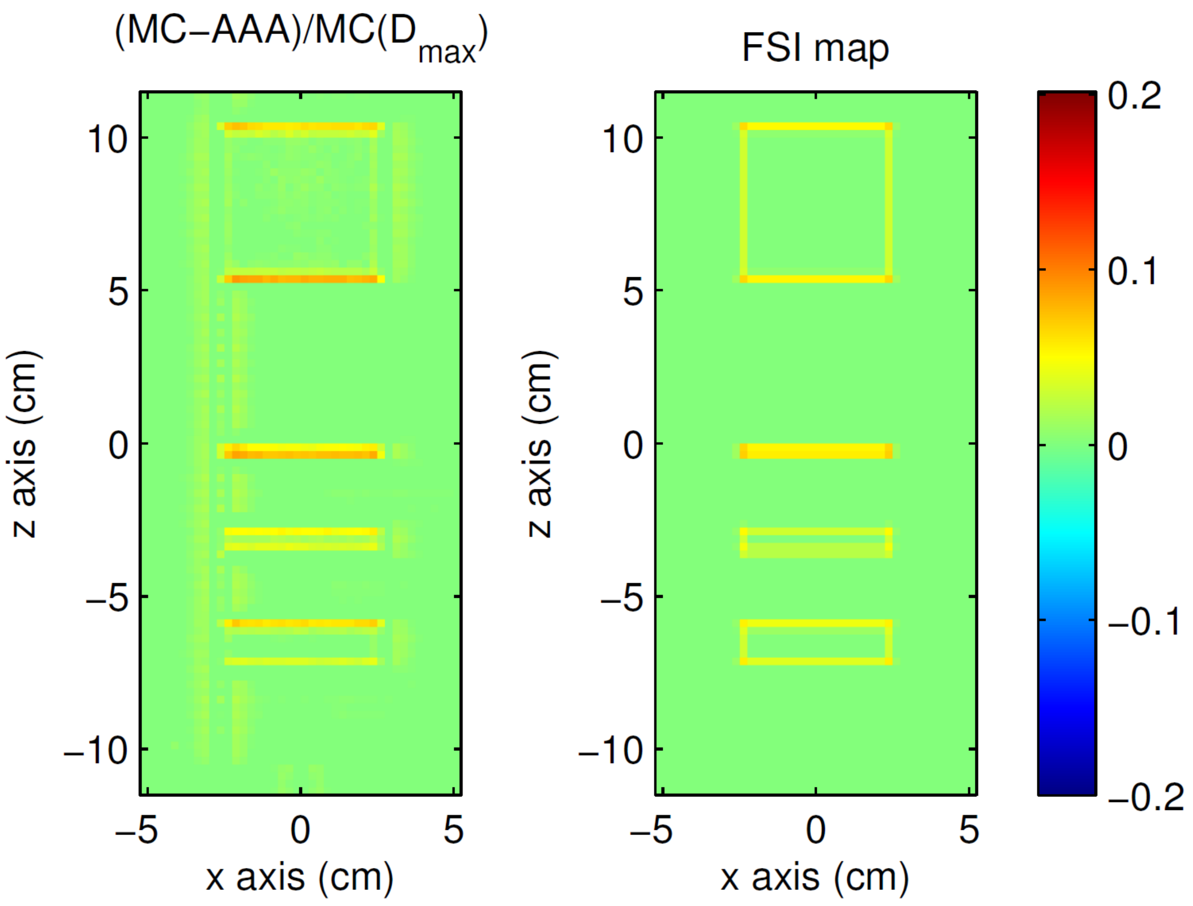


Figure 3.23 MC/AAA percent deviation and FSI planes for field 1 at 5 cm depth.

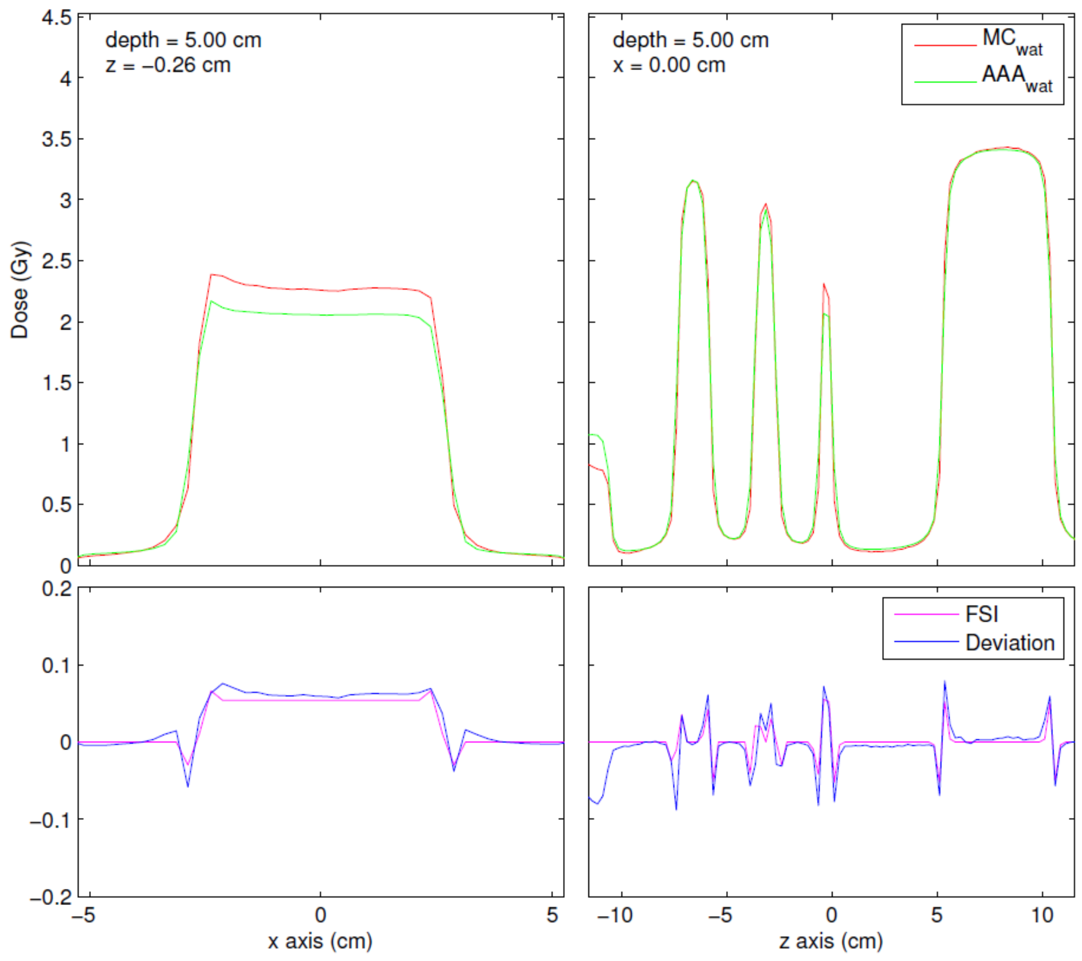


Figure 3.24 MC and AAA dose profiles for field 1 at 5 cm depth are displayed in the top frames. Percent deviation and FSI profiles are displayed in the lower frames.

The FSI was calculated for all four test cases and compared with FD between AAA and MC. Figure 3.25 illustrates the high-dose transverse plane for the prostate case. Here strong shape and magnitude correlations between the FSI map and the FD map are indicated with yellow. Correlations of this magnitude are common throughout most planes of all four test case comparisons.

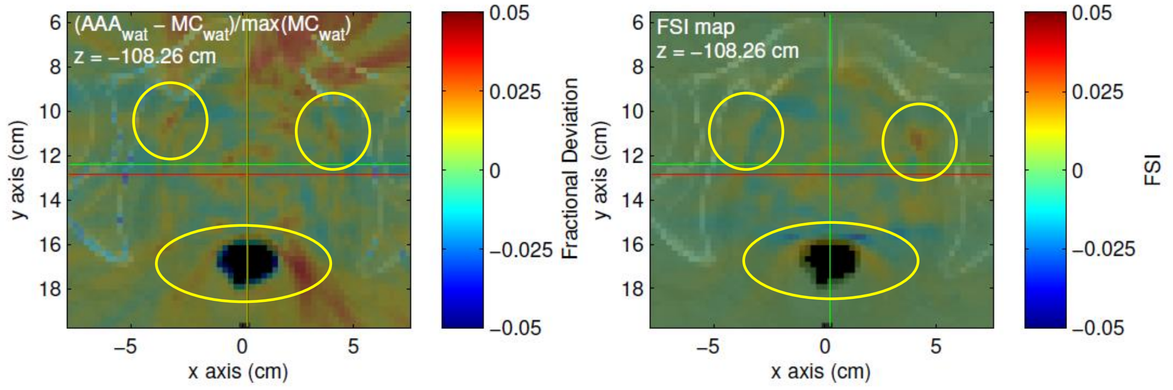


Figure 3.25 MC/AAA fractional deviation and FSI maps for the high-dose transverse plane for the prostate case. Regions marked in yellow show a strong direction, shape, and magnitude correlation. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

Figure 3.26 illustrates the high-dose coronal plane for the head-and-neck case. Here, a matching cross-hatching pattern is apparent in the both the deviation and error maps. The angle of this pattern matches that of the collimator angle which is a strong indicator that the pattern is directly attributable to MLC modulation. Similar cross-hatch pattern correlations can be found in almost all, beam-perpendicular, planes for the other cases as well.

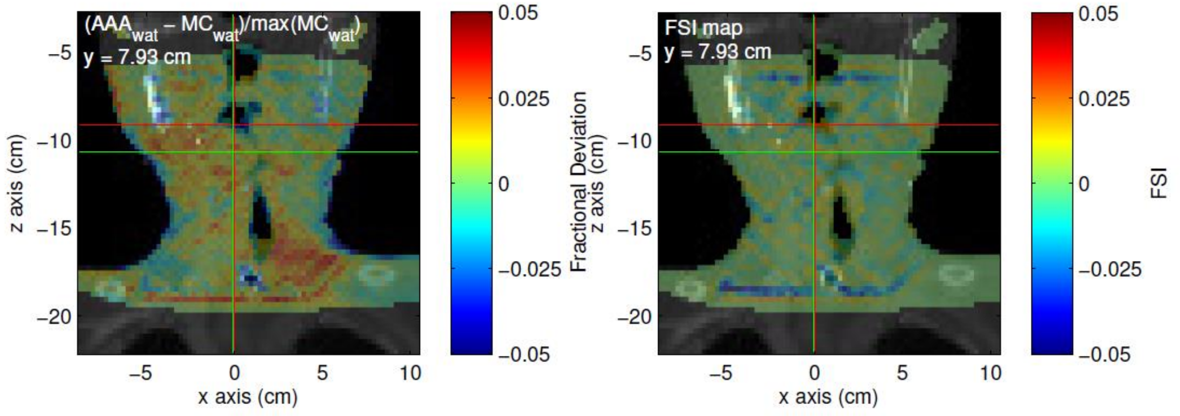


Figure 3.26 MC/AAA fractional deviation and FSI maps for the high-dose coronal plane for the head-and-neck case. Regions marked in yellow show a strong direction, shape, and magnitude correlation. Red lines indicate isocenter axes and green lines indicate the center-of-mass of voxels with dose >80% of maximum dose.

Regardless of the whether correlations exist or not, the FSI does predict small field/block overlap for highly modulated, multiple control point treatments and can be used as a complexity index based only on this fact. As would be expected from a properly functioning FSI, target volume edges and regions near modulation inducing organs (e.g. spinal cord) are strongly identified throughout the FSI maps.

3.4.2 Heterogeneous Scatter Index

As this index is still under development, there are not results to display for the theory introduced in the previous section. It is worth noting that the first iteration of the HSI involved detecting regions of lateral heterogeneities by taking a 3D numerical gradient of the ED distribution and calculating its projection onto planar “scatter” kernels lying perpendicular to the beam direction. While this method was robust in detecting lateral heterogeneities, it did not address the true physical principles being violated by the AAA rectilinear kernel scaling approximation. This method was able to detect a low level of deviation between MC and AAA, however it also identified large regions where no significant deviation existed. Work in that direction was therefore discontinued in lieu of the theory presented above.

Already an issue for this point-wise method is computation time. In its current form, applied to a $0.25\times0.25\times0.25$ cm^3 calculation grid, in a MATLAB script, the method is completely intractable on a single processor. It is likely however that lowering the resolution and/or reducing scatter box extent will yield a time feasible HSI calculation.

3.4.3 Low-density Index

The upper frame of Figure 3.27 displays the results of the low-density comparison study described in section 2.4.3. Here both MC and AAA values are fit to individual linear plus exponential models. The ratio of these curves constitutes the LDI and is plotted in the upper frame. For open fields, this relationship is well-behaved and easily fit. AAA was found to slightly under-dose in regions where the electron density is greater than approximately 0.1 g/cm^3. At lower densities, AAA begins to severely overdose as predicted by the literature[3,18].

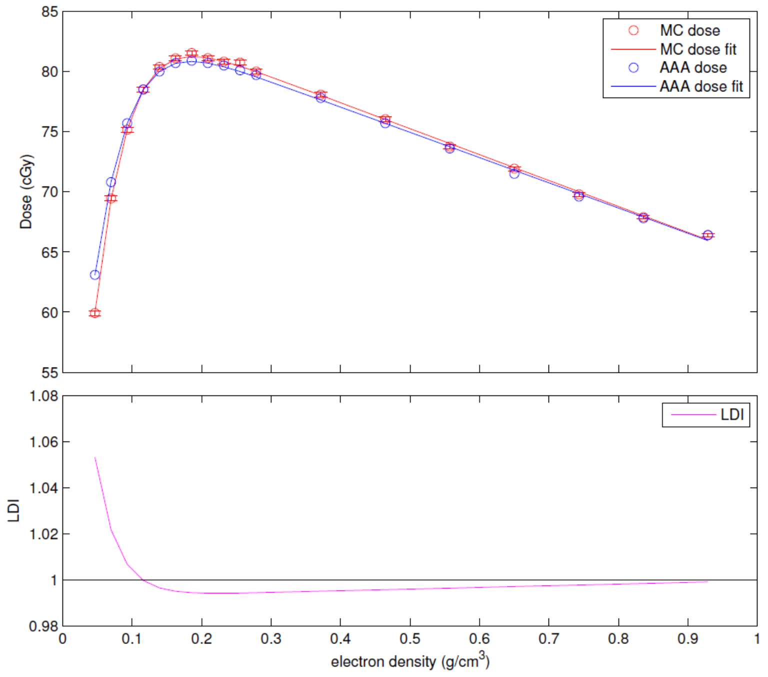


Figure 3.27 In the upper frame, the MC and AAA dose at 10 cm depth, 100 cm SSD, for a 10×10^2 field is plotted over a range of electron densities. The solid lines indicate the linear plus exponential fits of the data. The ratio of these fits (equation (2-11)) is the LDI and is plotted in the lower frame.

Figure 3.28 illustrates a selected transversal slice of the LDI applied to the clinical lung case. In the right frame, isolated regions of the lung volume are indicated to be either under or overdosed due to the low density effect. These predicted effects however, are vastly washed out by other deviations as indicated in the lower left frame. It is worth noting that only for a small fraction of the lung is the electron density low enough to suffer from the most severe effects of AAA overdosing. This behavior, combined with the resultant effects being relatively small in comparison to other deviations, suggests that the LDI is of limited utility, at least in its current form.

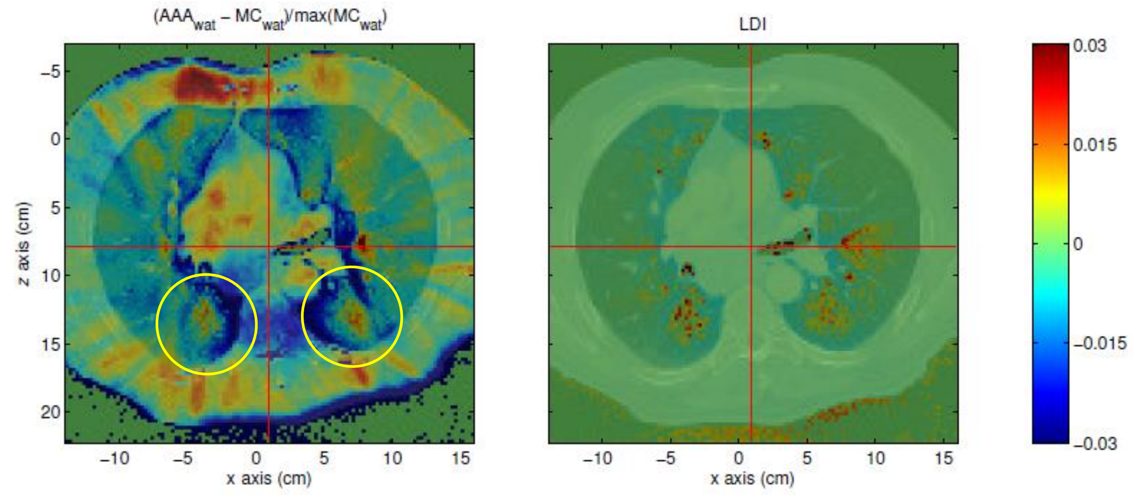


Figure 3.28 Percent deviation of AAA from MC is displayed in the left frame and the corresponding LDI map is displayed in the right frames. Regions marked in yellow indicate overdosing by AAA which is predicted to occur in all regions of the lung by the LDI.

Chapter 4 Conclusion

4.1 Section Conclusions

4.1.1 Linac Modeling

In addition to electron source energy and intensity distribution, optimization of beam divergence and bremsstrahlung transport settings was necessary in order to achieve the stated model accuracy. Under the specified parameter resolution (0.01 MeV, 0.05 cm, and 0.1°, agreement of all points within 1.3% and 1.0 mm DTA (σ_{mean} = 0.3% , σ_{std} = 0.08%, σ_{max} = 0.5%) was achieved when using the NRC total bremsstrahlung cross section, the higher termed Koch and Motz bremsstrahlung angular sampling, and with electron source parameters of 5.90 MeV, 0.15 cm, and 1.0°. Source parameter optimality as well as model accuracy were both found to be moderately (±0.01 MeV, ±0.05 cm, and 0.1°) sensitive to smoothing parameters, as well as normalization type. Type A normalizations remove the approximation that there is no deviation between the benchmark and the comparison lateral profiles at their central axis value. This produces a more correct model and in general reduces the level of agreement that might otherwise be achieved with a type B normalization.

4.1.2 MLC Validation

The MLC component module DYNVMC was used to implement the Varian Millennium 120 MLC in BEAMnrc geometry code. The ability of the model to reproduce the MLC modulated dose distributions has been confirmed. MC dose distribution geometry, i.e. field widths, was found to be in excellent agreement with film measured dose distributions. Dosimetric comparison between film and MC in open field regions was problematic due inconsistent film response. For the purposes of verifying the mechanical correctness of the MLC model however, this shortcoming is of little consequence. Based on geometric reasoning alone, a large measurement bias near a field edge will only slightly affect the lateral position of the 50% point of that field edge.

Nonetheless, an initial objective for this validation process was to confirm the accuracy of MC leaf transmission. These results show that this is not likely possible with the measurement method employed. Lack of conclusive transmission measurements however, is not a significant obstacle for this work due the highly benchmarked dosimetric accuracy of the EGSnrc transport engine. For this work, the MLC model is considered mechanically sound and therefore appropriate for implementation into the MC treatment plan calculation framework

4.1.3 MC Treatment Plan Calculation

A framework for accurately calculating highly modulated treatment plans with the EGSnrc MC code has been developed and validated. Because the underlying open field beam model is highly accurate, and all the mechanical functionality of the MCTPCF has been verified, this tool can be used as a benchmark for other dose calculations for the same treatment parameters and geometry.

Four clinical RapidArc plans from separate anatomical sites were calculated with the MCTPCF for the purposes of validating the mechanical functionality of the framework. The resultant dose distributions were compared with those of Eclipse AAA. All comparisons showed a high level of agreement between the two calculation methods thus verifying the usefulness of this tool as a dosimetric benchmark for the development of the proposed error prediction methods. Again it is stressed that the validation of the mechanical functionality of the framework with AAA is a separate issue from validating the dosimetric accuracy of the transport engine.

The current implementation of the absolute dose calibration described by equation (2-4) is flawed and an alternative method (see equation (3-1)) has been implemented. This alternative method is useful for calibration in homogeneous geometries, as AAA and MC are expected to calculate nearly identically in such cases; however its accuracy in more complex geometries is questionable. For this reason, a third method (see section 3.3.3) which uses the output factor calculated by Eclipse was implemented for the two heterogeneous plans (lung and head-and-neck). This third implementation produced improved AAA to MC agreement in these more complex geometries.

Though the original purpose for the development of the MCTPCF was to aid in the validation of the proposed error prediction metrics, other applications exist. For example, because RapidArc treatments can be calculated in a just a few hours, with sufficient additional benchmarking, the MCTPCF can be used for clinical treatment plan quality assurance. Also, any user specified beam model can be used with the framework, granted that it has been properly parameterized and benchmarked.

4.1.4 Error Prediction

Because the FSI and LDI both successfully identify deviations between MC and AAA calculated dose, with further benchmarking, a clinical tool able to detect plans at high risk for AAA dose calculation errors, is completely feasible. The FSI index in particular, even if unable to identify errors being washed out by other deviations, is still a useful tool for identifying overlapping projections of small fields and blocks for highly modulated, multiple control point treatments. This is one direct result of using a voxel wise approach to treatment plan evaluation. By retaining the three dimensional mapping of a field dependent effect, the overlapping of that effect can then be calculated. The LDI will likely be less useful due to its effects only being evident in isolated lung regions; however it is completely possible that the open field model upon which it is based is too simple and with further development, its utility may increase. As discussed the HSI is still under development and its ability to identify the lateral scaling approximation of the AAA MC kernel under in complex treatment plans remains to be seen.

4.2 Future Work

4.2.1 Linac Model

As discussed in section 1.3.3, more physical specification of the electron beam source would require a more complex intensity distribution model as well as a more correct divergence model. In most modern megavoltage linacs, the electron source is magnetically steered into the target. Somewhere (on the order of a few cm) above the target these bending magnets will create an effective divergence point from which the electrons will spread into their true intensity and angular distribution. This would create a simple off-axis dependence of the electron source angle, and combined with the method developed by Bush *et al*.[60] a more correct source model could be implemented into the BEAMnrc code.

4.2.2 MLC Validation

As discussed in section 1.4, a more robust method to measure MLC validation profiles would be the utilization of a diamond detector in a water tank. The high resolution and low energy dependence of this type of detector would not only further validate the mechanical functionality of the virtual MLC, but also provide an accurate benchmarking of leaf transmission. This will be especially valuable in determining to what extent AAA fails to model leaf edge effects. While some LDI and HIS will only be applicable to AAA calculations, the FSI models behavior common to all Eclipse algorithms that use binary MLC blocking, including more recently implemented Acuros XB.

4.2.3 MC Treatment Plan Calculation

It is important to note that the dosimetric accuracy of a multiple field MCTPCF calculation inherently depends on the accuracy of the constituent control point fields. Therefore, any endeavor to further validate individual highly modulated MLC fields in complex geometries will be useful in benchmarking the framework, especially if it is to be used for clinical work. The best measurement tools for this work will be carefully calibrated ion chamber measurements in low-gradient dose distributions and diamond detector measurement for high-gradient distributions. Heterogeneous slab and anthropomorphic phantoms should be used with both open and highly modulated fields in order to achieve an in-house dosimetric benchmark of the framework. Again, the EGSnrc transport engine is highly validated and widely trusted, but significant in-house benchmarking will be required for any clinical calculation tool, especially for one as complex as the MCTPCF.

At this point, a significant shortcoming of the current implementation of the MCTPCF is the lack of a TPS-independent method of calibrating the MC *dpp*. As discussed in section 3.3.3, a simple volumetric normalization of the MC dose to the AAA dose may not be correct in heterogeneous geometries. For the reason, the monitor-chamber-based absolute dose calibration must be correctly implemented. Currently, the monitor chamber is modeled in the BEAMnrc geometry exactly as specified by the manufacturer, however the dose calibration according to equation (2-4) is not yielding a correct scaling for field sizes significantly different than that of the calibration field. In order for MCTPCF dose calculations of clinical plans to be used as a comparison benchmark for other algorithms, this monitor-chamber-based absolute dose calibration will need to function correctly.

Currently, the bulk of the MCTPCF is implemented in the cumbersome, and proprietary, MATLAB computing environment. A more efficient clinical implementation would require recoding the framework MATLAB scripts in a non-proprietary lower-level code like C++ or java. A graphical interface would also add significant value as not all clinicians will be comfortable with command line program control. This recoding process would ideally be carried out by a computer science professional.

In any form, the MCTPCF can only be feasibly implemented in a parallel computing environment with at least 100 available processors. Because an in-house computing cluster is a) expensive to build, b) likely requires dedicated maintenance staff, and c) requires a dedicated climate controlled space, cloud computing may be a viable alternative. The MCTPCF requires very little network overhead as the largest files that need to be transmitted are the on the order of tens of MB in size. These data streams could be easily encrypted, however if all patient data that does not reside on the in-house database is anonymized, encryption may not be needed. The cost and time requirements of implementing the MCTPCF in this way should be assessed.

4.2.4 Error Prediction

As discussed throughout section 3.4, when comparing the predicted error maps with MC/AAA deviations, correlations can be difficult to discern from simple visual inspection. For this reason, concrete mathematical correlation metrics should be investigated so that the prediction power of these methods can be properly quantified.

Ultimately, as stated in the introduction of this document, the error prediction methods introduced here can be used to develop a clinical tool which can be used as part of the treatment plan evaluation process. This will require significantly more development and benchmarking. The most promising form of such a tool would be a FSI-based complexity metric as it will be applicable a wider range of algorithms.

4.3 Clinical Significance of Error Prediction

The ultimate value of any treatment plan evaluation tool is whether or not it can improve the efficacy of the radiotherapy treatment, i.e. improve the outcome for the patient. At the current state of the science, it is unclear how the under or over-dosing of a few percent in a specific voxel of the target volume correlates with a concept like tumor control, however it is known that reducing the dose to normal tissue can be beneficial to a patient’s quality of life. Therefore, knowing that the calculated dose is accurate, even in a single spinal cord voxel, even to within a single percent, is of the utmost importance. As a secondary goal, the accuracy of the dose distribution should be maximized for the purpose of improving the correlation between treatment outcome and plan quality. The benefit to the patient may not be immediately realizable; however over time, the ability of the physician to associate a given tumor dose or coverage metric to a treatment outcome will be improved.

In clinical practice, the most accurate final dose calculation algorithm is not always used. The most common reason is that accuracy almost always comes at the expense of computation time. Other reasons may include software costs as well as a physician’s comfort with the format of the calculated result. For example, clinical MC algorithms still suffer from the same error estimation constraints as does their non-clinical (like EGSnrc) counterparts. For a MC calculated arc, statistical uncertainty in the regions surrounding the target volume will invariable be larger than that within it. Also, running that algorithm for tens of minutes, despite its higher level of accuracy, may not be warranted for simpler treatments. The question that arises is, when should a more accurate algorithm be used? The error prediction methods described in this work were developed to help answer this question.

Applicability of all three of the described error prediction metrics is restricted to a single, though widely implemented, clinical algorithm. In particular, the LDI and HSI are specifically tailored to the mechanics of the AAA algorithm. The FSI, on the other hand, will be applicable to any external beam dose calculation algorithm which models beam modulation as a binary process. This fact, and in conjunction with its demonstrated correlation with MC/AAA deviations, makes the FSI much more useful metric.

The utility of a clinical tool based on these methods will of course be limited by the importance of the predicted errors to the treatment plan evaluation process. The demonstrated maximum magnitude of the errors predicted by the FSI and FSI are on the order 3%. This is lower than the often quoted “5%” total allowable radiotherapy error, however this leaves a relatively small margin for many other sources of delivery error. In particular, setup uncertainties and organ motion are much more difficult to manage. All sources of systematic error should be reduced whenever possible, and when a more accurate dose calculation is readily available, knowing when to usevit is valuable knowledge. As discussed, the prevalence of low-density induced errors is small and may only be important in a relatively small fraction of lung geometries. Modulation induced errors however, will be common in almost any RapdidArc treatment. When target structures are very close to avoidance structures, for example the rectal wall in a prostate treatment, modulation error will typically be found in the dose gradient region which divides these structures. Understanding the true dose gradient can be important in reducing toxicity in such cases. Therefore, any method which alerts the physician to potential dose calculation error in a clinically significant region will be a useful tool.

References

1. Yu CX, Li XA, Ma L, Chen D, Naqvi S, Shepard D, et al. Clinical implementation of intensity-modulated arc therapy. Int J Radiat Oncol Biol Phys 2002;53(2):453–63.

2. Ulmer W, Harder D. Applications of a triple Gaussian pencil beam model for photon beam treatment planning. Z Med Phys 1996;6:68–74.

3. Fogliata A, Vanetti E, Albers D, Brink C, Clivio A, Knöös T, et al. On the dosimetric behaviour of photon dose calculation algorithms in the presence of simple geometric heterogeneities: comparison with Monte Carlo calculations. Phys Med Biol 2007;52(5):1363–85.

4. Esch AV, Tillikainen L, Pyykkonen J, Tenhunen M, Helminen H, Siljamaki S, et al. Testing of the analytical anisotropic algorithm for photon dose calculation. Med Phys 2006;33(11):4130–48.

5. Otto K. Volumetric modulated arc therapy: IMRT in a single gantry arc. Med Phys 2008;35(1):310–7.

6. Das IJ, Ding GX, Ahnesjo A. Small fields: Nonequilibrium radiation dosimetry. Med Phys 2008;35(1):206–15.

7. Ong CL, Cuijpers JP, Senan S, Slotman BJ, Verbakel WFAR. Impact of the calculation resolution of AAA for small fields and RapidArc treatment plans. Med Phys 2011;38(8):4471–9.

8. Vanetti E, Nicolini G, Nord J, Peltola J, Clivio A, Fogliata A, et al. On the role of the optimization algorithm of RapidArc volumetric modulated arc therapy on plan quality and efficiency. Med Phys 2011;38(11):5844–56.

9. Gagne IM, Ansbacher W, Zavgorodni S, Popescu C, Beckham WA. A Monte Carlo evaluation of RapidArc dose calculations for oropharynx radiotherapy. Phys Med Biol 2008;53(24):7167–85.

10. Bush K, Zavgorodni S, Gagne I, Townson R, Ansbacher W, Beckham W. Monte Carlo evaluation of RapidArc^{TM} oropharynx treatment planning strategies for sparing of midline structures. Phys Med Biol 2010;55(16):4465–79.

11. Raeside DE. Monte Carlo principles and applications. Phys Med Biol 1976;21(2):181.

12. Rogers DWO. Fifty years of Monte Carlo simulations for medical physics. Phys Med Biol 2006;51(13):R287–R301.

13. Andreo P. Monte Carlo techniques in medical radiation physics. Phys Med Biol 2000;36(7):861.

14. Bielajew A, Hirayama H, Nelson WR, Rogers DWO. History, Overview and Recent Improvement of EGS4 [Internet]. United States. Department of Energy. Office of Science; 2004 [cited 2012 Nov 20]. Available from: http://www.irs.inms.nrc.ca/papers/ps_versions/history_and_overview_of_EGS4.ps.gz

15. Kawrakow I, Mainegra-Hing E, Rogers DWO, Tessier F, Walters BRB. The EGSnrc Code System: Monte Carlo simulation of electron and photon transport [Internet]. Ottawa, Canada: National Research Council of Canada; 2000. Available from: http://irs.inms.nrc.ca/software/egsnrc/documentation/pirs701/index.html

16. Tillikainen L, Helminen H, Torsti T, Siljamäki S, Alakuijala J, Pyyry J, et al. A 3D pencil-beam-based superposition algorithm for photon dose calculation in heterogeneous media. Phys Med Biol 2008;53(14):3821–39.

17. Gagne IM, Zavgorodni S. Evaluation of the analytical anisotropic algorithm (AAA) in an extreme water-lung interface phantom using Monte Carlo dose calculations. J Appl Clin Med Phys [Internet] 2007 [cited 2012 Oct 30];8(1). Available from: http://www.jacmp.org/index.php/jacmp/article/view/2324

18. Aarup LR, Nahum AE, Zacharatou C, Juhler-Nøttrup T, Knöös T, Nyström H, et al. The effect of different lung densities on the accuracy of various radiotherapy dose calculation methods: Implications for tumour coverage. Radiother Oncol 2009;91(3):405–14.

19. Mohan R, Chui C, Lidofsky L. Energy and angular distributions of photons from medical linear accelerators. Med Phys 1985;12(5):592–7.

20. Liu HH, Mackie TR, McCullough EC. A dual source photon beam model used in convolution/superposition dose calculations for clinical megavoltage x-ray beams. Med Phys 1997;24(12):1960–74.

21. Hartmann-Siantar CLH, Walling RS, Daly TP, Faddegon B, Albright N, Bergstrom P, et al. Description and dosimetric verification of the PEREGRINE Monte Carlo dose calculation system for photon beams incident on a water phantom. Med Phys 2001;28(7):1322–37.

22. Fix MK, Stampanoni M, Manser P, Born EJ, Mini R, Rüegsegger P. A multiple source model for 6 MV photon beam dose calculations using Monte Carlo. Phys Med Biol 2001;46(5):1407–27.

23. Ding GX. Energy spectra, angular spread, fluence profiles and dose distributions of 6 and 18 MV photon beams: results of Monte Carlo simulations for a Varian 2100EX accelerator. Phys Med Biol 2002;47(7):1025–46.

24. Cho SH, Vassiliev ON, Lee S, Liu HH, Ibbott GS, Mohan R. Reference photon dosimetry data and reference phase space data for the 6 MV photon beam from Varian Clinac 2100 series linear accelerators. Med Phys 2005;32(1):137–48.

25. Udale M. A Monte Carlo investigation of surface doses for broad electron beams. Phys Med Biol 1988;33(8):939–53.

26. Cranmer-Sargison G, Beckham WA, Popescu IA. Modelling an extreme water–lung interface using a single pencil beam algorithm and the Monte Carlo method. Phys Med Biol 2004;49(8):1557–67.

27. Scott AJD, Nahum AE, Fenwick JD. Using a Monte Carlo model to predict dosimetric properties of small radiotherapy photon fields. Med Phys 2008;35(10):4671–84.

28. Ma C-M, Mok E, Kapur A, Pawlicki T, Findley D, Brain S, et al. Clinical implementation of a Monte Carlo treatment planning system. Med Phys 1999;26(10):2133–43.

29. Keall PJ, Siebers JV, Libby B, Mohan R. Determining the incident electron fluence for Monte Carlo-based photon treatment planning using a standard measured data set. Med Phys 2003;30(4):574–82.

30. Fippel M. Efficient particle transport simulation through beam modulating devices for Monte Carlo treatment planning. Med Phys 2004;31(5):1235–42.

31. Chetty IJ, Curran B, Cygler JE, DeMarco JJ, Ezzell G, Faddegon BA, et al. 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 2007;34(12):4818–53.

32. Yamamoto T, Mizowaki T, Miyabe Y, Takegawa H, Narita Y, Yano S, et al. An integrated Monte Carlo dosimetric verification system for radiotherapy treatment planning. Phys Med Biol 2007;52(7):1991–2008.

33. Teke T, Bergman AM, Kwa W, Gill B, Duzenli C, Popescu IA. Monte Carlo based, patient-specific RapidArc QA using Linac log files. Med Phys 2010;37(1):116–23.

34. Verhaegen F, Seuntjens J. Monte Carlo modelling of external radiotherapy photon beams. Phys Med Biol 2003;48(21):R107.

35. Lovelock DMJ, Chui CS, Mohan R. A Monte Carlo model of photon beams used in radiation therapy. Med Phys 1995;22(9):1387–94.

36. Faddegon BA, O'Brien P, Mason DLD. The flatness of Siemens linear accelerator x-ray fields. Med Phys 1999;26(2):220–8.

37. Libby B, Siebers J, Mohan R. Validation of Monte Carlo generated phase-space descriptions of medical linear accelerators. Med Phys 1999;26(8):1476–83.

38. Sheikh-Bagheri D, Rogers DWO. Sensitivity of megavoltage photon beam Monte Carlo simulations to electron beam and other parameters. Med Phys 2002;29(3):379–90.

39. Tzedakis A, Damilakis JE, Mazonakis M, Stratakis J, Varveris H, Gourtsoyiannis N. Influence of initial electron beam parameters on Monte Carlo calculated absorbed dose distributions for radiotherapy photon beams. Med Phys 2004;31(4):907–13.

40. Pena J, Gonzalez-Castano DM, Gomez F, Sanchez-Doblado F, Hartmann GH. Automatic determination of primary electron beam parameters in Monte Carlo simulation. Med Phys 2007;34(3):1076–84.

41. Chibani O, Ma C-MC. On the discrepancies between Monte Carlo dose calculations and measurements for the 18 MV Varian photon beam. Med Phys 2007;34(4):1206–16.

42. Chibani O, Moftah B, Ma C-MC. On Monte Carlo modeling of megavoltage photon beams: A revisited study on the sensitivity of beam parameters. Med Phys 2011;38(1):188–201.

43. Scott AJD, Nahum AE, Fenwick JD. Monte Carlo modeling of small photon fields: Quantifying the impact of focal spot size on source occlusion and output factors, and exploring miniphantom design for small-field measurements. Med Phys 2009;36(7):3132–44.

44. Grevillot L, Frisson T, Maneval D, Zahra N, Badel J-N, Sarrut D. Simulation of a 6 MV Elekta Precise Linac photon beam using GATE/GEANT4. Phys Med Biol 2011;56(4):903–18.

45. Almberg SS, Frengen J, Kylling A, Lindmo T. Monte Carlo linear accelerator simulation of megavoltage photon beams: Independent determination of initial beam parameters. Med Phys 2012;39(1):40–7.

46. Fix MK, Keall PJ, Siebers JV. Photon-beam subsource sensitivity to the initial electron-beam parameters. Med Phys 2005;32(4):1164–75.

47. KOCH HW, MOTZ JW. Bremsstrahlung Cross-Section Formulas and Related Data. Rev Mod Phys 1959;31(4):920–55.

48. Seltzer SM, Berger MJ. Bremsstrahlung spectra from electron interactions with screened atomic nuclei and orbital electrons. Nucl Instrum Methods Phys Res Sect B Beam Interact Mater At 1985;12(1):95–134.

49. Fragoso M, Kawrakow I, Faddegon BA, Solberg TD, Chetty IJ. Fast, accurate photon beam accelerator modeling using BEAMnrc: A systematic investigation of efficiency enhancing methods and cross-section data. Med Phys 2009;36(12):5451–66.

50. Faddegon BA, Asai M, Perl J, Ross C, Sempau J, Tinslay J, et al. Benchmarking of Monte Carlo simulation of bremsstrahlung from thick targets at radiotherapy energies. Med Phys 2008;35(10):4308–17.

51. Tessier F, Kawrakow I. Calculation of the electron–electron bremsstrahlung cross-section in the field of atomic electrons. Nucl Instrum Methods Phys Res Sect B Beam Interact Mater At 2008;266(4):625–34.

52. Brice DK. Stopping powers for electrons and positrons (ICRU report 37; International commission on radiation units and measurements, Bethesda, Maryland, USA, 1984). Nucl Instrum Methods Phys Res Sect B Beam Interact Mater At 1985;12(1):187–8.

53. Ali ESM, McEwen MR, Rogers DWO. Unfolding linac photon spectra and incident electron energies from experimental transmission data, with direct independent validation. Med Phys 2012;39(11):6585–96.

54. Ali ESM, McEwen MR, Rogers DWO. Detailed high-accuracy megavoltage transmission measurements: A sensitive experimental benchmark of EGSnrc. Med Phys 2012;39(10):5990–6003.

55. Aubin JS, Steciw S, Kirkby C, Fallone BG. An integrated 6 MV linear accelerator model from electron gun to dose in a water tank. Med Phys 2010;37(5):2279–88.

56. Constantin M, Perl J, LoSasso T, Salop A, Whittum D, Narula A, et al. Modeling the TrueBeam linac using a CAD to Geant4 geometry implementation: Dose and IAEA-compliant phase space calculations. Med Phys 2011;38(7):4018–24.

57. Sawkey D, Faddegon BA. Simulation of large x-ray fields using independently measured source and geometry details. Med Phys 2009;36(12):5622–32.

58. Smedt BD, Reynaert N, Flachet F, Coghe M, Thompson MG, Paelinck L, et al. Decoupling initial electron beam parameters for Monte Carlo photon beam modelling by removing beam-modifying filters from the beam path. Phys Med Biol 2005;50(24):5935–51.

59. Van Laere K, Mondelaers W. Full Monte Carlo simulation and optimization of a high-power bremsstrahlung converter. Radiat Phys Chem 1997;49(2):207–19.

60. Cheung T, Butson MJ, Yu PKN. Measurement of high energy x-ray beam penumbra with Gafchromic[trademark sign] EBT radiochromic film. Med Phys 2006;33(8):2912–4.

61. Butson MJ, Yu PK., Cheung T, Metcalfe P. Radiochromic film for medical radiation dosimetry. Mater Sci Eng R Rep 2003;41(3-5):61–120.

62. Muench PJ, Meigooni AS, Nath R, McLaughlin WL. Photon energy dependence of the sensitivity of radiochromic film and comparison with silver halide film and LiF TLDs used for brachytherapy dosimetry. Med Phys 1991;18(4):769–75.

63. Sutherland JGH, Rogers DWO. Monte Carlo calculated absorbed-dose energy dependence of EBT and EBT2 film. Med Phys 2010;37(3):1110.

64. Micke A, Lewis DF, Yu X. Multichannel film dosimetry with nonuniformity correction. Med Phys 2011;38(5):2523–34.

65. Mayer RR, Ma F, Chen Y, Miller RI, Belard A, McDonough J, et al. Enhanced dosimetry procedures and assessment for EBT2 radiochromic film. Med Phys 2012;39(4):2147–55.

66. Bielajew AF. Improved angular sampling for pair production in the EGS4 code system. Rep PIRS-0287 Natl Res Counc Can [Internet] 1991 [cited 2012 Nov 20];Available from: http://irs.inms.nrc.ca/publications/reports/pdf/PIRS-0287r-1994.pdf

67. Bielajew AF, Mohan R, Chui CS. Improved bremsstrahlung photon angular sampling in the EGS4 code system. Natl Res Counc Can Rep PIRS-0203 [Internet] 1989 [cited 2012 Nov 20];Available from: http://irs.inms.nrc.ca/publications/reports/pdf/PIRS-0203-1989.pdf

68. Bielajew AF, Rogers DWO. Presta: The parameter reduced electron-step transport algorithm for electron monte carlo transport. Nucl Instrum Methods Phys Res Sect B Beam Interact Mater At 1986;18(1–6):165–81.

69. Ford RL, Nelson WR. Egs Code System: Computer Programs for the Monte Carlo Simulation of Electromagnetic Cascade Showers. Version 3. [egs, Pegs, Testsr, in Mortran] [Internet]. Stanford Linear Accelerator Center, CA (USA); 1978 [cited 2013 Oct 19]. Available from: http://www.osti.gov/scitech/biblio/6533126

70. Nelson WR, Rogers DWO, Hirayama H. The Egs4 Code System. 1985 [cited 2013 Oct 19];Available from: http://inspirehep.net/record/220592

71. Rogers DWO, Faddegon BA, Ding GX, Ma C-M, We J, Mackie TR. BEAM: A Monte Carlo code to simulate radiotherapy treatment units. Med Phys 1995;22(5):503–24.

72. Walters BRB, Kawrakow I, Rogers DWO. DOSXYZnrc Users Manual [Internet]. Ottawa, Canada: National Research Council of Canada; 2004. Available from: http://irs.inms.nrc.ca/software/beamnrc/documentation/pirs794/pirs794.html

73. Bush K, Townson R, Zavgorodni S. Monte Carlo simulation of RapidArc radiotherapy delivery. Phys Med Biol 2008;53(19):N359–N370.

74. Lobo J, Popescu IA. Two new DOSXYZnrc sources for 4D Monte Carlo simulations of continuously variable beam configurations, with applications to RapidArc, VMAT, TomoTherapy and CyberKnife. Phys Med Biol 2010;55(16):4431–43.

75. Ulmer W, Pyyry J, Kaissl W. A 3D photon superposition/convolution algorithm and its foundation on results of Monte Carlo calculations. Phys Med Biol 2005;50(8):1767–90.

76. Tillikainen L, Siljamäki S, Helminen H, Alakuijala J, Pyyry J. Determination of parameters for a multiple-source model of megavoltage photon beams using optimization methods. Phys Med Biol 2007;52(5):1441–67.

77. Bueno M, Paganetti H, Duch MA, Schuemann J. An algorithm to assess the need for clinical Monte Carlo dose calculation for small proton therapy fields based on quantification of tissue heterogeneity. Med Phys 2013;40(8):081704.

78. Pflugfelder D, Wilkens JJ, Szymanowski H, Oelfke U. Quantifying lateral tissue heterogeneities in hadron therapy. Med Phys 2007;34(4):1506–13.

79. Disher B, Hajdok G, Gaede S, Battista JJ. An in-depth Monte Carlo study of lateral electron disequilibrium for small fields in ultra-low density lung: implications for modern radiation therapy. Phys Med Biol 2012;57(6):1543.

80. Younge KC, Matuszak MM, Moran JM, McShan DL, Fraass BA, Roberts DA. Penalization of aperture complexity in inversely planned volumetric modulated arc therapy. Med Phys 2012;39(11):7160–70.

81. Masi L, Doro R, Favuzza V, Cipressi S, Livi L. Impact of plan parameters on the dosimetric accuracy of volumetric modulated arc therapy. Med Phys 2013;40(7):071718.

82. McNiven AL, Sharpe MB, Purdie TG. A new metric for assessing IMRT modulation complexity and plan deliverability. Med Phys 2010;37(2):505–15.

83. Giorgia N, Antonella F, Eugenio V, Alessandro C, Filippo A, Luca C. What is an acceptably smoothed fluence? Dosimetric and delivery considerations for dynamic sliding window IMRT. Radiat Oncol Lond Engl 2007;2:42.

84. Kawrakow I. On the de-noising of Monte Carlo calculated dose distributions. Phys Med Biol 2002;47(17):3087–103.

85. Kapur A, Ma CM, Boyer AL. Monte Carlo simulations for multileaf collimator leaves: design and dosimetry. Med Phys 2000;47:40.

86. Siebers JV, Keall PJ, Kim JO, Mohan R. A method for photon beam Monte Carlo multileaf collimator particle transport. Phys Med Biol 2002;47(17):3225–49.

87. Heath E, Seuntjens J. Development and validation of a BEAMnrc component module for accurate Monte Carlo modelling of the Varian dynamic Millennium multileaf collimator. Phys Med Biol 2003;48(24):4045–63.

88. Low DA, Harms WB, Mutic S, Purdy JA. A technique for the quantitative evaluation of dose distributions. Med Phys 1998;25(5):656–61.

89. Siebers JV, Keall PJ, Nahum AE, Mohan R. Converting absorbed dose to medium to absorbed dose to water for Monte Carlo based photon beam dose calculations. Phys Med Biol 2000;45(4):983–95.

90. Sterpin E, Tomsej M, Smedt BD, Reynaert N, Vynckier S. Monte Carlo evaluation of the AAA treatment planning algorithm in a heterogeneous multilayer phantom and IMRT clinical treatments for an Elekta SL25 linear accelerator. Med Phys 2007;34(5):1665–77.

91. Hoffmann L. Implementation and experimental validation of the high dose rate stereotactic treatment mode at Varian accelerators. Acta Oncol 2009;48(2):201–8.

92. Zhu TC, Bjarngard BE. The head-scatter factor for small field sizes. Med Phys 1994;21(1):65–8.

93. Zhu TC, Bjarngard BE, Shackford H. X-ray source and the output factor. Med Phys 1995;22(6):793–8.

94. Zhu TC, Bjarngard BE. The fraction of photons undergoing head scatter in X-ray beams. Phys Med Biol 1995;40(6):1127–34.

95. Caprile P, Hartmann GH. Development and validation of a beam model applicable to small fields. Phys Med Biol 2009;54(10):3257–68.

96. Sterpin E, Chen Y, Lu W, Mackie TR, Olivera GH, Vynckier S. On the relationships between electron spot size, focal spot size, and virtual source position in Monte Carlo simulations. Med Phys 2011;38(3):1579–86.

97. Sawkey DL, Faddegon BA. Determination of electron energy, spectral width, and beam divergence at the exit window for clinical megavoltage x-ray beams. Med Phys 2009;36(3):698–707.

98. Bush K, Zavgorodni S, Beckham W. Inference of the optimal pretarget electron beam parameters in a Monte Carlo virtual linac model through simulated annealing. Med Phys 2009;36(6):2309–19.

Appendix A Beam Modeling Code

Cluster Submission Script

run_series.m

```
#!/usr/bin/python

# This script runs a series of simulations, varying certain parameters.
# It only works for non-CT geometries (NMED = 0), and dosxyz isourse = 9.

# imports
from subprocess import call
import math

# paths
linac = 'Clinac6X_noMLC'
phantom = 'head_val'
access_dir = '/nfs/stak/students/e/egana/'
beam_dir = access_dir + 'egsnrc/BEAM_' + linac + '/'
dosxyz_dir = access_dir + 'egsnrc/dosxyznrc/'

# study and number of jobs
BT = 'NRC'
BA = 'KM'
NJ = 300

# field sizes
FS_all = [3, 6, 10, 20, 30, 40]

# number of histories
# for 0.5% sigma
NH_all = [50000000, 100000000, 500000000, 1000000000, 2000000000, 3500000000]

# for 1.0% sigma
#NH_all = [10000000, 30000000, 50000000, 200000000, 200000000, 350000000]

# test
#NH_all = [100000, 100000, 100000, 100000, 100000, 100000]

# jaw settings
with open(access_dir + 'egsnrc/scripts/jaw_settings', 'r') as JS_fid:
        JS_lines = JS_fid.readlines()
        Xx_top = map(float,JS_lines[0].split(','))
        Xz_top = map(float,JS_lines[1].split(','))
        Xx_bot = map(float,JS_lines[2].split(','))
        Xz_bot = map(float,JS_lines[3].split(','))

# parameter ranges
FS_sel = [3, 20, 40]
#FS_sel = [10, 30]
E_sel = [5.89, 5.91]
I_sel = [0.1, 0.2]
D_sel = [0.9, 1.0, 1.1]

# change BEAM input file parameters and run jobs
for FS in FS_sel:
        temp = NH_all[FS_all.index(FS)]
        NH = float("%3.0g" % temp)
        for E in E_sel:
```

```
			for I in I_sel:
				for D in D_sel:

					# form parameter string for naming
					pars = "_%s%s_%s_%0.2f_%0.2f_%0.1f" % (BT, BA, FS, E, I, D)

					# modify dosxyz input file
					with open(dosxyz_dir + phantom + '.egsinp', 'r') as
phantom_fid:
						phantom_lines = phantom_fid.readlines()

						# modify linac and BEAM input file names
						args = map(int,phantom_lines[5].split(','))
						rec = (-1 * sum(args[:3]) + 15)
						args = phantom_lines[rec - 1].split(',')
						args[0] = 'BEAM_' + linac
						args[1] = linac + pars
						phantom_lines[rec - 1] = ",".join(args)

						# modify number of particles
						args = phantom_lines[23].split(',')
						args[0] = "%i" % NH
						phantom_lines[23] = ",".join(args)

					# write new dosxyz input file
					with open(dosxyz_dir + phantom + pars + '.egsinp', 'w') as
phantom_fid:
						phantom_fid.writelines(phantom_lines)

					# modify BEAM input file
					with open(beam_dir + linac + '.egsinp', 'r') as linac_fid:
						linac_lines = linac_fid.readlines()

						# find relevant record numbers
						for num, line in enumerate(linac_lines,1):
							if 'CM JAWS' in line:
								JS_rec = num
							if 'Brems angular sampling' in line:
								BS_rec = num

						# modify DB radius
						args = linac_lines[4].split(',')
						rad = (float(FS) / 2) + 5
						args[0] = "%0.2f" % rad
						linac_lines[4] = ",".join(args)

						# modify I and D
						args = linac_lines[5].split(',')
						args[2] = args[7] = " -%0.2f" % I
						args[6] = " %0.2f" % D
						linac_lines[5] = ",".join(args)

						# modify E
						linac_lines[7] = "%0.2f\n" % E

						# y settings
						args = linac_lines[JS_rec + 4].strip().split(',')
						args[0:2] = map(str,map(float,args[0:2]))
						args[2:4] = map(str,[(float(z) * float(FS) / 2 / 100)
for z in args[:2]])
						args[4:6] = map(str,[(-1 * float(z) * float(FS) / 2 /
100) for z in args[:2]])
						linac_lines[JS_rec + 4] = ", ".join(args) + '\n'

						# x settings
						args = linac_lines[JS_rec + 6].strip().split(',')
```

```
                                        args[0] = str(Xz_top[FS_all.index(FS)])
                                        args[1] = str(Xz_bot[FS_all.index(FS)])
                                        args[2] = str(Xx_top[FS_all.index(FS)])
                                        args[3] = str(Xx_bot[FS_all.index(FS)])
                                        args[4] = str(-1*float(args[2]))
                                        args[5] = str(-1*float(args[3]))
                                        linac_lines[JS_rec + 6] = ", ".join(args) + '\n'

                                        # modify bremsstrahlung settings
                                        args = linac_lines[BS_rec - 1].strip().split(' ')
                                        if BA == "S":
                                                args[-1] = "Simple"
                                        else:
                                                args[-1] = BA
                                        linac_lines[BS_rec - 1] = " " + " ".join(args) + '\n'
                                        args = linac_lines[BS_rec].strip().split(' ')
                                        args[-1] = BT
                                        linac_lines[BS_rec] = " " + " ".join(args) + '\n'

                                # write new BEAM input file
                                with open(beam_dir + linac + pars + '.egsinp', 'w') as
linac_fid:
                                        linac_fid.writelines(linac_lines)

                                # send jobs to cluster
                                call(['sh', access_dir + 'egsnrc/scripts/dosxyz_parallel.sh',
phantom + pars, str(NJ)])
```

MC vs. Measured Comparison Scripts

analyze.m

```
% This function compares BEAMnrc/DOSXYZnrc Monte Carlo (MC) data with
% measured data. It produces comparison statistics and
% plots for the purposes of commissioning the linac head model.

function [data_cell] = analyze(MC_file,plim,dlim,fast,smooth,x2,M_data)

%----------STUDY PARAMETERS AND FILE NAMES---------------------------------

% Extract names and paramteters from the MC file name.
job_dir=regexprep(MC_file,'(.*)/.*$','$1');
dir_name=regexprep(job_dir,'.*/([\w.]*)$','$1');
FS_char=regexprep(dir_name,'\w*_(\d*)_[\d.]*_[\d.]*_[\d.]*','$1');
BS=regexprep(dir_name,'\w*_(\w*)_\d*_[\d.]*_[\d.]*_[\d.]*','$1');
E_char=regexprep(dir_name,'\w*_\d*_([\d.]*)_[\d.]*_[\d.]*','$1');
I_char=regexprep(dir_name,'\w*_\d*_[\d.]*_([\d.]*)_[\d.]*','$1');
D_char=regexprep(dir_name,'\w*_\d*_[\d.]*_[\d.]*_([\d.]*)','$1');

%----------READ IN AND FORMAT DATA-----------------------------------------

% Read in the MC data and then execute MC_process on the raw data for
% symmetrizing and smoothing.
D=[1.5 5 10 20 30];
[MC_3d_dpp,MC_3d_unc,X_raw,Y_raw,Z_raw]=MC_read(MC_file);
[MC_x_dpp,MC_x_unc,MC_z_dpp,MC_z_unc,X,Z]=...
    MC_process(MC_3d_dpp,MC_3d_unc,X_raw,Y_raw,Z_raw,smooth,D,x2);

% Read in the selected measured data set.  The choices are either the
% Eclipse Beam Data (EBD) or the in house measured (OHSU).
fs=str2double(FS_char);
if strcmp(M_data,'EBD') % read in Eclipse data
```

```
    [M_x_cell,M_z_data]=GB_read(fs,D);
    M_z_norm=interp1(M_z_data(:,2),M_z_data(:,1),Z,'linear',0);
    M_x_norm=WT_prof_format(M_x_cell,X);
elseif strcmp(M_data,'OHSU') % read in WT data
    M_file='/home/alex/Projects/Beam Model/OHSU Measured/2009/X06 in water scans.mcc';
    [WT_z_norm,WT_Z]=WT_read(M_file,fs,0,'PDD');
    M_z_norm=interp1(WT_Z,WT_z_norm,Z,'linear',0);
    M_x_cell=cell(numel(D),1);
    for d=1:numel(D)
        [WT_x_norm,WT_X]=WT_read(M_file,fs,D(d),'INPLANE');
        M_x_cell(d)={[WT_x_norm WT_X]};
    end
    M_x_norm=WT_prof_format(M_x_cell,X);
end

% Normalize everything to one point on a 3rd degree polynomial fit of depth
% dose curve at 10 cm. Scale profiles by their depth-dose CAX intersection.
% This is a type A normalization.
MC_z_fit_10=polyval(polyfit(Z(Z>=5 and Z<=15),MC_z_dpp(Z>=5 and Z<=15),3),10);
MC_z_norm=MC_z_dpp/MC_z_fit_10;
M_z_fit_10=polyval(polyfit(Z(Z>=5 and Z<=15),M_z_norm(Z>=5 and Z<=15),3),10);
M_z_norm=M_z_norm/M_z_fit_10;
MC_z_unc=sqrt(MC_z_unc.^2+MC_z_unc(Z==10)^2); % MC FDD uncertainty
MC_x_norm=zeros(size(MC_x_dpp));
for d=1:numel(D)
    MC_x_norm(:,d)=MC_x_dpp(:,d)/MC_x_dpp(1,d)*MC_z_norm(Z==D(d));
    MC_x_unc=sqrt(MC_x_unc.^2+MC_z_unc(Z==10)^2+MC_z_unc(Z==D(d))^2);
    M_x_norm(:,d)=M_x_norm(:,d)/M_x_norm(1,d)*M_z_norm(Z==D(d));
end

% Calculate the locally weighted percent deviation (PD) of MC from the
% measured data.  Note the that the MC uncertainty must also be weighted by
% the measured data.
dev_z=(MC_z_norm-M_z_norm)./M_z_norm*100;
dev_z_unc=MC_z_unc.*MC_z_norm./M_z_norm*100;
dev_x=(MC_x_norm-M_x_norm)./M_x_norm*100;
dev_x_unc=MC_x_unc.*MC_x_norm./M_x_norm*100;

%----------DEPTH-DOSE ANALYSIS---------------------------------------------

% Check if points in the build-up regin meet the 3/3 threshold criteria.
BUi=(1:4); % BU region indices
BUn=numel(BUi);
BUcz=interp1(M_z_norm(BUi),Z(BUi),MC_z_norm(BUi),'spline'); % comparison z values
BUcd=interp1(Z,M_z_norm,Z(BUi),'spline'); % comparison dose values
BUp=numel(find(abs(BUcz-Z(BUi))<0.3 | (BUcd-MC_z_norm(BUi))./BUcd*100<3)); % PD < 3% or z
dev. < 3mm

% Check if points in the depth-dose fall-off (FO) region meet the specified
% threshold.
FOi=find(Z>1.5 and Z<=30); % FO region boundary indices
FOn=numel(FOi);
FOp=numel(find(abs(dev_z(FOi))<plim)); % PD > 1%
X2=sqrt(mean((dev_z(FOi)./dev_z_unc(FOi)).^2)); % FO region X2
% X2=sqrt(mean((dev_z(FOi)/plim).^2)); % FO region X2

% Find the max and mean uncertainty in FO region.
max_unc=zeros(numel(D),1);
mean_unc=zeros(numel(D),1);
max_unc(1)=max(dev_z_unc(FOi));
mean_unc(1)=mean(dev_z_unc(FOi));

% Extract the DOSXYZ input file name and # of particles run.
phant_file=regexprep(MC_file,'.3ddose','.egsinp');
job_lines=textscan(fopen(phant_file),'%s','Delimiter','\n');
job_lines=job_lines{1};
```

```
rec=find(~cellfun(@isempty,strfind(job_lines,'BEAM_')),1);
name_line=textscan(job_lines{rec},'%s','Delimiter',',');
LINAC_file=name_line{1}{2};
N=textscan(job_lines{rec+1},'%s',1,'Delimiter',',');
N=str2double(N{1});

%----------PROFILE ANALYSIS-----------------------------------------------

% Build some arrays.
np=numel(D);
PLn=zeros(1,np);
PLp=zeros(1,np);
PNn=zeros(1,np);
PNp=zeros(1,np);
X2=[X2; zeros(np,1)];
MC_x_50=zeros(np,1);
proj_x_50=zeros(np,1);
adj=zeros(np,1);

% Execute PN_region in order to separate the profile plateau and penumbra
% regions.
PNi=PN_region(fs,D,MC_x_norm);

% Loop through each profile.
for d=1:np

    % Check if points in the penumbra region (PN) meet the specified DTA
    % threshold.
    PNi_temp=PNi{d};
    PNii=(PNi_temp(1)-1:PNi_temp(end)+1);
    PNcx=interp1(MC_x_norm(PNii,d),X(PNii),M_x_norm(PNi_temp,d)); % interp x compare values
    PNn(d)=numel(PNi_temp); % N of PN region points
    PNp(d)=numel(find(abs(PNcx-X(PNi_temp))<dlim | abs(dev_x(PNi_temp,d))<plim)); % peneumbra
pass (1% or 1mm)

    % Check if points in the plateau region (PL) meet the specified PD
    % threshold.
    PLi=(1:PNi_temp(1)-1);
    PLn(d)=numel(PLi);
    PLp(d)=numel(find(abs(dev_x(PLi,d))<plim));
    X2(d+1)=sqrt(mean((dev_x(PLi,d)./dev_x_unc(PLi,d)).^2)); % PL region X2
    %     X2(d+1)=sqrt(mean((dev_x(PLi,d)/plim).^2)); % PL region X2

    % Locate the 50% dose value in the penumbra region for jaw tuning.
    % Deviations between this value and the projected jaw position will be
    % printed to the .stat file below.
    proj_x_50(d)=(100+D(d))/100*fs/2;
    MC_x_50(d)=interp1(MC_x_norm(PNi_temp,d),X(PNi_temp),0.5*MC_z_norm(Z==D(d))); % MC %50
dose x value
    adj(d)=100*(proj_x_50(d)-MC_x_50(d))/(100+D(d)); % difference in %50 dose values

    % Find the max and the mean uncertainty in PL region.
    max_unc(d+1)=max(dev_x_unc(PLi,d));
    mean_unc(d+1)=mean(dev_x_unc(PLi,d));
end

%----------STORE DATA FOR OUTPUT-------------------------------------------

% Build the data cell. This will house the analysis data so that it can be
% used later by other analysis tools.
data_cell=cell(20,1);

% Define the job paramters that will be used to query the data.
data_cell(1)={FS_char};
data_cell(2)={BS};
data_cell(3)={E_char};
data_cell(4)={I_char};
data_cell(5)={D_char};

% Fill the cell with the comparison analysis values.
data_cell(6)={N};
data_cell(7)={max_unc};
data_cell(8)={X2};
data_cell(9)={BUp};
data_cell(10)={BUn};
data_cell(11)={FOp};
data_cell(12)={FOn};
data_cell(13)={PLp};
data_cell(14)={PLn};
data_cell(15)={PNp};
data_cell(16)={PNn};
data_cell(17)={mean_unc};
data_cell(18)={plim};
data_cell(19)={dlim};
```

MC_process.m

```
% This function applies Savitzey-Golay and ion chamber smoothing to the
% data set.

function [MC_x_dpp,MC_x_unc,MC_z_dpp,MC_z_unc,X,Z]=...
    MC_process(MC_3d_dpp,MC_3d_unc,X_raw,Y_raw,Z_raw,smooth,P,x2)

% Interpolate xz plane onto a finer grid so that it can be smoothed.
% by the ion chamber filter.
MC_2d_dpp=squeeze(MC_3d_dpp(Y_raw==0,:,:))';
MC_2d_unc=squeeze(MC_3d_unc(Y_raw==0,:,:))';
fine_vox=0.25;
Z_fine=(0:fine_vox:Z_raw(end))';
X_fine=(0:fine_vox:X_raw(end))';
MC_2d_dpp_fine=interp2(X_raw(2:end),Z_raw(1:end-1),MC_2d_dpp(1:end-
1,2:end),X_fine,Z_fine','spline');
MC_2d_unc_fine=interp2(X_raw(2:end),Z_raw(1:end-1),MC_2d_unc(1:end-
1,2:end),X_fine,Z_fine','spline');

% Extract 1d z data and apply SG and IC smoothing.
IC_rad=0.3; % cm
MC_z_dpp_fine=interp1(Z_fine,MC_2d_dpp_fine(:,X_raw==0),Z_fine,'linear','extrap');
MC_z_unc_fine=interp1(Z_fine,MC_2d_unc_fine(:,X_raw==0),Z_fine,'linear','extrap');
if smooth==1
    [MC_z_dpp_fine,MC_z_unc_fine]=SG_smooth(MC_z_dpp_fine,MC_z_unc_fine,10,x2);
    [MC_z_dpp_fine,MC_z_unc_fine]=IC_smooth(MC_z_dpp_fine,MC_z_unc_fine,IC_rad,fine_vox);
end

% Trim the extra (x<0) voxel from the x grid.
X=X_raw(2:end);

% Add a zero to original MC z grid. This is basically a padding value.
Z=[0; Z_raw(2:end-1)];

% Interpoalate the z data back onto the original grid.
MC_z_dpp=interp1(Z_fine,MC_z_dpp_fine,Z,'linear');
MC_z_unc=interp1(Z_fine,MC_z_unc_fine,Z,'linear');

% Extract 1d x data from the xz plane.
MC_x_dpp=interp2(X_fine,Z_fine,MC_2d_dpp_fine,X,P);
MC_x_unc=interp2(X_fine,Z_fine,MC_2d_unc_fine,X,P);

% Pad 1d x data and filter in the negative x direciton so that smoothing
% can be applied to the CAX region. Then smooth the x data.
MC_x_dpp_ext=[flipdim(MC_x_dpp(:,find(X==0)+1:find(X==3)),2) MC_x_dpp]';
MC_x_unc_ext=[flipdim(MC_x_unc(:,find(X==0)+1:find(X==3)),2) MC_x_unc]';
X_ext=[flipdim(-X(find(X==0)+1:find(X==3)),1);X(find(X==0):end)];
if smooth==1
    for p=1:numel(P)

[MC_x_dpp_ext(:,p),MC_x_unc_ext(:,p)]=SG_smooth(MC_x_dpp_ext(:,p),MC_x_unc_ext(:,p),20,x2);

[MC_x_dpp_ext(:,p),MC_x_unc_ext(:,p)]=IC_smooth(MC_x_dpp_ext(:,p),MC_x_unc_ext(:,p),IC_rad,fi
ne_vox);

    end
end

% Remove the padded values such that profiles on the original grid remain.
MC_x_dpp=MC_x_dpp_ext(find(X_ext==0):end,:);
MC_x_unc=MC_x_unc_ext(find(X_ext==0):end,:);
```

SG_smooth.m

```
% This funtion applies 1d 2nd order daptive Savitzy-Golay filter to the
% data. It essentially moves across the data applying a 2nd order
% polynomial fit to regions of varied size. If smoothing is applied to a
% region, X^2 goodness-of-fit statistic must be lower than the specified
% threshold. The algorithm also takes a window size threshold that limits
% the lenght of any smoothed segment.

function [MCd_sm,MCe_sm]=SG_smooth(MCd,MCe,win_max,chi2_max)

% Build some arrays.
MCd_sm=zeros(numel(MCd),1);
MCe_sm=zeros(numel(MCe),1);
chi2=zeros(numel(MCd),1);
win=zeros(numel(MCd),1);

% For each point in the distribtion...
for i=1:numel(MCd)
    win(i)=win_max;
    chi2(i)=chi2_max+1;
    while i>numel(MCd)-win(i) || i<=win(i)
        win(i)=win(i)-1;
    end
    while chi2(i)>chi2_max
        if win(i)<=1
            MCd_sm(i)=MCd(i);
            MCe_sm(i)=MCe(i);
            break
        else
            x=[-(win(i):-1:1) 0 (1:win(i))]';
            N=numel(x);
            x2=x.*x;
            x4=x2.*x2;
            d=MCd(i-win(i):i+win(i));

            dx=d.*x;
            dx2=d.*x2;
            c1=sum(d);
            c2=sum(dx);
            c3=sum(dx2);
            s2=sum(x2);
            s4=sum(x4);
            b0=(s2*c1-c3)/(s2*(N-1));
            b1=c2/s2;
            b2=(N*c3-s2*c1)/(s4*(N-1));
            e=MCe(i-win(i):i+win(i));
            MCd_sm(i)=b0;
            MCe_sm(i)=sqrt(sum(e.^2))/N;
            chi2(i)=sum(((b0+b1*x+b2*x2-d)./(e.*d)).^2)/(N-3);
%             if i==win_max+20
%                 plot(i+x,b0+b1*x+b2*x2,'*b')
%                 hold on
%                 plot(i+x,d,'or')
%             end
            win(i)=win(i)-1;
        end
    end
end
[win chi2];
```

Ion Chamber Filter Script: IC_smooth

```
% This funtion applies an averaging filter to 1d data to simulate IC
% measurement.  The averaging is weighted by the chord of a circle centered
% at point interest.

function [MCd_sm,MCe_sm] = IC_smooth (MC_dose,MC_unc,rad,vox)

% Build the circular filter.
grid=2*ceil((2*rad-vox)/2/vox)+1;
filt=zeros(grid,grid);
c=ceil(grid/2);
for i=1:grid
    for j=1:grid
        d=round(1000*vox*sqrt((i-c)^2+(j-c)^2))/1000;
        if d<=rad
            filt(i,j)=1;
        end
    end
end
dim=1;
if dim==1
    filt=filt(c,:);
    filt=filt/sum(filt);
else
    filt=filt/sum(sum(filt));
end

% Apply the filter to the data.
MCd_sm=conv(MC_dose,filt,'same');
MCe_sm=sqrt(conv(MC_unc.^2,filt,'same'));
```

Penumbra Region Definition Script: PN_region

```
% This function takes normalized profile data and finds the penumbra region
% indices.

function [PNi]=PN_region(fs,P,MCfx)

% Use empirically found gradient limits for each field size
% in order to plot gradient limit dependency.
a=polyfit([3 10 20 40],[-0.04 -0.03 -0.025 -0.016],2);

% For each profile...
PNi=cell(numel(P),1);
for p=1:numel(P)

    % Find points with gradient > grad_lim
    grad_lim=(a(3)+a(2)*fs+a(1)*fs^2)*100/(100+P(p))*MCfx(1,p);
    PNi_temp=find(gradient(MCfx(:,p))<grad_lim);

    % Make sure there are no gaps in the indices.
    gaps=find((PNi_temp(2:end)-PNi_temp(1:end-1))>1);
    while gaps~=0 % test for gaps between negative MC gradient values
        grad_lim=grad_lim-0.001;
        PNi_temp=find(gradient(MCfx(:,p))<grad_lim);
        gaps=find((PNi_temp(2:end)-PNi_temp(1:end-1))>1);
    end
    PNi(p)={PNi_temp};
end
```

Source Parameter Optimization Scripts

par_analyze.m

```
% This script analyzes and plots the objective function for a single
% bremsstrahlung combination.

close all

% Specify some analysis parameters.
M_data='EBD'; % Specify the measured data set to use for comparison {OHSU,GB}.
BS_name='NRC/KM'; % Specify the brem combination to analyze.
plim=1.3; % Specify the percent deviation threshold.
dlim=0.1; % Specify the DTA threshold.
recalc=1; % Set to one to reanalyze all simulation data.
smooth=1; % Set to  one to SG smooth the MC data.
x2=2; % Specify the chi-square limit for SG smoothing.

% Set the plotting weights and thresholds.
X2_w=0; % Weighting of the chi-squared term of the OF_X2.
ROI_w=2; % Weighting of ROI term of the OF_X2.
PN_w=1; % Weighting of the chi-squared term of the OF_X2.
data_lims=[0.7 1.0];
plot_thresh=0; % Set to one to plot the points meeting the OF_X2 threshold.
OF_X2_thresh=0.8; % Specify OF_X2 threshold.

% Set directory names. Each BCS_analyze_1 job_data cells will vary with
% the type of measured data used in the analysis as well as the precent
% deviation threshold.
archive_dir='/home/alex/Projects/job_archive';
source_dir=[archive_dir '/Clinac6X'];

% Set the parameter ranges for the analysis.  The list of field sizes (FS)
% will determine which set of field sizes are used in the CF and threshold
% analysis.  The brem. list (BS_list) defines which combinations will be
% compared.  Each brem. combination will be plotted accross the energy (E),
% intensity dist. FWHM (W), and divergence parametery (D).
FS_1=[3 20 40];
FS_2=[3 10 20 30 40];
BS=regexprep(BS_name,'/','');
E=round((5.85:0.01:5.95)*1000)/1000;
W=round((0.0:0.05:0.3)*1000)/1000;
D=round((0.6:0.1:1.2)*1000)/1000;

% Name the save file.
save_file=['/home/alex/Projects/MATLAB/Beam Model/' BS '_analyze_data/'...
    M_data '.' num2str(plim,'%0.2f') '.mat'];

% Analyze the requested parameterizations.  If recalc=1 then the entire
% available set of jobs is analyzed, otherwise only the un-analyzed
% jobs are analyzed.
[new_job_data]=analyze_all(source_dir,recalc,{BS},FS_2,E,W,D,plim,...
    dlim,smooth,x2,M_data);

% If a save file exist open it.
```

```
if exist(save_file,'file')
    load(save_file)
end

% If re-analyzing, or if no job_data cell exists yet...
if recalc==1 || ~exist('job_data','var')
    job_data=new_job_data;

    % Else concatenate the new data onto the old...
else
    job_data=[job_data;new_job_data];
end

% Save job_data for recalc=0 runs.
save(save_file,'job_data')

% Call OF_calc to calculate OF and OF_X2 for each FS set.
[OF_1,OF_X2_1,unc_1]=OF_calc(job_data,BS,FS_1,E,W,D,ROI_w,PN_w,X2_w);
[OF_2,OF_X2_2,unc_2]=OF_calc(job_data,BS,FS_2,E,W,D,ROI_w,PN_w,X2_w);

% Locate the grid indices for the plottable parameterizations, the
% parameterizations that completely meet the specified accuracy
% threshold, i.e. OF=1, and the parameterizations for which OF_X2
% meets the specified threshold.
[I_1,J_1,K_1]=ind2sub(size(OF_1),find(OF_1~=0));
[I_2,J_2,K_2]=ind2sub(size(OF_2),find(OF_2~=0));
[I_pass_1,J_pass_1,K_pass_1]=ind2sub(size(OF_1),find(OF_1==1));
[I_pass_2,J_pass_2,K_pass_2]=ind2sub(size(OF_2),find(OF_2==1));
[I_thr_1,J_thr_1,K_thr_1]=ind2sub(size(OF_X2_1),find(OF_X2_1>=OF_X2_thresh));
[I_thr_2,J_thr_2,K_thr_2]=ind2sub(size(OF_X2_2),find(OF_X2_2>=OF_X2_thresh));

% Extract the plottable OF_X2 and max_unc values.
OF_X2_plot_1=OF_X2_1(OF_X2_1~=0);
OF_X2_plot_2=OF_X2_1(OF_X2_2~=0);
unc_plot_1=unc_1(OF_1~=0);
unc_plot_2=unc_2(OF_2~=0);
```

analyze_all.m

```
% This script analyzes all the jobs in the job analysis archive. If
% recalc=1, every job within the specified range is analyzed. If recalc=0
% then only recently added jobs (with not .stat file yet) are analyzed.

function [job_data]=analyze_all(archive_dir,recalc,BS,FS,E,W,D,plim,dlim,smooth,x2,M_data)

% Build a list of requested parameterizations.
requested_pars=cell(numel(BS)*numel(FS)*numel(E)*numel(W)*numel(D),1);
n=1;
for b=1:numel(BS)
    for f=1:numel(FS)
        for e=1:numel(E)
            for w=1:numel(W)
                for d=1:numel(D)

                    % Get char versions of the parameters.
                    E_char=num2str(E(e),'%0.2f');
                    W_char=num2str(W(w),'%0.2f');
                    D_char=num2str(D(d),'%0.1f');

                    % Find the simulation directory and rsync the
                    % simulation data to the analysis archive.
                    requested_pars{n}=[BS{b} '_' num2str(FS(f)) '_' E_char '_' W_char '_'
D_char];
                    n=n+1;
                end
            end
        end
    end
end

% Build a list of simulated parameterizations and then find the
% intersections of this list with the requested_pars.
[~,dose_files]=system(['find ' archive_dir ' -name "*.3ddose"']);
dose_files=textscan(dose_files,'%s','Delimiter','\n');
simulated_pars=regexprep(dose_files{1},'.*/\w*_(\w*_\d*_[\d.]*_[\d.]*_[\d.]*)\.3ddose','$1');
job_pars=intersect(requested_pars,simulated_pars);

% If not re-analyzing all the jobs, find a list of parameterizations that
% are available and have also not yet been analyzed.
if recalc==0

    % Build a list of already analyzed parameterizations.  This is done by
    % searching for .stat files which are outputs of the analysis process.
    % Then find the jobs
    [~,stat_files]=system(['find ' archive_dir ' -name "*' M_data '_' num2str(plim,'%0.3f')
'.stat"']);
    stat_files=textscan(stat_files,'%s','Delimiter','\n');
    analyzed_pars=regexprep(stat_files{1},...
        '.*/\w*_(\w*_\d*_[\d.]*_[\d.]*_[\d.]*)_\w*_[\d.]*\.stat','$1');

    % Find the parameterizations that are not in the intersection of the
    % analyze list and the analyzed list.  This must be done in this order
    % to prevent analyzing un-analyzed jobs that have not been requested.
    job_pars=setxor(job_pars,intersect(job_pars,analyzed_pars));
end

% Find the actual list of .3ddose files that need to be analyzed.
J=numel(job_pars);
job_data=cell(J,1);
for j=1:numel(job_pars)
    [~,job_file]=system(['find ' archive_dir ' -name "*' job_pars{j} '.3ddose"']);
    job_name=regexprep(job_file,'.*/(.*)$','$1');
    ['Analyzing ' job_name]
    ['Job ' num2str(j) ' of ' num2str(J) '.']
    job_data{j}=analyze(strtrim(job_file),plim,dlim,1,smooth,x2,M_data);
end
```

OF_calc.m

```
% This function calcl utes the two objective functions.

function [OF,OF_X2,max_unc]=OF_calc(job_data,BS,FS,E,W,D,ROI_w,PN_w,X2_w)

% Define the arrays that will accept the various accuracy metrics.  FS_bin
% is 4D array used to check if a given parameterization is available for a
% given field size.  OF and OF_X2 are objective funtions which are used to
% plot the paramterization trends.  In addition, when OF(FS,E,W,D)=1, that
% parameterization "passes" the specified accuracy threshold.
FS_bin=zeros(numel(E),numel(W),numel(D),numel(BS),numel(FS));
OF=zeros(numel(E),numel(W),numel(D),numel(BS),numel(FS));
OF_X2=zeros(numel(E),numel(W),numel(D),numel(BS),numel(FS));
max_unc=zeros(numel(E),numel(W),numel(D),numel(BS),numel(FS));

% Build binary array of where parameterizations exist.
for n=1:numel(job_data)
    FS_bin(...
        E==str2double(job_data{n}{3}),...
        W==str2double(job_data{n}{4}),...
        D==str2double(job_data{n}{5}),...
        strcmp(BS,job_data{n}{2}),...
        FS==str2double(job_data{n}{1}))=1;
end

% Extract analysis data and calulate the objective functions.  This
% loop searches each job element of the job_data cell for a match
% of the current interation's set of parameters.  When a match is
% found the passing fraction data is extracted and used to
% calculate OF and OF_X2.
for n=1:numel(job_data)
    BSi=find(strcmp(BS,job_data{n}{2}));
    FSi=find(FS==str2double(job_data{n}{1}));
    Ei=find(E==str2double(job_data{n}{3}));
    Wi=find(W==str2double(job_data{n}{4}));
    Di=find(D==str2double(job_data{n}{5}));
    ROI_f=mean([job_data{n}{11}/job_data{n}{12} job_data{n}{13}./job_data{n}{14}]);
    PN_f=mean(job_data{n}{15}./job_data{n}{16});
    X2=mean(job_data{n}{8});
    if sum(FS_bin(Ei,Wi,Di,BSi,:))>=numel(FS)
        OF_X2(Ei,Wi,Di,BSi,FSi)=(ROI_w*ROI_f+PN_w*PN_f+X2_w/X2)/(ROI_w+PN_w+X2_w);
        OF(Ei,Wi,Di,BSi,FSi)=(ROI_w*ROI_f+PN_w*PN_f)/(ROI_w+PN_w);
        max_unc(Ei,Wi,Di,BSi,FSi)=max(job_data{n}{7});
    end
end

% Average the arrays accross the field size dimension.
OF=mean(OF,5);
OF_X2=mean(OF_X2,5);

% Find the max uncertainty accross the field size dimension.
max_unc=max(max_unc,[],5);
```

Appendix B MLC Model Validation Code

MLC Geometry Verification Script

MLC_dimensions.m

```
% This function calculates the BEAM MLC dimensions based on the Varian
% Monte Carlo Data Package specifications.

function [MLC_dims,MLC_thickness,MLC_z_start,MLC_y_start,leaf_gap]...
    =MLC_dimensions(MLC_z_center)

% Varian MLC dimensions as listed in the MCDP. Triplets
% are in the order [full,target,isocenter]. Note that dashes are
% placeholders for proprietary specifications that cannot be listed.
wl_var=[- - -];
wt_var=[- - -];
wg_var=wt_var;
wtip_var=[- - -];
wts_var=[- - -];
wbs_var=[- - -];
ztip_var=[- - -];
zl_var=[- - -];
zt_var=[- - -];
zg_var=[- - -];
zts_var=[- - -];
zbs_var=[- - -];
zth_var=[- - -];
zbh_var=[- - -];
hole_pos_var=-;
leaf_gap_var=-;

% Calculate leaf offsets in the z direction.  The center_offset equation
% was developed by using a ruler as no actual spec was provided.
zl_center=(zl_var(2)+ztip_var(1)-zts_var(1))/2;
MLC_z_start=MLC_z_center-zl_center;
zl_offset=zl_center-[ztip_var(1)-zl_center zl_center ztip_var(3)-zl_center];
zl_0=MLC_z_start+zl_offset;


% Calculated widths at isocenter. Leaf width calculations use dimensions
% defined at different z's so they are more complex.
r=(zl_0+[ztip_var(1) zt_var(2) zg_var(3)])./(zl_0+[zts_var(1) 0 0]);
d=wl_var.*(r-1)./(r+1);
wl_var_iso=(wl_var-d)./(zl_0+[zts_var(1) 0 0])*100;
wt_var_iso=wt_var./(zl_0+zt_var)*100;
wg_var_iso=wg_var./(zl_0+zg_var)*100;
wtip_var_iso=wtip_var./(zl_0+[0 ztip_var(2) 0])*100;
wts_var_iso=wts_var./(zl_0+zts_var)*100;
wbs_var_iso=wbs_var./(zl_0+zbs_var)*100;
leaf_gap_iso=leaf_gap_var/MLC_z_center*100;
leaf_gap=leaf_gap_iso*MLC_z_start/100;

% The above values produce projections at iso that are too wide.  Either the
% leaf dimensions are wrong, or the z location of the MLC is wrong.  We now
% calculate leaf width dimensions based on a chosen leaf gap and 0.5 and
% 1.0 cm width leaf projections at isocenter.
wn_iso=[1.0 0.5 0.5];
wl_iso=wn_iso-leaf_gap_iso;
wt_iso=wt_var./MLC_z_center*100;
```

```
wg_iso=wt_iso;

% Build MLC dimensions array.
widths=[wl_iso' wt_iso' wg_iso' wtip_var_iso' wts_var_iso' wbs_var_iso']*...
    MLC_z_start/100;
ztip=[zl_offset(1) ztip_var(2) zl_offset(3)]+MLC_z_start;
zl=zl_offset+zl_var+MLC_z_start;
zt=zl_offset+zt_var+MLC_z_start;
zg=zl_offset+zg_var+MLC_z_start;
zth=zl_offset+zth_var+MLC_z_start;
zbh=zl_offset+zbh_var+MLC_z_start;
zts=zl_offset+zts_var+MLC_z_start;
zbs=zl_offset+zbs_var+MLC_z_start;

full_leaf_dims=[widths(1,:) ztip(1) zl(1) zt(1) zg(1) zth(1) zbh(1)...
    hole_pos_var zts(1) zbs(1)];
target_leaf_dims=[widths(2,:) zts(2) zbs(2) zth(2) zbh(2) hole_pos_var zt(2)...
    zg(2) zl(2) ztip(2)];
isocenter_leaf_dims=[widths(3,:) ztip(3) zl(3) zt(3) zg(3) zth(3) zbh(3)...
    hole_pos_var zts(3) zbs(3)];
MLC_dims=[full_leaf_dims; target_leaf_dims; isocenter_leaf_dims];

% Calculate MLC thickness.
MLC_thickness=zl_offset(3)+ztip_var(3);

% Calculate MLC y start position.
overlap_iso=(wt_iso(1)-leaf_gap_iso)/2;
overlap=overlap_iso*MLC_z_start/100;
MLC_y_start=-20*MLC_z_start/100-overlap;
Profile and Film Analysis Script: profile_analyze.m
% This script compares MC, film, AAA, and WT profiles.

% Select the patient, plan, beam, and depth to analyze.
plan_num=1;
patient='MLC_val';
plan='MLC_val_1';
beam='gaps';
d_plane=10;
phantom='wat_phant_2d';

close all

% Set to one to update dose arrays, otherwise load them from file.
update=0;

% Define analysis parameters.
D=[1.5 5 10 15]'; % Available plane depths for MC, WT, and AAA.
ebar=0; % Set this to one to view error bars for MC and film.
reg=1; % Set this to one to register film to MC.
reg2=1; % Set this to one to further optimize film registration (takes some time).
MC_smooth=2; % Set this to one to Savitzey-Golay filter the MC profiles.
MC_X2_lim=2; % X^2 limit for SG smoothing of the film.
film_smooth=1; % Set this to one to Savitzey-Golay filter the film profiles.
film_sigma=0.01; % Estimate a film random error for smoothing.
film_X2_lim=1; % X^2 limit for SG smoothing of the film.
chan=1; % Set this the film scan color index to view (RGB).
view=2; % Set this to one to view the film to MC registration.
PD_view=0;
WT_scan=0; % Set this to one to include water tank scans (only z scans exist at x=0).
screen='laptop';
dev_lim=10; % (%)
zprof_x=1; % x axis value for the y profile.
xprof_z=7.5; % y axis value for the x profile.
vox=0.1; % Set the resolution for the final grid (cm).
publish=1; % Set to one to print a publishable figure to file.
```

```
% Define some constant names and directories.
linac='Clinac6X';
patient_dir=['/home/alex/Projects/Plan Recalculation/Patients/' patient];
film_dir=['/home/alex/Projects/MLC Validation/Film/' beam];
WT_dir='/home/alex/Projects/MLC Validation/WT_scans';
save_file=['/home/alex/Projects/MATLAB/MLC Validation/profile_analyze_data/'...
    'profile_analyze.' beam '.' num2str(d_plane*10) 'mm.mat'];

% calibration constants (adjust per source parameterization)
model_factor=0.98; % AAA/MC
chamb_dpp_cal=9.785e-16; % from 10x10 calibration run (sigma = 0.2%)
phantom_dpp_cal=1.1594e-16; % from 10x10 water phantom run (sigma = 0.17%)
MU_cal=0.01; % (Gy/MU)

% Update dose arrays, otherwise load them from file.
if update==1

    %------------EXTRACT DOSE ARRAYS------------------------------------

    % Extract MC dose and interpolate onto selected depths and the ROI
    % grid, which will be defined as the MC grid limited by a set margin
    % around the deepest jaw projection.
    MC_file=[patient_dir '/' phantom '.' patient '.' plan '.' beam '.3ddose'];
    [MC_3d_dpp,MC_3d_unc,MC_X,MC_Y,MC_Z]=MC_read(MC_file);
    RP_file=[patient_dir '/RP.' patient '.' plan '.dcm'];
    [beam_lab,JP,~,~,~,MU,~,~]=plan_pars(RP_file);
    b_sel=strcmp(beam_lab(2,:),beam);

    div_factor=max(d_plane)/100+1;
    margin=0.5;
    x_lims=div_factor*JP{b_sel}(1,:);
    z_lims=div_factor*JP{b_sel}(2,:);
    X=MC_X(MC_X>=x_lims(1)-margin and MC_X<=x_lims(2)+margin);
    Z=MC_Z(MC_Z>=z_lims(1)-margin and MC_Z<=z_lims(2)+margin);
    MC_3d_dpp=interp3(MC_X,MC_Y,MC_Z,MC_3d_dpp,X',D,Z);
    MC_3d_unc=interp3(MC_X,MC_Y,MC_Z,MC_3d_unc,X',D,Z);

    % Calibrate the MC dose.
    cal_file=[patient_dir '/' linac '.' patient '.' plan '.' beam '.egslst'];
    cal_lines=textscan(fopen(cal_file),'%s','Delimiter','\n');
    rec_1=find(~cellfun(@isempty,strfind(cal_lines{1},'DOSE RESULTS')),1);
    chamb_dpp_plan=sscanf(cal_lines{1}{rec_1+18},'%*d %*f %f+/- %*f %% %*f+/- %*f %%');
    N_part=chamb_dpp_cal/chamb_dpp_plan*MU_cal/phantom_dpp_cal*MU{1,1}*model_factor;
    MC_3d_dose=MC_3d_dpp*N_part;
    MC_dose=squeeze(MC_3d_dose(D==d_plane,:,:))';
    MC_unc=squeeze(MC_3d_unc(D==d_plane,:,:))';

    % Extract AAA dose, and interplate onto selected profiles.
    RD_file=[patient_dir '/RD.' patient '.' plan '.' beam '.dcm'];
    RD_data=dicominfo(RD_file);
    IPP=RD_data.ImagePositionPatient;
    AAA_vox=[RD_data.PixelSpacing(2) RD_data.PixelSpacing(1)];
    AAA_X=((0:double(RD_data.Columns)-1)'*AAA_vox(1)+IPP(1))/10;
    AAA_Y=((0:double(RD_data.Rows)-1)'*AAA_vox(2)+IPP(2))/10;
    AAA_Z=double(IPP(3)+RD_data.GridFrameOffsetVector)/10;
    AAA_vox=[AAA_vox AAA_Z(2)-AAA_Z(1)];
    AAA_3d_dose=double(squeeze(dicomread(RD_file)))*RD_data.DoseGridScaling;
    AAA_dose=squeeze(interp3(AAA_X,AAA_Y,AAA_Z,AAA_3d_dose,...
        AAA_X',D(D==d_plane),AAA_Z))';

    % Execute film_register to extract and register the film dose
    % from the .tif files. Then extract the selected dose channel.
    film_dose_rgb=film_register(film_dir,beam,X,Z,d_plane,reg2,MC_dose);
    film_dose=film_dose_rgb(:,:,chan);

    % Save arrays so the film registraton need not be repeated.
    save(save_file,'film_dose','MC_dose','MC_unc','AAA_dose',...
        'X','Z','AAA_X','AAA_Z');
else
    load(save_file)
end

%-----------------VIEW COMPARISONS-----------------------------------

% If a 2d plane image comarison is requested.
if view>0

    % Compare AAA to MC.
    if view==1
        % Map AAA dose onto the film grid.
        comp_dose=interp2(AAA_X,AAA_Z,AAA_dose,X',Z);
        comp_name='AAA';

        % Comapre film to MC.
    elseif view==2
        comp_dose=film_dose;
        comp_name='Film';
    end

    % Set comparison labels and build the image comparison.
    names={'MC',comp_name,...
        ['Plane comparison for ' patient '.' plan '.'...
        beam ' at ' num2str(d_plane) ' cm depth.']};
    plane_compare(MC_dose,comp_dose,MC_unc,X,Z,names,screen,xprof_z,zprof_x)
end

% Extract film profiles.
film_xprof_dose=interp2(X,Z,film_dose,X,xprof_z)';
film_zprof_dose=interp2(X,Z,film_dose,zprof_x,Z);

% Build film uncertainy arrays.
film_xprof_unc=ones(numel(film_xprof_dose),1)*film_sigma;
film_zprof_unc=ones(numel(film_zprof_dose),1)*film_sigma;

% Extract MC profiles.
MC_xprof_dose=interp2(X,Z,MC_dose,X,xprof_z)';
MC_xprof_unc=interp2(X,Z,MC_unc,X,xprof_z)';
MC_zprof_dose=interp2(X,Z,MC_dose,zprof_x,Z);
MC_zprof_unc=interp2(X,Z,MC_unc,zprof_x,Z);
if MC_smooth==1
    [MC_xprof_dose,MC_xprof_unc]=SG_smooth(MC_xprof_dose,MC_xprof_unc,50,MC_X2_lim);
    [MC_zprof_dose,MC_zprof_unc]=SG_smooth(MC_zprof_dose,MC_zprof_unc,50,MC_X2_lim);
end
if film_smooth==1
    [film_xprof_dose,film_xprof_unc]=...
        SG_smooth(film_xprof_dose,film_xprof_unc,20,film_X2_lim);
    [film_zprof_dose,film_zprof_unc]=...
        SG_smooth(film_zprof_dose,film_zprof_unc,20,film_X2_lim);
end

% Extract AA profiles.
AAA_xprof_dose=interp2(AAA_X,AAA_Z,AAA_dose,AAA_X,xprof_z);
AAA_xprof_unc=interp2(AAA_X,AAA_Z,AAA_dose,AAA_X,xprof_z);
AAA_zprof_dose=interp2(AAA_X,AAA_Z,AAA_dose,zprof_x,AAA_Z);
AAA_zprof_unc=interp2(AAA_X,AAA_Z,AAA_dose,zprof_x,AAA_Z);

% Extract WT scan profiles.
if WT_scan==1 andand zprof_x==0

    % Read in the selected WT profile.
    Y_jaw_width=JP{b_sel}(2,2)-JP{b_sel}(2,1);
    [WT_zprof_norm,WT_Z]=WT_read([WT_dir '/' plan '.mcc'],Y_jaw_width,...
        d_plane,'INPLANE');


    % Calculate profile widths and centers, then register the water scan
    % profile to the AAA profile as it will have the best alignment.
    level=[0 0.5 0.5 0.35 0.8];
    AAA_50_Z=edges(AAA_zprof_dose,AAA_Z,level(plan_num));
    trans_srch=-0.1:0.02:0.1; % cm
    OF=zeros(numel(trans_srch),1);
    for t=1:numel(trans_srch)
        WT_50_Z=edges(WT_zprof_norm,WT_Z-trans_srch(t),level(plan_num));
        OF(t)=sum((AAA_50_Z-WT_50_Z).^2);
    end
    shift=trans_srch(OF==min(OF));
    WT_zprof_norm=interp1(WT_Z-shift,WT_zprof_norm,WT_Z,'linear',0);

    % Fit a 2nd order polynomial to a region in both the WT and film
    % profiles that is assumed to be accurate.  Use a central point of
    % these fits to scale the WT dose.
    fit_range=div_factor*[0 0; 6 9; 3 7; 6 9; 3 7];
    fit_ind=(Z>=fit_range(plan_num,1) and Z<=fit_range(plan_num,2));
    WT_scaling=polyval(polyfit(Z(fit_ind),film_zprof_dose(fit_ind),2),...
        mean(fit_range(plan_num,:)))/...
        polyval(polyfit(Z(fit_ind),WT_zprof_norm(fit_ind),2),...
        mean(fit_range(plan_num,:)));
    WT_zprof_dose=WT_zprof_norm*WT_scaling;

    % Calculate WT deviation.
    WT_dev=(interp1(WT_Z,WT_zprof_dose,Z)./film_zprof_dose-1)*100;
end
% This script compares MC, film, AAA, and WT profiles.
```

MLC Validation Analysis Scripts

profile_analyize.m

```
% Select the patient, plan, beam, and depth to analyze.
plan_num=1;
patient='MLC_val';
plan='MLC_val_1';
beam='gaps';
d_plane=10;
phantom='wat_phant_2d';


% Use this loop setup tp run all available film planes.
beam_list={'gaps','leaves'}%,'squares','ends'};
for b=1:4
beam=beam_list{b};
for d_plane=[1.5 5 10]


close all


% Set to one to update dose arrays, otherwise load them from file.
update=0;


% Define analysis parameters.
D=[1.5 5 10 15]'; % Available plane depths for MC, WT, and AAA.
ebar=0; % Set this to one to view error bars for MC and film.
reg=1; % Set this to one to register film to MC.
reg2=1; % Set this to one to further optimize film registration (takes some time).
MC_smooth=2; % Set this to one to Savitzey-Golay filter the MC profiles.
```

```
MC_X2_lim=2; % X^2 limit for SG smoothing of the film.
film_smooth=1; % Set this to one to Savitzey-Golay filter the film profiles.
film_sigma=0.01; % Estimate a film random error for smoothing.
film_X2_lim=1; % X^2 limit for SG smoothing of the film.
chan=1; % Set this the film scan color index to view (RGB).
view=2; % Set this to one to view the film to MC registration.
PD_view=0;
WT_scan=0; % Set this to one to include water tank scans (only z scans exist at x=0).
screen='laptop';
dev_lim=10; % (%)
zprof_x=1; % x axis value for the y profile.
xprof_z=7.5; % y axis value for the x profile.
vox=0.1; % Set the resolution for the final grid (cm).
publish=1; % Set to one to print a publishable figure to file.

% Define some constant names and directories.
linac='Clinac6X';
patient_dir=['/home/alex/Projects/Plan Recalculation/Patients/' patient];
film_dir=['/home/alex/Projects/MLC Validation/Film/' beam];
WT_dir='/home/alex/Projects/MLC Validation/WT_scans';
save_file=['/home/alex/Projects/MATLAB/MLC Validation/profile_analyze_data/'...
    'profile_analyze.' beam '.' num2str(d_plane*10) 'mm.mat'];

% calibration constants (adjust per source parameterization)
model_factor=0.98; % AAA/MC
chamb_dpp_cal=9.785e-16; % from 10x10 calibration run (sigma = 0.2%)
phantom_dpp_cal=1.1594e-16; % from 10x10 water phantom run (sigma = 0.17%)
MU_cal=0.01; % (Gy/MU)

% Update dose arrays, otherwise load them from file.
if update==1

    %------------EXTRACT DOSE ARRAYS------------------------------------

    % Extract MC dose and interpolate onto selected depths and the ROI
    % grid, which will be defined as the MC grid limited by a set margin
    % around the deepest jaw projection.
    MC_file=[patient_dir '/' phantom '.' patient '.' plan '.' beam '.3ddose'];
    [MC_3d_dpp,MC_3d_unc,MC_X,MC_Y,MC_Z]=MC_read(MC_file);
    RP_file=[patient_dir '/RP.' patient '.' plan '.dcm'];
    [beam_lab,JP,~,~,~,MU,~,~]=plan_pars(RP_file);
    b_sel=strcmp(beam_lab(2,:),beam);

    div_factor=max(d_plane)/100+1;
    margin=0.5;
    x_lims=div_factor*JP{b_sel}(1,:);
    z_lims=div_factor*JP{b_sel}(2,:);
    X=MC_X(MC_X>=x_lims(1)-margin and MC_X<=x_lims(2)+margin);
    Z=MC_Z(MC_Z>=z_lims(1)-margin and MC_Z<=z_lims(2)+margin);
    MC_3d_dpp=interp3(MC_X,MC_Y,MC_Z,MC_3d_dpp,X',D,Z);
    MC_3d_unc=interp3(MC_X,MC_Y,MC_Z,MC_3d_unc,X',D,Z);

    % Calibrate the MC dose.
    cal_file=[patient_dir '/' linac '.' patient '.' plan '.' beam '.egslst'];
    cal_lines=textscan(fopen(cal_file),'%s','Delimiter','\n');
    rec_1=find(~cellfun(@isempty,strfind(cal_lines{1},'DOSE RESULTS')),1);
    chamb_dpp_plan=sscanf(cal_lines{1}{rec_1+18},'%*d %*f %f+/- %*f %% %*f+/- %*f %%');
    N_part=chamb_dpp_cal/chamb_dpp_plan*MU_cal/phantom_dpp_cal*MU{1,1}*model_factor;
    MC_3d_dose=MC_3d_dpp*N_part;
    MC_dose=squeeze(MC_3d_dose(D==d_plane,:,:))';
    MC_unc=squeeze(MC_3d_unc(D==d_plane,:,:))';
    % Extract AAA dose, and interplate onto selected profiles.
    RD_file=[patient_dir '/RD.' patient '.' plan '.' beam '.dcm'];
    RD_data=dicominfo(RD_file);
    IPP=RD_data.ImagePositionPatient;
    AAA_vox=[RD_data.PixelSpacing(2) RD_data.PixelSpacing(1)];
    AAA_X=((0:double(RD_data.Columns)-1)'*AAA_vox(1)+IPP(1))/10;
    AAA_Y=((0:double(RD_data.Rows)-1)'*AAA_vox(2)+IPP(2))/10;
    AAA_Z=double(IPP(3)+RD_data.GridFrameOffsetVector)/10;
    AAA_vox=[AAA_vox AAA_Z(2)-AAA_Z(1)];
    AAA_3d_dose=double(squeeze(dicomread(RD_file)))*RD_data.DoseGridScaling;
    AAA_dose=squeeze(interp3(AAA_X,AAA_Y,AAA_Z,AAA_3d_dose,...
        AAA_X',D(D==d_plane),AAA_Z))';

    % Execute film_register to extract and register the film dose
    % from the .tif files. Then extract the selected dose channel.
    film_dose_rgb=film_register(film_dir,beam,X,Z,d_plane,reg2,MC_dose);
    film_dose=film_dose_rgb(:,:,chan);

    % Save arrays so the film registraton need not be repeated.
    save(save_file,'film_dose','MC_dose','MC_unc','AAA_dose',...
        'X','Z','AAA_X','AAA_Z');
else
    load(save_file)
end

%-----------------VIEW COMPARISONS-----------------------------------

% If a 2d plane image comarison is requested.
if view>0

    % Compare AAA to MC.
    if view==1
        % Map AAA dose onto the film grid.
        comp_dose=interp2(AAA_X,AAA_Z,AAA_dose,X',Z);
        comp_name='AAA';

        % Comapre film to MC.
    elseif view==2
        comp_dose=film_dose;
        comp_name='Film';
    end

    % Set comparison labels and build the image comparison.
    names={'MC',comp_name,...
        ['Plane comparison for ' patient '.' plan '.'...
        beam ' at ' num2str(d_plane) ' cm depth.']};
    plane_compare(MC_dose,comp_dose,MC_unc,X,Z,names,screen,xprof_z,zprof_x)
end

% Extract film profiles.
film_xprof_dose=interp2(X,Z,film_dose,X,xprof_z)';
film_zprof_dose=interp2(X,Z,film_dose,zprof_x,Z);

% Build film uncertainy arrays.
film_xprof_unc=ones(numel(film_xprof_dose),1)*film_sigma;
film_zprof_unc=ones(numel(film_zprof_dose),1)*film_sigma;

% Extract MC profiles.
MC_xprof_dose=interp2(X,Z,MC_dose,X,xprof_z)';
MC_xprof_unc=interp2(X,Z,MC_unc,X,xprof_z)';
MC_zprof_dose=interp2(X,Z,MC_dose,zprof_x,Z);
```

```
MC_zprof_unc=interp2(X,Z,MC_unc,zprof_x,Z);
if MC_smooth==1
    [MC_xprof_dose,MC_xprof_unc]=SG_smooth(MC_xprof_dose,MC_xprof_unc,50,MC_X2_lim);
    [MC_zprof_dose,MC_zprof_unc]=SG_smooth(MC_zprof_dose,MC_zprof_unc,50,MC_X2_lim);
end
if film_smooth==1
    [film_xprof_dose,film_xprof_unc]=...
        SG_smooth(film_xprof_dose,film_xprof_unc,20,film_X2_lim);
    [film_zprof_dose,film_zprof_unc]=...
        SG_smooth(film_zprof_dose,film_zprof_unc,20,film_X2_lim);
end

% Extract AA profiles.
AAA_xprof_dose=interp2(AAA_X,AAA_Z,AAA_dose,AAA_X,xprof_z);
AAA_xprof_unc=interp2(AAA_X,AAA_Z,AAA_dose,AAA_X,xprof_z);
AAA_zprof_dose=interp2(AAA_X,AAA_Z,AAA_dose,zprof_x,AAA_Z);
AAA_zprof_unc=interp2(AAA_X,AAA_Z,AAA_dose,zprof_x,AAA_Z);

% Extract WT scan profiles.
if WT_scan==1 andand zprof_x==0

    % Read in the selected WT profile.
    Y_jaw_width=JP{b_sel}(2,2)-JP{b_sel}(2,1);
    [WT_zprof_norm,WT_Z]=WT_read([WT_dir '/' plan '.mcc'],Y_jaw_width,...
        d_plane,'INPLANE');



    % Calculate profile widths and centers, then register the water scan
    % profile to the AAA profile as it will have the best alignment.
    level=[0 0.5 0.5 0.35 0.8];
    AAA_50_Z=edges(AAA_zprof_dose,AAA_Z,level(plan_num));
    trans_srch=-0.1:0.02:0.1; % cm
    OF=zeros(numel(trans_srch),1);
    for t=1:numel(trans_srch)
        WT_50_Z=edges(WT_zprof_norm,WT_Z-trans_srch(t),level(plan_num));
        OF(t)=sum((AAA_50_Z-WT_50_Z).^2);
    end
    shift=trans_srch(OF==min(OF));
    WT_zprof_norm=interp1(WT_Z-shift,WT_zprof_norm,WT_Z,'linear',0);

    % Fit a 2nd order polynomial to a region in both the WT and film
    % profiles that is assumed to be accurate.  Use a central point of
    % these fits to scale the WT dose.
    fit_range=div_factor*[0 0; 6 9; 3 7; 6 9; 3 7];
    fit_ind=(Z>=fit_range(plan_num,1) and Z<=fit_range(plan_num,2));
    WT_scaling=polyval(polyfit(Z(fit_ind),film_zprof_dose(fit_ind),2),...
        mean(fit_range(plan_num,:)))/...
        polyval(polyfit(Z(fit_ind),WT_zprof_norm(fit_ind),2),...
        mean(fit_range(plan_num,:)));
    WT_zprof_dose=WT_zprof_norm*WT_scaling;

    % Calculate WT deviation.
    WT_dev=(interp1(WT_Z,WT_zprof_dose,Z)./film_zprof_dose-1)*100;
end

% Build plotting cells and calculate deviations.
MC_prof={MC_xprof_dose,MC_zprof_dose};
MC_unc={MC_xprof_unc,MC_zprof_unc};
MC_dev={(MC_xprof_dose./film_xprof_dose-1)*100,...
    (MC_zprof_dose./film_zprof_dose-1)*100};
AAA_prof={AAA_xprof_dose,AAA_zprof_dose};
AAA_dev={(interp1(AAA_X,AAA_xprof_dose,X)./film_xprof_dose-1)*100,...
```

```
    (interp1(AAA_Z,AAA_zprof_dose,Z)./film_zprof_dose-1)*100};
AAA_coords={AAA_X,AAA_Z};
coords={X,Z};

% Plot
if strcmp(screen,'OHSU')
    fig_size=[1 0.5 12 10];
elseif strcmp(screen,'home')
    fig_size=[1 0.5 16 11];
elseif strcmp(screen,'laptop')
    fig_size=[0.1 0.1 12 8];
end

% Plot a publishable figure
if publish==1

    pub_fig=figure('Units','inches','OuterPosition',[0 0 8 8]);
    dose_lims=[0 1.1*max([max(film_xprof_dose) max(film_zprof_dose)])];
    y_lims=dose_lims*1.1;

    % X profile on the left pane
    axes('Position',[0.1 0.1 0.3 0.8]);
    x_lims=[X(1) X(end)];
    plot(AAA_X,AAA_xprof_dose,'g');
    hold on
    plot(X,MC_xprof_dose,'-r');
    plot(X,film_xprof_dose,'m');
    hold off
    set(gca,'XLim',x_lims,'YLim',y_lims)
    xlabel('x coordinate')
    ylabel('Dose (Gy)')
    text(0.94,0.95,['z = ' num2str(xprof_z) ' cm'],...
        'Units','normalized','HorizontalAlignment','right')
    legend({'AAA dose','Film dose','MC dose','WT dose'},'Location','northwest')

    % X profile on the left pane
    axes('Position',[0.4 0.1 0.5 0.8]);
    x_lims=[Z(1) Z(end)];
    plot(AAA_Z,AAA_zprof_dose,'g');
    hold on
    plot(Z,MC_zprof_dose,'-r');
    plot(Z,film_zprof_dose,'m');
    hold off
    set(gca,'XLim',z_lims,'YLim',y_lims)
    xlabel('z coordinate (cm)')
    set(gca,'YTickLabel','')
    text(0.94,0.95,['x = ' num2str(zprof_x) ' cm'],...
        'Units','normalized','HorizontalAlignment','right')

    % Print the figure to file.
    print_file=['/home/alex/Projects/Documents/Dissertation/figs/'...
        beam '.' num2str(d_plane*10) 'mm.prof'];
    export_fig(print_file,'-eps','-pdf','-nocrop',pub_fig)
end
```

film_register.m

```
% This script uses fiducial isocenter marks in the film image to find the
% isocenter, then shift and rotate the image to match the MC image.
```

```
function [film_dose]=film_register(film_dir,beam,X,Y,d_plane,reg2,MC_dose)

% Extract image arrays and header info from the .tif files.
d_char_mm=num2str(d_plane*10);
film_img=flipdim(imread([film_dir '/' beam '.' d_char_mm 'mm.dose.crop.tif']),1);
film_info=imfinfo([film_dir '/' beam '.' d_char_mm 'mm.dose.tif']);

% Build the film grid.
scan_res=2.54./[film_info.XResolution film_info.YResolution];
xlims=[0 numel(film_img(1,:,1))*scan_res(1)];
ylims=[0 numel(film_img(:,1,1))*scan_res(2)];
X_img=(xlims(1)+scan_res(1)/2:scan_res(1):xlims(end)-scan_res/2);
Y_img=(ylims(1)+scan_res(2)/2:scan_res(2):ylims(end)-scan_res/2);

% The projection of the crosshairs onto to the edge of the film are
% marked with marker so that the isocenter can be found for
% registration.  Here the search region for these fisucials is defined
% as a line of points with a certain length and offset from each edge
% of the film.
srch_offset=0.2; % in cm from edge of film
srch_len=2.0; % cm in each direction from center of film
lft_srch_j=find(X_img>srch_offset and X_img<srch_offset+scan_res(1));
rt_srch_j=find(X_img<xlims(2)-srch_offset and X_img>xlims(2)-srch_offset-scan_res(1));
top_srch_i=find(Y_img>srch_offset and Y_img<srch_offset+scan_res(2));
bot_srch_i=find(Y_img<ylims(2)-srch_offset and Y_img>ylims(2)-srch_offset-scan_res(2));
srch_J=round(numel(X_img)/2)+(-round(srch_len/scan_res(2)):round(srch_len/scan_res(2)));
srch_I=round(numel(Y_img)/2)+(-round(srch_len/scan_res(1)):round(srch_len/scan_res(1)));

% The line along the center of each edge of the film is searched for
% the max value.  These points are used to generate two "perpendicular"
% lines.
chan=1; % Use the red channel for fiducials.
x_lft=X_img(lft_srch_j);
y_lft=mean(Y_img(...
    srch_I(1)-1+...
    find(film_img(srch_I,lft_srch_j,chan)==max(film_img(srch_I,lft_srch_j,chan)))));
x_rt=X_img(rt_srch_j);
y_rt=mean(Y_img(...
    srch_I(1)-1+...
    find(film_img(srch_I,rt_srch_j,chan)==max(film_img(srch_I,rt_srch_j,chan)))));
x_top=mean(X_img(...
    srch_J(1)-1+...
    find(film_img(top_srch_i,srch_J,chan)==max(film_img(top_srch_i,srch_J,chan)))));
y_top=Y_img(top_srch_i);
x_bot=mean(X_img(...
    srch_J(1)-1+...
    find(film_img(bot_srch_i,srch_J,chan)==max(film_img(bot_srch_i,srch_J,chan)))));
y_bot=Y_img(bot_srch_i);

% The intersection of the these two lines defines the isocenter.
hor_slope=(y_rt-y_lft)/(x_rt-x_lft);
vert_slope=(y_bot-y_top)/(x_bot-x_top);
iso=[(y_lft-y_top-x_lft*hor_slope+x_top*vert_slope)/(vert_slope-hor_slope) 0];
if y_lft-y_rt~=0
    iso(2)=(x_top-x_lft+y_lft/hor_slope-y_top/vert_slope)/(1/hor_slope-1/vert_slope);
else
    iso(2)=y_lft;
end

% Use the slope of the lined in pixel space to calculate a rotation.
rot=mean(atand([hor_slope -1/(vert_slope)]));

% Register the raw image. Translation is accomplished by mounting the
% image onto the riginal grid shifted by the isocenter position.
% Rotation is accomplished by interpolating onto a symmetric grid and
% then rotating.
x_max=max([-X(1) X(end)]);
y_max=max([-Y(1) Y(end)]);
X_img_iso=[-fliplr(scan_res(1)/2:scan_res(1):x_max)...
    (scan_res(1)/2:scan_res(1):x_max)];
Y_img_iso=[-fliplr(scan_res(2)/2:scan_res(2):y_max)...
    (scan_res(2)/2:scan_res(2):y_max)];
film_img_reg1=imrotate(interp3(X_img-iso(1),Y_img-iso(2),[1 2 3],...
    double(film_img),X_img_iso',Y_img_iso,[1 2 3],'linear',0),...
    rot,'bilinear','crop');

% Convert to dose. Cropping images in GIMP reduces the the image to 8 bits,
% however the film header still says 16. Here we manually normalize the
% film image by 2^8 and apply the dose scaling contained in the header.
film_dose_reg1=film_img_reg1/(2^8-1)*...
    str2double(regexprep(film_info.YClipPathUnits,'0 - ([0-9.]*).*','$1'));

% Execute an optimization registration based on pixel-wise deviations.
if reg2==1

    % Define the optimization region (OR). These are the dimensions of the
    % plane that will be used compute the objective function below.
    OR=[-4 4; -7.5 7.5]*(100+d_plane)/100;
    OR_J=find(X_img_iso>OR(1,1) and X_img_iso<OR(1,2));
    OR_I=find(Y_img_iso>OR(2,1) and Y_img_iso<OR(2,2));

    % Define the search range and resolution.  Note that including a
    % rotational dimension takes a long time.
    trans_lim=0.2; % cm
    x_trans_ext=round(trans_lim/scan_res(1));
    y_trans_ext=round(trans_lim/scan_res(2));
    srch_I=-y_trans_ext:y_trans_ext;
    srch_J=-x_trans_ext:x_trans_ext;
    %     rot_srch=-0.5:0.02:0.5; % degrees
    rot_srch=[-0.1 -0.05 0 0.05 0.1];

    % Interpolate the comparison plane onto the film grid. This will allow
    % simple shifts of the film plane by scan_res increments.
    comp_dose=interp2(X,Y,MC_dose,X_img_iso(OR_J),Y_img_iso(OR_I)');

    % Conduct the optimization by first rotating, and then shifting the
    % film image by the specified increments. For each iteration, an
    % objective funtion value, along with its shifts, will be stored. Do
    % rotations first so that if there arent any, imrotate is only called
    % once.
    OF=zeros(numel(srch_I)*numel(srch_J)*numel(rot_srch),4);
    p=1;
    for nk=1:numel(rot_srch)

        % Shift the entire image first, so that edge values of the
        % extracted comparison plane are not corrupted by the rotation,
        % which could sabatoge the objective function values near the
        % edges.
        temp=imrotate(film_dose_reg1,rot_srch(nk),'bilinear','crop');

        % Now iterate through the shifts.
        for nj=srch_J
            for ni=srch_I

                % Calculate the objective function accross the OR. Use RMS
                % of the deviations so that OF value has some intuitive
                % meaning as the average deviation per pixel.
                OF(p,1)=sqrt(sum(sum(...
                    (temp(OR_I+ni,OR_J+nj)-comp_dose).^2))/...
                    (numel(OR_I)*numel(OR_J)));
                OF(p,2)=nj;
                OF(p,3)=ni;
                OF(p,4)=nk;
                p=p+1;
            end
        end
        [num2str(p) ' iterations complete']
    end

    % Find the shift and rotation increments for the minimum objective
    % function value.
    OF_min=min(OF(:,1))

    % Apply the optimized rotation and shifts. Rotation must occur first as
    % that is the order of the search above. Here an interpolation is used
    % because we can't index outside the dimensions of the array. Keep in
    % mind that a negative shift index is equivalent to a positive image
    % shift in reference to the comparison image. Rotations on the other
    % hand are indexed from a preselected grid of rotations (the rot_srch
    % array) so these are applied with the same sign.
    x_shift=scan_res(1)*OF(OF(:,1)==OF_min,2)
    y_shift=scan_res(2)*OF(OF(:,1)==OF_min,3)
    rot=rot_srch(OF(OF(:,1)==OF_min,4))

    % Rotate, then interpolate by shifting the source
    % grid.  Because we are shifting the source grid, basically
    % extracting a recentered image, we keep the sign of the shifts
    % above.
    film_dose_reg2=interp3(X_img_iso-x_shift,Y_img_iso-y_shift,[1 2 3],...
        imrotate(film_dose_reg1,rot,'bilinear','crop'),...
        X_img_iso,Y_img_iso',[1 2 3],'linear',0);
else
    film_dose_reg2=film_dose_reg1;
end

% Interpolate back onto the original grid.
film_dose=interp3(X_img_iso,Y_img_iso,[1 2 3],film_dose_reg2,X',Y,[1 2 3],...
    'linear',0);
```

gamma_calc.m

```
% This funtion compares two dose distributions.

function [gamma_map]=gamma_calc(base_dose,comp_dose,res,p_lim,d_lim)

% Set the starting resolution of search such that at least 10 test
% increments lie within the gamma elipse with the gradient of the base dose
% equal to zero. Also set a limit to how fine the resolution can become.
res_factor_0=10;
res_factor_lim=110;

% Calculate the absolute dose deviation limit from p_lim.
base_dose_max=max(max(max(base_dose)));
D_lim=p_lim*base_dose_max/100;
```

```
% For id calculations...
if isvector(base_dose)

    % Build index grid.
    I=(1:numel(base_dose(:,1)));

    % Calclute the magnitude of the gradient of the base array in order to
    % set an adaptive test resolution.
    grad=abs(gradient(base_dose))/res;

    % Calculate the resolution factor array and limit it by
    % res_factor_0 and the res_factor_lim. This adaptive res_factor will
    % insure that a least sufficient number of test points are checked in
    % high gradient regions.
    res_factor=res_factor_0*d_lim*grad/(2*D_lim);
    res_factor(res_factor<res_factor_0)=res_factor_0;
    res_factor(res_factor>res_factor_lim)=res_factor_lim;

    % Calculate the search resolution array and the
    srch_res=res./res_factor;

    % Set the coarse span of the search region.
    crse_span=round(d_lim/res);

    % Initialize gamma array
    gamma_map=ones(size(base_dose));

    % Loop through the coarse index grid.
    for i=I

        % Set coarse and fine test spans.
        I_span=intersect(i+(-crse_span:crse_span),I);
        I_span_fine=I_span(1):1/res_factor(i):I_span(end);

        % Interpolate the comparison dose onto the search grid and
        % extract the search profile.
        comp_dose_test=interp1(I_span,comp_dose(I_span),I_span_fine,'spline');

        % Build distance kernel and calculate the distance term of the
        % for the local gamma calculation.
        X=(I_span_fine-i)*srch_res(i);
        dist_term=abs(X)/d_lim;

        % Build the dose deviation term for the local gamma calculation.
        dev_term=(comp_dose_test-base_dose(i))/D_lim;

        % Find minimum gamma value
        %         [dist_term' dev_term' sqrt(dist_term.^2+dev_term.^2)']
        gamma_map(i)=min(sqrt(dist_term.^2+dev_term.^2));

        % test plot
        %         close all
        %         figure('Units','inches','OuterPosition',[7 1 6 6]);
        %         base_dose_test=interp1(I_span,base_dose(I_span),...
        %             I_span_fine,'spline');
        %         x_lims=i*res+1.2*[-d_lim d_lim];
        %         y_lims=base_dose(i)+1.2*[-D_lim D_lim];
        %         theta=linspace(0,2*pi);
        %         ellipse_y=base_dose(i)+D_lim*cos(theta);
        %         ellipse_x=i*res+d_lim*sin(theta);
        %         gamma_plot=plot(ellipse_x,ellipse_y,'-g');
        %         hold on
        %         plot(i*res+[-d_lim d_lim],[base_dose(i) base_dose(i)],'-g');
        %         comp_plot=plot(res*I_span_fine,comp_dose_test,'-or');
        %         base_plot=plot(res*I_span_fine,base_dose_test,'-b');
        %         hold off
        %         set(gca,'XLim',x_lims,'YLim',y_lims)
        %         legend([base_plot comp_plot gamma_plot],{'base','comp','gamma'})
        %         text(0.1,0.9,['Gamma = ' num2str(gamma_map(i),'%0.2f')],...
        %             'Units','normalized')
    end

elseif ndims(base_dose)==2

    % Build index grid.
    I=(1:numel(base_dose(:,1)));
    J=(1:numel(base_dose(1,:)));

    % Calclute the magnitude of the gradient of the base array in order to
    % set an adaptive test resolution.
    [grad_x,grad_y]=gradient(base_dose);
    grad=sqrt(grad_x.^2+grad_y.^2)/res;

    % Calculate the resolution factor array and limit it by
    % res_factor_0 and the res_factor_lim. This adaptive res_factor will
    % insure that a least sufficient number of test points are checked in
    % high gradient regions.
    res_factor=res_factor_0*d_lim*grad/(2*D_lim);
    res_factor(res_factor<res_factor_0)=res_factor_0;
    res_factor(res_factor>res_factor_lim)=res_factor_lim;

    % Calculate the search resolution array and the
    srch_res=res./res_factor;

    % Set the coarse span of the search region.
    crse_span=round(d_lim/res);

    % Initialize gamma array
    gamma_map=ones(size(base_dose));

    % Loop through the coarse index grid.
    for i=I
        for j=J

            % Set coarse and fine test spans.
            I_span=intersect(i+(-crse_span:crse_span),I);
            J_span=intersect(j+(-crse_span:crse_span),J);
            I_span_fine=I_span(1):1/res_factor(i,j):I_span(end);
            J_span_fine=J_span(1):1/res_factor(i,j):J_span(end);

            % Interpolate the comparison dose onto the search grid and
            % extract the search plane.
            comp_dose_test=interp2(J_span,I_span,comp_dose(I_span,J_span),...
                J_span_fine,I_span_fine','spline');

            % Build distance kernel and calculate the distance term of the
            % for the local gamma calculation.
            X=(J_span_fine-j)*srch_res(i,j);
            Y=(I_span_fine-i)*srch_res(i,j);
            [X_2d,Y_2d]=meshgrid(X,Y);
            dist_term=sqrt(X_2d.^2+Y_2d.^2)/d_lim;
            % Build the dose deviation term for the local gamma calculation.
            dev_term=(comp_dose_test-base_dose(i,j))/D_lim;

            % Find minimum gamma value
            % [dist_term' dev_term' sqrt(dist_term.^2+dev_term.^2)']
            gamma_map(i,j)=min(min(sqrt(dist_term.^2+dev_term.^2)));
        end
    end
end
```

Appendix C MCTPCF Code

Import and Parameter Extraction Scripts

anonymize.m

```
% This script anonymizes DICOM files.

% Set to one to re-anonymize patient data.
recalc=0;

patient_dir='/home/alex/Import Data';
recalc_dir='/home/alex/Projects/Plan Recalculation/Patients';

% Build a list of patients that have not yet been anonymized.
[~,patients]=system(['find ''' patient_dir '/'' -maxdepth 1 -mindepth 1']);
patients=textscan(patients,'%s','Delimiter','\n');
if recalc==0
    [~,anon_patients]=system(['find ''' recalc_dir '/'' -maxdepth 1 -mindepth 1']);
    ind=ones(numel(patients{1}),1);
    if ~isempty(anon_patients)
        anon_patients=textscan(anon_patients,'%s','Delimiter','\n');
        for m=1:numel(patients{1})
            if sum(strcmp(regexprep(anon_patients{1},'.*/(\w*)','$1'),...
                    regexprep(patients{1}{m},'.*/(\w*).*','$1')))>0
                ind(m)=0;
            end
        end
    end
    patients=patients{1}(ind==1);
else
    patients=patients{1};
end

% loop through patients
patients
for n=1:numel(patients);

    anon_patient=regexprep(patients{n},'.*/(\w*).*','$1');
    PN=struct('FamilyName',anon_patient,'GivenName',anon_patient);
    anon_dir=[recalc_dir '/' anon_patient];
    mkdir(anon_dir);
    copyfile(patients{n},anon_dir);

    % rename and anonymize CT set
    [~,CT_files]=system(['find ''' anon_dir ''' -regex ''.*/CT.*''' ]);
    if ~isempty(CT_files)
        CT_files=textscan(CT_files,'%s','Delimiter','\n');
        mkdir([anon_dir '/CT'])
        slice_z=zeros(numel(CT_files{1}),1);
        for c=1:numel(CT_files{1})
            CT_data=dicominfo(CT_files{1}{c});
            IPP_temp=CT_data.ImagePositionPatient;
            slice_z(c)=IPP_temp(3);
        end
        slice_num=(slice_z-min(slice_z)+1)*100;
        for c=1:numel(CT_files{1})
            ['Anonymizing ' anon_patient ' CT slice at ' num2str(slice_z(c)) ' mm.']
            CT_file=[anon_dir '/CT/CT.' anon_patient '.' num2str(slice_num(c)) '.dcm'];
            dicomwrite(dicomread(CT_files{1}{c}),CT_files{1}{c},dicominfo(CT_files{1}{c}),...
                'CreateMode','Copy',...
                'ReferringPhysicianName','',...
                'PhysicianOfRecord','',...
                'PatientName',PN,...
                'PatientID',anon_patient,...
                'PatientBirthDate','')
            movefile(CT_files{1}{c},CT_file)
        end
    end

    % rename and anonymize plan file
    [~,RP_files]=system(['find ''' anon_dir ''' -regex ''.*/RP\.[1-9].*''' ]);
    RP_files=textscan(RP_files,'%s','Delimiter','\n');
    plan_refs=cell(3,numel(RP_files{1}));
    for p=1:numel(RP_files{1})
        RP_data=dicominfo(RP_files{1}{p});
        plan=RP_data.RTPlanLabel;
        plan=regexprep(plan,' ','_');
        plan=regexprep(plan,':','-');
        plan=regexprep(plan,'and','');
        ['Anonymizing plan ' plan ' for patient ' anon_patient '.']
        dicomwrite([],RP_files{1}{p},dicominfo(RP_files{1}{p}),'CreateMode','Copy',...
            'ReferringPhysicianName','',...
            'PhysicianOfRecord','',...
            'PatientName',PN,...
            'PatientID',anon_patient,...
            'PatientBirthDate','')
        movefile(RP_files{1}{p},[anon_dir '/RP.' anon_patient '.' plan '.dcm'])

        % build beam number to beam name correlation cell
        B=numel(fieldnames(RP_data.BeamSequence));
        beam_refs=cell(2,B);
        for b=1:B
            beam_name=eval(['RP_data.BeamSequence.Item_' num2str(b) '.BeamName']);
            beam_name=regexprep(beam_name,' ','_');
            beam_name=regexprep(beam_name,':','-');
            beam_name=regexprep(beam_name,'and','');
            beam_refs{1,b}=beam_name;
            beam_refs{2,b}=eval(['RP_data.BeamSequence.Item_' num2str(b) '.BeamNumber']);
        end
        plan_refs{1,p}=plan;
        plan_refs{2,p}=RP_data.SOPInstanceUID;

plan_refs{3,p}=RP_data.ReferencedStructureSetSequence.Item_1.ReferencedSOPInstanceUID;
        plan_refs{4,p}=beam_refs;
    end

    % rename and anonymize structure set file
    [~,RS_files]=system(['find ''' anon_dir ''' -regex ''.*/RS.[1-9]*.*''' ]);
    if ~isempty(RS_files)
        RS_files=textscan(RS_files,'%s','Delimiter','\n');
        for s=1:numel(RS_files{1})
            RS_data=dicominfo(RS_files{1}{s});
            plan=plan_refs{1,strcmp(plan_refs(3,:),RS_data.SOPInstanceUID)};
            ['Anonymizing the structure set for ' plan ' for patient ' anon_patient '.']
            dicomwrite([],RS_files{1}{s},dicominfo(RS_files{1}{s}),'CreateMode','Copy',...
                'ReferringPhysicianName','',...
                'PhysicianOfRecord','',...
                'PatientName',PN,...
                'PatientID',anon_patient,...
                'PatientBirthDate','')
            movefile(RS_files{1}{s},[anon_dir '/RS.' anon_patient '.dcm'])
        end
    end

    % rename and anonymize dose files
    [~,RD_files]=system(['find ''' anon_dir ''' -regex ''.*/RD.[1-9]*.*''' ]);
    if ~isempty(RD_files)
        RD_files=textscan(RD_files,'%s','Delimiter','\n');
        for d=1:numel(RD_files{1})
            RD_file=RD_files{1}{d};
            RD_data=dicominfo(RD_file);
            UID=RD_data.ReferencedRTPlanSequence.Item_1.ReferencedSOPInstanceUID;
            p_sel=strcmp(plan_refs(2,:),UID);
            plan=plan_refs{1,p_sel};
            if strcmp(RD_data.DoseSummationType,'BEAM')
                beam_refs=plan_refs{4,p_sel};
                beam_num=eval(['RD_data.ReferencedRTPlanSequence.Item_1'...
                    '.ReferencedFractionGroupSequence.Item_1'...
                    '.ReferencedBeamSequence.Item_1'...
                    '.ReferencedBeamNumber']);
                beam=beam_refs{1,[beam_refs{2,:}]==beam_num};
            else
                beam='total';
            end
            ['Anonymizing the dose file for ' anon_patient ', beam: ' beam]
            dicomwrite(dicomread(RD_file),RD_file,RD_data,...
                'CreateMode','Copy',...
                'ReferringPhysicianName','',...
                'PhysicianOfRecord','',...
                'PatientName',PN,...
                'PatientID',anon_patient,...
                'PatientBirthDate','',...
                'DVHSequence','')
            movefile(RD_file,[anon_dir '/RD.' anon_patient '.' plan '.' beam '.dcm'])
        end
    end
end
```

plan_prep.m

```
% Script for preparing BEAM and DOSXYZ input files for the
% recalculation of RapidArc plans

recalc_dir='/home/alex/Projects/Plan Recalculation/Patients';
N=1e9;
calc_CT=1;

% suppress warnings
warning('off','all')

% loop through all patient plans
[~,patient_dirs]=system(['find ''' recalc_dir '/'' -maxdepth 1 -mindepth 1']);
patient_dirs=textscan(patient_dirs,'%s','Delimiter','\n');
patient_dirs{1}
for p=3%1:numel(patient_dirs{1});

    plan_name=regexprep(patient_dirs{1}{p},'.*/(\w*)','$1');
    ['Preparing patient ' plan_name ' for EGS recalculation.']

    % prepare input files
    RP_prep(patient_dirs{1}{p},N)

    % prepare dosxyz CT volume
    if calc_CT==1
        CT_prep(patient_dirs{1}{p})
    end
```

```
end
```

RP_prep.m

```
% This script prepares BEAM and DOSXYZ input files from DICOM RP files

function RP_prep(patient_dir,N_recalc)

patient=regexprep(patient_dir,'.*/(.*)$','$1');

% Get the list of plans for the selected patient.
[~,RP_files]=system(['find ''' patient_dir ''' -regex ''.*/RP\..*\.dcm''']);
RP_files=textscan(RP_files,'%s','Delimiter','\n');

% If CT data exists only one EGS input file set is created for each beam.
% If no CT data exists then two input file sets are generated.  One for
% each of the two water phantoms, wat_phant_3d and wat_phant_2d.
if exist([patient_dir '/CT'],'dir')
    phantom={'CT_phant'};
elseif strcmp(patient,'HSI_val')
    phantom={'HSI_wat_phant'};
elseif strcmp(patient,'LDI_val')
    phantom={'LDI_wat_phant'};
else
    phantom={'wat_phant_3d';'wat_phant_2d'};
end

% loop through plans
for p=1:numel(RP_files{1})

    plan=regexprep(RP_files{1}{p},'.*/RP\..*\.(.*)*\.dcm$','$1');
    linac='Clinac6X';

    linac_dir=[getenv('HOME') '/egsnrc/BEAM_' linac];
    phantom_dir=[getenv('HOME') '/egsnrc/dosxyznrc'];

    % source parameters
    E=5.9;
    I=0.15;
    D=1.0;

    % MLC leaf gap.
    tip_gap=0.03; % (cm)

    % MLC dimensions
    MLC_z_center=51.00;

[MLC_dims,MLC_thickness,MLC_z_start,MLC_y_start,MLC_leaf_gap]=MLC_dimensions(MLC_z_center);

    % get plan parameters
    [beam,JP,iso,TA,CA,MU,MLC,GA]=plan_pars(RP_files{1}{p});

    % build input files for each beam
    for b=1:numel([beam{1,:}])

        % create MLC file from control point data
        C=numel(GA{b});
        MLC_file=[patient '.' plan '.' beam{2,b} '.MLC'];
        MLC_fid=fopen([linac_dir '/' MLC_file],'w');
        fprintf(MLC_fid,'%s\n%u\n',MLC_file,C);
        for c=1:C
            fprintf(MLC_fid,'%f\n',MU{2,b}(c));
            L=numel(MLC{b}(c,:));
            % Write MLC projected MLC positions to the .MLC file.  Here, leaves
            % are listed in the Y2 to Y1 direction, or leaf 60 to leaf 1 in
            % Eclipse's leaf editor.  Positions are +/- distance in cm from
            % x=0, e.g. an X1 leaf to leaf to left of iso has a negative value.
            %  This is also what SYNCVMLC uses.
            X1_side=MLC_z_center*fliplr(MLC{b}(c,1:L/2)-tip_gap/2)'/100;  % bank B in Eclipse
            X2_side=MLC_z_center*fliplr(MLC{b}(c,L/2+1:end)+tip_gap/2)'/100; % bank A in
Eclipse
            for l=1:L/2
                fprintf(MLC_fid,'%f, %f, 1\n',X1_side(l),X2_side(l));
            end
        end

        %-------------BEAM INPUTS----------------------------------------

        linac_lines=textscan(fopen([linac_dir '/' linac
'_template.egsinp']),'%s','Delimiter','\n');

        % modify title
        linac_lines{1}{1}=[linac '.' patient '.' plan '.'...
            beam{2,b} '.egsinp, Beam Name: ' beam{2,b}];

        % Modify the directional bremsstrahlung radius.
        DBS_rad=max([abs(JP{b}(1,:)) abs(JP{b}(2,:))])+5;
        linac_lines{1}{5}=[num2str(DBS_rad) ', 100, 4, 21, 1, 12.5975'];

        % Modify the electron source parameters.
        src_pars=textscan(linac_lines{1}{6},'%s','Delimiter',',');
        src_pars{1}{3}=num2str(I,'-%0.2f');
        src_pars{1}{7}=num2str(D,'-%0.2f');
        src_pars{1}{8}=num2str(I,'-%0.2f');
        linac_lines{1}{6}=sprintf('%s, ',src_pars{1}{:});
        linac_lines{1}{8}=num2str(E,'%0.2f');

        % Modify the jaw positions.
        rec_1=find(~cellfun(@isempty,strfind(linac_lines{1},'identifier SECJAWS')),1);
        Y_z=sscanf(linac_lines{1}{rec_1+5},'%f,',2);
        X_z=sscanf(linac_lines{1}{rec_1+7},'%f,',2);
        Y_jaw=Y_z*-fliplr(JP{b}(2,:))/100;
        X_jaw=X_z*JP{b}(1,:)/100;
        linac_lines{1}{rec_1+5}=sprintf('%0.5f,
',Y_z,Y_jaw(1,2),Y_jaw(2,2),Y_jaw(1,1),Y_jaw(2,1));
        linac_lines{1}{rec_1+7}=sprintf('%0.5f,
',X_z,X_jaw(1,2),X_jaw(2,2),X_jaw(1,1),X_jaw(2,1));

        % Calculate the distance from the bottom of the accelerator to the
        % isocenter.
        rec_2=find(~cellfun(@isempty,strfind(linac_lines{1},'identifier AIRSLAB')),1);
        SAD=100-(double(sscanf(linac_lines{1}{rec_2+4},'%f',1))+...
            double(sscanf(linac_lines{1}{rec_2+5},'%f',1)));

        % Name the MLC file.
        linac_lines{1}{rec_1+30}=[linac_dir '/' patient '.' plan '.' beam{2,b} '.MLC'];

        % Modify MLC dimensions and positions
        linac_lines{1}{rec_1+17}=sprintf('%0.5f',MLC_z_start);
        linac_lines{1}{rec_1+18}=sprintf('%0.5f',MLC_thickness);
        linac_lines{1}{rec_1+19}=sprintf('%0.5f, ',MLC_dims(1,:));
        linac_lines{1}{rec_1+20}=sprintf('%0.5f, ',MLC_dims(2,:));
        linac_lines{1}{rec_1+21}=sprintf('%0.5f, ',MLC_dims(3,:));
        linac_lines{1}{rec_1+25}=sprintf('%0.5f',MLC_y_start);
        linac_lines{1}{rec_1+26}=sprintf('%0.5f',MLC_leaf_gap);

        % write new BEAM input file
        linac_file=[linac '.' patient '.' plan '.' beam{2,b} '.egsinp'];
        fprintf(fopen([linac_dir '/' linac_file],'w'),'%s\n',linac_lines{1}{:});
        %-------------DOSXYZ INPUT FOR RECALC----------------------------

        for g=1:numel(phantom)

            phantom_lines=textscan(fopen([phantom_dir '/CT_template.egsinp']),...
                '%s','Delimiter','\n');

            % modify title
            phantom_lines{1}{1}=[phantom{g} '.' patient '.'...
                plan '.' beam{2,b} '.egsinp, Beam Name: ' beam{2,b}];

            % Set the geometry file name.
            phantom_lines{1}{3}=[phantom_dir '/' phantom{g} '.' patient '.egsphant'];

            % set number of gantry positions
            gantry_pars=sscanf(phantom_lines{1}{6},'%d,');
            gantry_pars(3)=C;
            phantom_lines{1}{6}=sprintf('%d, ',gantry_pars);

            % start writing new DOSXYZ input file
            phantom_file=[phantom{g} '.' patient '.' plan '.' beam{2,b} '.egsinp'];
            phantom_fid=fopen([phantom_dir '/' phantom_file],'w');
            fprintf(phantom_fid,'%s\n',phantom_lines{1}{1:6});

            % list gantry positions
            gantry_lines=cell(C,1);
            for c=1:C
                gantry_lines{c}=sprintf('%f, ',iso(:)',TA{b}+90,...
                    GA{b}(c)-90,-CA{b}-90,SAD,MU{2,b}(c));
            end

            % Write the gantry positions to file.
            fprintf(phantom_fid,'%s\n',gantry_lines{:});

            % Set enflag etc.  The first value (enflag) must be 2 for a BEAM
            % simulation input.  The third (medsur) is set to 1 for air,
            % which is the first medium listed in the .egsphant files
            % created by CT_prep.m.  The fourth (dsurround(1)) is set to 50
            % for 50 cm of air surrounding the phantom on all sides.  The
            % fifth (dflag) is set to zero so all dsurround(1) applies to
            % all sides.
            enflag_pars=[2 0 1 50 0 0 0 0];
            phantom_lines{1}{8}=sprintf('%d, ',enflag_pars);

            % Set linac and BEAM name.
            phantom_lines{1}{9}=['BEAM_' linac ',' linac '.' patient '.'...
                plan '.' beam{2,b} ',700icru,0,0,'];

            % set number of particles to run
            hist_pars=sscanf(phantom_lines{1}{10},'%d,');
            hist_pars(1)=N_recalc;
            phantom_lines{1}{10}=sprintf('%d, ',hist_pars);

            % write the rest of the file
            fprintf(phantom_fid,'%s\n',phantom_lines{1}{8:end});
        end
    end
end
```

plan_pars.m

```
% This script extracts plan parameters from a DICOM plan file.
```

```
function [beam,JP,iso,TA,CA,MU,MLC,GA]=plan_pars(RP_file)

% Extract dicom data.
RP_data=dicominfo(RP_file);

% Build parameter arrays.
B=RP_data.FractionGroupSequence.Item_1.NumberOfBeams;
beam=cell(2,B);
JP=cell(B,1);
TA=cell(B,1);
CA=cell(B,1);
MU=cell(2,B);
MLC=cell(B,1);
GA=cell(B,1);

% Extract plan-wide parameters.  Here we assume that the isocenter does not
% change from beam to beam.
iso=eval(['RP_data.BeamSequence.Item_1'...
    '.ControlPointSequence.Item_1'...
    '.IsocenterPosition'])/10;
N_frac=eval(['RP_data.FractionGroupSequence.Item_1'...
    '.NumberOfFractionsPlanned']);

% Extract beam specific parameters.
for b=1:B

    % Extract the beam name to beam number correlation so that dose files
    % can be matched to the planned beam.
    beam{1,b}=eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.BeamNumber']);
    beam_name=eval(['RP_data.BeamSequence.Item_' num2str(b) '.BeamName']);
    beam_name=regexprep(beam_name,' ','_');
    beam_name=regexprep(beam_name,':','-');
    beam_name=regexprep(beam_name,'and','');
    beam{2,b}=beam_name;

    % Extract the MU per fraction and multiply by N_frac to get the total
    % treatment MUs.
    MU{1,b}=eval(['RP_data.FractionGroupSequence.Item_1'...
        '.ReferencedBeamSequence.Item_' num2str(b)...
        '.BeamMeterset'])*N_frac;

    % Extract the number of control points.
    C=eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.NumberOfControlPoints']);

    % Extract jaw positions.  In the DICOM file, the x positions are in the
    % order X1, X2, the y positions are flipped however, and must be
    % switched and negated, i.e. [-2,4] -> [-4,2].
    JP{b}=[eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.ControlPointSequence.Item_1'...
        '.BeamLimitingDevicePositionSequence.Item_1'...
        '.LeafJawPositions'])';...
        eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.ControlPointSequence.Item_1'...
        '.BeamLimitingDevicePositionSequence.Item_2'...
        '.LeafJawPositions'])']/10;

    % Extract table and collimator angle. For the treatments this script is
    % applicable for, jaw position, table angle, and collimator angle are
    % all static while the beam is on.
    TA{b}=eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.ControlPointSequence.Item_1'...
        '.TableTopEccentricAngle']);
    CA{b}=eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.ControlPointSequence.Item_1'...
        '.BeamLimitingDeviceAngle']);

    % Build temp arrays for MLC positions, cumulative beam weight, and
    % gantry angle.
    MU_temp=zeros(C,1);
    MLC_temp=zeros(C,120);
    GA_temp=zeros(C,1);



    % Get the initial MLC postions. DICOM lists leaf positons first along the
    % X1 side, then along the X2 side, in the Y1 to Y2 direction.
    MLC_temp(1,:)=eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.ControlPointSequence.Item_1'...
        '.BeamLimitingDevicePositionSequence.Item_3'...
        '.LeafJawPositions'])/10;

    % Get the initial gantry angle.
    GA_temp(1)=eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.ControlPointSequence.Item_1'...
        '.GantryAngle']);

    % Get the initial cumulative MU weight.
    MU_temp(1)=eval(['RP_data.BeamSequence.Item_' num2str(b)...
        '.ControlPointSequence.Item_1'...
        '.CumulativeMetersetWeight']);

    % Extract parameters for the remaining control points.
    for c=2:C

        % Every other control point should have an MU weight.
        MU_temp(c)=eval(['RP_data.BeamSequence.Item_' num2str(b)...
            '.ControlPointSequence.Item_' num2str(c)...
            '.CumulativeMetersetWeight']);

        % First check if different MLC positions exist. If they do extract
        % them if not, copy the previous control points positions.
        MLC_struct=eval(['RP_data.BeamSequence.Item_' num2str(b)...
            '.ControlPointSequence.Item_' num2str(c)]);
        if isfield(MLC_struct,'BeamLimitingDevicePositionSequence')

MLC_temp(c,:)=MLC_struct.BeamLimitingDevicePositionSequence.Item_1.LeafJawPositions/10;
        else
            MLC_temp(c,:)=MLC_temp(c-1,:);
        end

        % First check a different gantry angle exists. If if it does, extract
        % it, if not, copy the previous angle.
        GA_struct=eval(['RP_data.BeamSequence.Item_' num2str(b)...
                '.ControlPointSequence.Item_' num2str(c)]);
        if isfield(GA_struct,'GantryAngle')
            GA_temp(c)=GA_struct.GantryAngle;
        else
            GA_temp(c)=GA_temp(c-1);
        end
    end
    % Fill parameter cells.
    MU{2,b}=MU_temp;
    MLC{b}=MLC_temp;
    GA{b}=GA_temp;
end
```

CT_prep.m

```
% This function prepares a DOSXYZ compatible CT geometry file from
% a list DICOM CT files

function CT_prep(patient_dir)

patient=regexprep(patient_dir,'.*/(\w)*$','$1');
phantom_dir=[getenv('HOME') '/egsnrc/dosxyznrc'];

% Media definition for phantom construction.
media_def_tissue={
    {'AIR700ICRU',[-3000,-950]};
    {'LUNG700ICRU',[-950 -600]};
    {'ICRUTISSUE700ICRU',[-600 500]};
    {'ICRPBONE700ICRU',[500 3000]}};
media_def_water={
    {'AIR700ICRU',[-3000,-950]};
    {'H2O700ICRU',[-950 10000]}};

% If CT data exists build a patient based .egsphant file.
if exist([patient_dir '/CT'],'dir')

    % Build the EGS calculation grid based on the RD grid.
    [~,RD_files]=system(['find ''' patient_dir ''' -regex ''.*/RD\..*\.dcm''']);
    RD_files=textscan(RD_files,'%s','Delimiter','\n');
    RD_data=dicominfo(RD_files{1}{1});
    IPP=RD_data.ImagePositionPatient;
    vox=[RD_data.PixelSpacing(2) RD_data.PixelSpacing(1)];
    X=(0:double(RD_data.Columns)-1)'*vox(1)+IPP(1);
    Y=(0:double(RD_data.Rows)-1)'*vox(2)+IPP(2);
    Z=double(IPP(3)+RD_data.GridFrameOffsetVector);
    vox=[vox Z(2)-Z(1)];
    X_lims=[X-vox(1)/2; X(end)+vox(1)/2]/10;
    Y_lims=[Y-vox(2)/2; Y(end)+vox(2)/2]/10;
    Z_lims=[Z-vox(3)/2; Z(end)+vox(3)/2]/10;

    % Get a list of the CT slice files and extract the IPP's so they can be
    % reordered via the offset_vec array.
    [~,CT_files]=system(['find ''' patient_dir '/CT/'' -regex ''.*/CT\..*\.dcm''']);
    CT_files=textscan(CT_files,'%s','Delimiter','\n');
    offset_vec=zeros(numel(CT_files{1}),1);
    for f=1:numel(CT_files{1})
        CT_data=dicominfo(CT_files{1}{f});
        IPP_temp=CT_data.ImagePositionPatient;
        offset_vec(f)=IPP_temp(3);
    end

    % Build the raw CT grid with a matching HU array.
    vox_raw=[CT_data.PixelSpacing(1) CT_data.PixelSpacing(2)];
    IPP_raw=CT_data.ImagePositionPatient;
    HU_offset=CT_data.RescaleIntercept;
    HU_slope=CT_data.RescaleSlope;
    X_raw=(0:double(CT_data.Columns)-1)'*vox_raw(2)+IPP_raw(1);
    Y_raw=(0:double(CT_data.Rows)-1)'*vox_raw(1)+IPP_raw(2);
    Z_raw=sort(offset_vec);
    HU_raw=zeros(numel(Y_raw),numel(X_raw),numel(CT_files{1}));
    for k=1:numel(CT_files{1})
        ['Converting ' patient ' CT slice at z = ' num2str(offset_vec(k)) ' mm.']
        slice_temp=HU_offset+double(dicomread(CT_files{1}{k}))*HU_slope;
        HU_raw(:,:,Z_raw==offset_vec(k))=slice_temp;
    end

    % Interpolate the raw HU grid onto the calculation grid.  Set extra
    % voxel to HU air.
    HU=interp3(X_raw,Y_raw,Z_raw,HU_raw,X',Y,Z,'linear',-1000);

    % Write the .egsphant file.
    egsphant_file=[patient_dir '/CT_phant.' patient '.egsphant'];
    egsphant_write(egsphant_file,HU,X_lims,Y_lims,Z_lims,media_def_tissue)
    copyfile(egsphant_file,phantom_dir)

    % Build water phantoms for MLC validation. The first is 30x25x30 cm,
    % 0.25x0.25x0.25 cm, 100 SSD water phantom for 3d comparisons.  The second
    % is the same size but with 0.1x0.1 cm planes centered at z=[1.5 5 10 15]
    % for film comparisons.
elseif strcmp(patient,'MLC_val')

    % Build lower resolution fully voxelized phantom.
    %     vox=0.25; % (cm)
        width=30; % (cm)
        depth=30; % (cm)
    %     X_lims=-width/2:vox:width/2;
    %     Z_lims=X_lims;
    %     Y_lims=0:vox:depth;
    %     HU=zeros(numel(Y_lims)-1,numel(X_lims)-1,numel(Z_lims)-1);
    %     egsphant_file=[patient_dir '/wat_phant_3d.' patient '.egsphant'];
    %     egsphant_write(egsphant_file,HU,X_lims,Y_lims,Z_lims,media_def_water)
    %     copyfile(egsphant_file,phantom_dir)

    % Build higher resolution plane phantom.
    vox=0.1; % (cm)
    X_lims=-width/2:vox:width/2;
    Z_lims=X_lims;
    Y_lims=[0 1.45 1.55 4.95 5.05 9.95 10.05 14.95 15.05 20];
    HU=ones(numel(Y_lims)-1,numel(X_lims)-1,numel(Z_lims)-1);
    HU(:,:,:)=90; % for solid water
    egsphant_file=[patient_dir '/wat_phant_2d.' patient '.egsphant'];
    egsphant_write(egsphant_file,HU,X_lims,Y_lims,Z_lims,media_def_water)
    copyfile(egsphant_file,phantom_dir)

    % Build the HSI validation phantom with 1x1x1 mm^2 voxels z with a half of
    % the volume (x>0) having a series of density (or material) changes.  The
    % order is 2 cm water, 1 cm cortical bone, 6 cm lung, and the rest is
    % water.
elseif strcmp(patient,'HSI_val')

    % Build the HSI validation phantom.
    y_iso_offset=0;
    bone_lims=[2 3]+y_iso_offset;
    lung_lims=[3 9]+y_iso_offset;
    vox=0.1; % (cm)
    width=20; % (cm)
    depth=12; % (cm)
    X_lims=[-width/2 -width/10:vox:width/10 width/2];
    X=(X_lims(2:end)+X_lims(1:end-1))/2;
    Z_lims=X_lims;
    Y_lims=(0:vox:depth)+y_iso_offset;
    Y=(Y_lims(2:end)+Y_lims(1:end-1))/2;
    HU=ones(numel(Y_lims)-1,numel(X_lims)-1,numel(Z_lims)-1)*100;
    HU((Y>bone_lims(1) and Y<bone_lims(2)),X>0,:)=770;
    HU((Y>lung_lims(1) and Y<lung_lims(2)),X>0,:)=-700; % lung
    egsphant_file=[patient_dir '/HSI_wat_phant.' patient '.egsphant'];
    egsphant_write(egsphant_file,HU,X_lims,Y_lims,Z_lims,media_def_water)
    copyfile(egsphant_file,phantom_dir)

    % Build the heterogeneous tissue phantom at 0.25 cm^3 resolution.
    %     egsphant_file=[patient_dir '/hetero_tiss_phant.' patient '.egsphant'];
    %     egsphant_write(egsphant_file,HU,X_lims,Y_lims,Z_lims,media_def_tissue)
    %     copyfile(egsphant_file,phantom_dir)
end

% Get the list of .egsphant files created for the selected patient.
[~,phant_files]=system(['find ''' patient_dir ''' -regex ''.*/.*\.egsphant''']);
if ~isempty(phant_files)
    phant_files=textscan(phant_files,'%s','Delimiter','\n');
    for g=1:numel(phant_files{1})
        [ED,media,X,Y,Z,~]=egsphant_read(phant_files{1}{g});
        egsphant_view(ED,media,X,Y,Z)
    end
end
```

CT_ramp.m

```
% density ramp function
function [rho]=CT_ramp(hu)

% HU ramp data
HU_data=[...
    -3000.0 0.0;
    -1000.0 0.0;
     -714.7 0.265;
     -546.6 0.431;
     -103.6 0.924;
      -60.2 0.957;
       -8.3 0.990;
       12.7 1.049;
       62.4 1.062;
      186.6 1.082;
      207.4 1.099;
      437.1 1.279;
      793.1 1.471;
     1190.8 1.693;
     2212.0 2.680;
     3260.0 3.300
     5000.0 4.150
    10000.0 5.331
    29768.0 8.1];

[N,~]=size(HU_data);
rho=0;
for n=1:N
    if hu>=HU_data(n,1)
        rho=HU_data(n,2)+(hu-HU_data(n,1))*...
            (HU_data(n+1,2)-HU_data(n,2))/...
            (HU_data(n+1,1)-HU_data(n,1));
    end
end
```

```
end
```

Cluster Submission and Extraction Scripts

dosxyz_parallel.sh

```
#!/bin/bash
#
# This script will send DOSXYZ jobs to the cluster.  It inputs a DOSXYZ input
# file and the number of parallel jobs to run.  It also builds a clean up script
# that cleans up the working directories on the scratch volume.  The environment variable
# SCRATCH_HOME must be defined for this script to run.
#
# Usage: sh dosxyz_parallel_job.sh <phantom input file> <num. of parallel jobs>

# Name the inputs.
phant_file=$1
N_parallel=$2

# Jobs will run more efficiently from the scratch volume as it is located physically
# closer to the computation nodes.  Because the scratch volume is likely to be out
# of the user's control, the user's originial home directory is used to store and
# maintain the egsnrc and HEN_HOUSE directories.  Each time a parallel job is started,
# these directories are copied (or simple updated) over to the scratch volume.  Input
# files, MLC files, and geometry files will be excluded here and copied over in
# separate commands in order to reduce congestion in the working directories.
# Here this is done by executing the bash script sync_egs.sh.
sh $HOME/egsnrc/scripts/sync_egs.sh

# Name the directory where the clean up script will send the important
# output files.
projects_dir=/nfs/depot/nerc_u1/egana/Projects

# Name the pegs input file that will be used.
PEGS_file=700icru

# Extract the system name from the EGS configuration file.  This tells EGS where
# the relevent binaries and libraries are located.
system=`cat $EGS_CONFIG | grep "my_machine =" | sed 's/my_machine = //'`

# Extract the BEAM executable and input file names from the DOSXYZ input file.
linac_file=`awk -F, '/BEAM_/ {print $2}' $HOME/egsnrc/dosxyznrc/$phant_file.egsinp`
linac=`awk -F, '/BEAM_/ {print $1}' $HOME/egsnrc/dosxyznrc/$phant_file.egsinp`

# Create a job directory to hold the submit scripts and log files so that multiple
# groups of parallel jobs can run at the same time while using the same naming schemes.
# This will also be where the clean up script collects the ouputs and is the directory
# that is sent to the projects directory.
job_dir=$SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file
mkdir -p $job_dir

# Extract the DOSXYZ source number from the input file.  This will determine
# what kind of job is being run and how the input files will be modified, e.g. some
# path names may need to be changed to coincide with the scratch volume working
# environment.
src=`awk -F, '
        NR==2   {
                        r0=$1
                        if (r0 < 1)
                                r1 = 6
                        else
                                r1 = r0 + 4
                        }
        NR==r1  {
                        if (r0 < 1)
                                r4 = r1
                        else{
                                if ($1 < 1)
                                        r2 = r1 + $1 * -1 + 1
                                else
                                        r2 = r1 + $1 + 1
                                if ($2 < 1)
                                        r3 = r2 + $2 * -1 + 1
                                else
                                        r3 = r2 + $2 + 1
                                if ($3 < 1)
                                        r4 = r3 + $3 * -1 + 5
                                else
                                        r4 = r3 + $3 + 5
                                }
                        }
        NR==r4  {print $2}' $HOME/egsnrc/dosxyznrc/$phant_file.egsinp`
src=`echo $src | sed 's/ *//'`

# If DOSXYZ is using the SYNCVMLC source...
if [ $src -eq 21 ]; then

        # Copy over the input files.
        cp $HOME/egsnrc/$linac/$linac_file.egsinp $SCRATCH_HOME/egsnrc/$linac
        cp $HOME/egsnrc/dosxyznrc/$phant_file.egsinp $SCRATCH_HOME/egsnrc/dosxyznrc

        # Extract various names from the DOSXYZ input file name so that the .egsphant
        # and .MLC file names can be identified.
        geom_name=`echo $phant_file | sed 's/\(.*\)\..*\..*\..*/\1/'`
        patient_name=`echo $phant_file | sed 's/.*\.\(.*\)\..*\..*/\1/'`
        plan_name=`echo $phant_file | sed 's/.*\..*\.\(.*\)\..*/\1/'`
        beam_name=`echo $phant_file | sed 's/.*\..*\..*\.\(.*\)/\1/'`
        plan_dir=${projects_dir}/Patients/$patient_name


        # Copy over the .MLC file and modify its path in the BEAM input file.
        cp $HOME/egsnrc/$linac/$patient_name.$plan_name.$beam_name.MLC
$SCRATCH_HOME/egsnrc/$linac
        awk -v name=$SCRATCH_HOME/egsnrc/$linac/$patient_name.$plan_name.$beam_name.MLC '
                /SYNCVMLC/ {rec=NR+17}
                NR==rec {$0=name}
                {print}' $SCRATCH_HOME/egsnrc/$linac/$linac_file.egsinp > temp
        mv temp $SCRATCH_HOME/egsnrc/$linac/$linac_file.egsinp

        # Extract the number of media from the DOSXYZ input file.  This value will be
        # zero if the simulation is using an .egsphant geometry file.
        ct=`awk -F, 'NR==2     {print $1}' $HOME/egsnrc/dosxyznrc/$phant_file.egsinp`
        ct=`echo $ct | sed 's/ *//'`

        # If using an .egsphant geometry file...
        if [ $ct -eq 0 ]; then

                # Check is the .egsphant file already exists.  If it doesn't copy it over to
                # the scratch volume.
                if [ ! -f $SCRATCH_HOME/egsnrc/dosxyznrc/$geom_name.$patient_name.egsphant ];
then
                        cp $HOME/egsnrc/dosxyznrc/$geom_name.$patient_name.egsphant
$SCRATCH_HOME/egsnrc/dosxyznrc
                fi

                # Modify the path of the .egsphant file in the DOSXYZ input file.
                awk -v name1=$SCRATCH_HOME/egsnrc/dosxyznrc/$geom_name.$patient_name.egsphant
'
                        NR==3 {$0=name1}
                {print}' $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.egsinp > temp
        mv temp $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.egsinp
    fi

    # Define where the clean up script will send the simulation outputs.
    archive_dir="$projects_dir/Plan Recalculation/Patients/$patient_name"

# If using a static gantry BEAM source...
elif [ $src -eq 9 ]; then

    geom_name_base=`echo $phant_file | sed 's/\([a-zA-Z]*\)_.*/\1/'`

    if [ $geom_name_base = 'head' ]; then

        # MOVE over the input files, so they don't clog things up.
        mv $HOME/egsnrc/$linac/$linac_file.egsinp $SCRATCH_HOME/egsnrc/$linac
        mv $HOME/egsnrc/dosxyznrc/$phant_file.egsinp $SCRATCH_HOME/egsnrc/dosxyznrc

        # Extract the source parameters from the BEAM input file name so that
simulations
        # results can be placed in the right location.
        FS=`echo $linac_file | sed 's/\w*_\w*_\([0-9.]*\)_[0-9.]*_[0-9.]*_[0-
9.]*$/\1/'`
        BS=`echo $linac_file | sed 's/\w*_\(\w*\)_[0-9.]*_[0-9.]*_[0-9.]*_[0-
9.]*$/\1/'`
        E=`echo $linac_file | sed 's/\w*_\w*_[0-9.]*_\([0-9.]*\)_[0-9.]*_[0-
9.]*$/\1/'`
        I=`echo $linac_file | sed 's/\w*_\w*_[0-9.]*_[0-9.]*_\([0-9.]*\)_[0-
9.]*$/\1/'`

        # Define where the clean up script will send the simulation outputs.
        par_dir=$BS/${FS}x${FS}/$E/$I
        archive_dir=$projects_dir/job_archive/Clinac6X/$par_dir/$phant_file

    elif [ $geom_name_base = 'LDI' ]; then

        # MOVE over the input files, so they don't clog things up.
        cp $HOME/egsnrc/$linac/$linac_file.egsinp $SCRATCH_HOME/egsnrc/$linac
        cp $HOME/egsnrc/dosxyznrc/$phant_file.egsinp $SCRATCH_HOME/egsnrc/dosxyznrc

        # Extract the phantom HU build the .egsphant name.
        HU=`echo $phant_file | sed 's/\w*\.\(n[0-9]*\)\..*/\1/'`
        geom_name=LDI_wat_phant.$HU

        # Copy .egsphant file over to the scratch volume.
        cp $HOME/egsnrc/dosxyznrc/$geom_name.egsphant $SCRATCH_HOME/egsnrc/dosxyznrc

        # Modify the path of the .egsphant file in the DOSXYZ input file.
        awk -v name1=$SCRATCH_HOME/egsnrc/dosxyznrc/$geom_name.egsphant '
                NR==3 {$0=name1}
                {print}' $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.egsinp > temp
        mv temp $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.egsinp

        archive_dir=$projects_dir/LDI\ Validation/$phant_file
    fi

# If using a simple source (source 1)...
elif [ $src -eq 1 ]; then

    # Copy over the input file.
    cp $HOME/egsnrc/dosxyznrc/$phant_file.egsinp $SCRATCH_HOME/egsnrc/dosxyznrc

    # Define where the clean up script will send the simulation outputs.
    archive_dir="$projects_dir/HSI Validation"

fi
```

```
# Build the clean up script.  This script will first test for the existence of the
# .lock file to see if the job is even running.  If the job is running, the script then
# extracts the number of particles left to be simulated.  If the .3ddose output file doesn't
# exist and there are no particles left to simulate, the script calls dosxyznrc to force the
# summing of the .pardose files in the working directory.  The script then tests for the
.3ddose
# file as it would if no .lock file exists.  If it does, the script copies important outputs
# to the job directory, cleans both the working directory and job directory of extraneous
# files and directories, copies the cleaned job directory to the archive location, and then
# finally deletes itself.
# NOTE: There is a lag between the start of the last batches, which is when the remaining
# particles field in the .lock file is set to zero, and the automatic summing of the .pardose
# files.  Because of this, a small fraction of histories can be lost, but it is worth it as
# sometimes the last job of a parallel run will fail and never produce a .3ddose file on its
own.
# This functionality should probably be replaced with automatic resubmission of failed jobs.
mkdir -p $HOME/egsnrc/dosxyznrc/clean_up
echo -e "#! /bin/bash \
\n# Script for cleaning up job $phant_file \
\n \
\n# get number of remaining particles \
\nif [ -f $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.lock ]; then \
\n     RH=\`awk 'NR == 1 {print \$1}' $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.lock\` \
\n \
\n     # test if .3ddose doesn't exist and all histories are run, if so resum \
\n     if [ ! -f $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.3ddose ] andand [ \$RH -eq 0 ];
then \
\n             sh $HOME/egsnrc/scripts/resum.sh dosxyznrc $phant_file \
\n     fi \
\nfi \
\n \
\n# test if job is done by existence of .3ddose file \
\nif [ -f $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.3ddose ]; then \
\n \
\n     # move DOSXYZ files to job directory \
\n     mv $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.3ddose $job_dir \
\n     mv $SCRATCH_HOME/egsnrc/dosxyznrc/$phant_file.egsinp $job_dir \
\n \
\n     # rename fort.1 file \
\n  cp $job_dir/fort.1 $job_dir/$phant_file.log \
\n \
\n     # delete submit scripts, joblog files, and fort files \
\n     rm -f $job_dir/submit* \
\n     rm -f $job_dir/*joblog \
\n     rm -f $job_dir/fort* \
\n \
\n  # sum .egsdat files into an .egslst file and move BEAM files the job directory
\n  sh $HOME/egsnrc/scripts/resum.sh $linac $linac_file \
\n     mv $SCRATCH_HOME/egsnrc/$linac/$linac_file.egsinp $job_dir \
\n     mv $SCRATCH_HOME/egsnrc/$linac/$linac_file.egslst $job_dir \
\n     mv $SCRATCH_HOME/egsnrc/$linac/$patient_name.$plan_name.$beam_name.MLC $job_dir \
\n \
\n     # move job directory to archive locations \
\n     mkdir -p \"$archive_dir\" \
\n     mv -f $job_dir/* \"$archive_dir\" \
\n  rm -rf $job_dir
\n \
\n     # clean up the working directories \
\n     rm -rf $SCRATCH_HOME/egsnrc/dosxyznrc/*.pardose \
\n     rm -rf $SCRATCH_HOME/egsnrc/dosxyznrc/*.errors \
\n     rm -rf $SCRATCH_HOME/egsnrc/dosxyznrc/egsrun*$phant_file \
\n     rm -rf $SCRATCH_HOME/egsnrc/$linac/egsrun*$linac_file \
\n \
\n  # delete this script \
\n  rm -f $HOME/egsnrc/dosxyznrc/clean_up/clean_up_$phant_file \
\n \
\nelse \
\n      echo \"The .3ddose file doesn't exist.  The job is either not done or broken.\" \
\nfi" > $HOME/egsnrc/dosxyznrc/clean_up/clean_up_$phant_file

# Change the clean up script permissions so that it can be executed.
chmod u+x $HOME/egsnrc/dosxyznrc/clean_up/clean_up_$phant_file

# Build the actual DOSXYZ executable string with the required inputs.
command="$SCRATCH_HOME/egsnrc/bin/$system/dosxyznrc -p $PEGS_file -i $phant_file -P
$N_parallel"

# Build the submit scripts.  Executable output is directed to log files that are named
# by job number and use the .joblog extension.  Export the necessary environment variables
# so that the jobs execute on the scratch volume.
for j in $(seq 1 $N_parallel); do
       echo -e "#! /bin/bash \
       \n# DOSXYZ submit script for $phant_file \
       \n#$ -cwd -j y -V \
       \n#$ -o $job_dir/${phant_file}_$j.joblog \
       \n#$ -N job_$j \
       \nexport HOME=$SCRATCH_HOME \
       \nexport EGS_HOME=$SCRATCH_HOME/egsnrc \
       \nexport HEN_HOUSE=$SCRATCH_HOME/HEN_HOUSE \
       \n$command -j $j -f 1" > $job_dir/submit_$j
done

# Change the current directory to the job directory and run the submit scripts.
# Delays between submissions may need to be tweeked depending on the rate job
# failures.
cd $job_dir
qsub $job_dir/submit_1
sleep 1
for j in $(seq 2 $N_parallel); do
       qsub $job_dir/submit_$j
       sleep 0.1
done
```

resum.sh

```
#!/bin/bash

# This script changes the IRESTART value of either a BEAM or DOSXYZ input
# file from zero to 4 and then sums the .pardose or .egslst files.
# EGS_HOME must set for this to work.

# Usage: resum <executable> <input file, no ext.>

PEGS_file='700icru'

# If a DOSXYZ input file...
if [ $1 = "dosxyznrc" ]; then
       awk 'BEGIN {FS = ",";OFS = ","}
              /BEAM_/ {rec = NR+1}
              NR==rec {$8 = " 4"}
              {print}' $EGS_HOME/$1/$2.egsinp > temp
       mv temp $EGS_HOME/$1/$2.egsinp

# Else it is a BEAM input...
else
       awk 'BEGIN {FS = ",";OFS = ","}
              NR==3 {$3 = " 4"}
              {print}' $EGS_HOME/$1/$2.egsinp > temp
       mv temp $EGS_HOME/$1/$2.egsinp
```

```
fi

# Execute the summation.
$1 -p $PEGS_file -i $2
```

Output Analysis Scripts

output_analyze.m

```
% This script compares MC calculated treatment plans to AAA calclulations.
% It offers profile, planar, and 3d viewing options as well as the
% capablility of switching between transversem, coronal, and sagital views.
% The user must select a patient, plan, and beam (or total dose) to view.
% Gamma analysis is available for profile and planar comparisons. 3d gamma
% has not yet been implemented.

% Select patient, plan, and beam to view.
patient='IMAT_P1';
plan='Prostate';
beam='CW_Partial'; % set to 'total' for total dose
phant='CT_phant';

% Set to one to update dose arrays, otherwise load them from file.
update=1;

% Define imaging and analysis parameters.
image_mode='profile'; % Define the type of imaging {'profile','planar','3D'}.
view_type='coronal'; % Select the type of planes to view.
dev_type='gamma'; % Select deviation type for the 3D view {'percent','absolute'}.
cal_method=2; % Set to 1 to use monitor chamber dose, 2 to use AAA dose scaling.
p_lim=3; % Set the gamma percent deviaiton limit (%).
d_lim=0.3; % Set the gamma DTA limit (cm).
pause_val=0; % set to zero for user stepping
unc_lim=2; % (%)
unc_mask_lim=2; % (%)
ED_lim=0.1;
abs_dev_lim=5; % (Gy)
pcnt_dev_lim=10; % (%)
low_dose_lim=5; % (Gy)
base_type='AAA_wat';
comp_type='MC_wat';

% Define fig size for the current machine.
global fig_size

% Some constant directories and names.
recalc_dir='/home/alex/Projects/Plan Recalculation/Patients';
patient_dir=[recalc_dir '/' patient];
save_file=[patient_dir '/' patient '.mat'];
linac='Clinac6X';

%==============================================================================
%                     DATA EXTRACTION
%==============================================================================

% extract arrays from files or use saved arrays
if update==1

    % Get the MC dose file names and extract the MC grid.
    [~,MC_files]=system(...
        ['find ''' patient_dir '/'' -regex ''.*/' phant...
        '\.' patient '\..*\.3ddose''']);
    MC_files=textscan(MC_files,'%s','Delimiter','\n');
    [~,~,X,Y,Z]=MC_read(MC_files{1}{1});
    I=numel(Y);
    J=numel(X);
    K=numel(Z);

    % Get plan parameters
    RP_file=[patient_dir '/RP.' patient '.' plan '.dcm'];
    [beam_lab,JP,iso,TA,CA,MU,MLC,GA]=plan_pars(RP_file);
    beams={beam_lab{2,:} 'total'}';
    B=numel(beams);

    % Get RD file name and extract the AAA grid.
    RD_file=[patient_dir '/RD.' patient '.' plan '.' beams{1} '.dcm'];
    RD_data=dicominfo(RD_file);
    IPP=RD_data.ImagePositionPatient;
    X_AAA=round(((0:double(RD_data.Columns)-1)'*...
        RD_data.PixelSpacing(1)+IPP(1))*100)/1000;
    Y_AAA=round(((0:double(RD_data.Rows)-1)'*...
        RD_data.PixelSpacing(2)+IPP(2))*100)/1000;
    Z_AAA=(IPP(3)+RD_data.GridFrameOffsetVector)/10;

    % Build arrays from the MC grid.
    MC_med_dpp=zeros(I,J,K,B);
    MC_wat_dpp=zeros(I,J,K,B);
    MC_unc=zeros(I,J,K,B);
    AAA_wat=zeros(I,J,K,B);

    % loop through beams
    for b=1:B-1

        % Get MC dose-to-water and uncertainty.
        MC_file=[patient_dir '/' phant '.' patient '.' plan...
            '.' beams{b} '.3ddose'];
        [MC_med_dpp(:,:,:,b),MC_unc(:,:,:,b),~,~,~]=MC_read(MC_file);

        % Get the MC dose-to-medium.
        MC_wat_dpp(:,:,:,b)=dose2water(MC_file,phant);

        % Get AAA dose for each beam and interpolate onto the MC grid.
        RD_file=[patient_dir '/RD.' patient '.' plan '.' beams{b} '.dcm'];
        if exist(RD_file,'file')
            AAA_wat(:,:,:,b)=interp3(X_AAA,Y_AAA,Z_AAA,...
                double(squeeze(dicomread(RD_file)))*...
                RD_data.DoseGridScaling,X',Y,Z,'linear',0);
        end

        % sum MC dose per particle and uncertainty
        MC_med_dpp(:,:,:,end)=MC_med_dpp(:,:,:,end)+MC_med_dpp(:,:,:,b);
        MC_wat_dpp(:,:,:,end)=MC_wat_dpp(:,:,:,end)+MC_wat_dpp(:,:,:,b);
        MC_unc(:,:,:,end)=MC_unc(:,:,:,end)+MC_unc(:,:,:,b).^2;
        AAA_wat(:,:,:,end)=AAA_wat(:,:,:,end)+AAA_wat(:,:,:,b);
    end
    MC_unc(:,:,:,end)=sqrt(MC_unc(:,:,:,end));

    % save arrays for faster plotting
    save(save_file,'X','Y','Z','beams','iso','MC_med_dpp',...
        'MC_wat_dpp','MC_unc','AAA_wat')
else
```

```
    load(save_file)
end

%==============================================================================
%              MC ABSOLUTE DOSE CALIBRATION
%==============================================================================

% Some calibration constants specific to the linac being modeled and the
% normalization used by the base algorithm e.g. AAA.
chamb_dpp_cal=9.785e-16; % from 10x10 calibration run (sigma = 0.2%)
phantom_dpp_cal=1.1594e-16; % from 10x10 water phantom run (sigma = 0.17%)
MU_cal=0.01; % (Gy/MU)

% Use the monitor chamber dose for absolute calibration
if cal_method==1;
    cal_file=[patient_dir '/' linac '.' patient '.' plan '.' beams{b} '.egslst'];
    cal_lines=textscan(fopen(cal_file),'%s','Delimiter','\n');
    rec_1=find(~cellfun(@isempty,strfind(cal_lines{1},'DOSE RESULTS')),1);
    chamb_dpp_plan=sscanf(cal_lines{1}{rec_1+18},...
        '%*d %*f %f+/- %*f %% %*f+/- %*f %%');
    N_part=chamb_dpp_cal/chamb_dpp_plan*MU_cal/phantom_dpp_cal*MU{1,b};

% Or take an average value of MC dpp and AAA dose in a region surrounding
% the isocenter to calibrate the MC results
elseif cal_method==2;
    win=1;
    norm_j=(X>=iso(1)-win and X<=iso(1)+win);
    norm_i=(Y>=iso(2)-win and Y<=iso(2)+win);
    norm_k=(Z>=iso(3)-win and Z<=iso(3)+win);
    N_part=sum(sum(sum(AAA_wat(norm_i,norm_j,norm_k,end))))/...
        sum(sum(sum(MC_wat_dpp(norm_i,norm_j,norm_k,end))));
end
MC_med=MC_med_dpp*N_part;
MC_wat=MC_wat_dpp*N_part;

%==============================================================================
%              COMPARISON DEFINITION and VIEW SWITCHING
%==============================================================================

% find index of selected beam
b_sel=strcmp(beams,beam);

% define base dose
if strcmp(base_type,'MC_wat')
    base_dose=MC_wat(:,:,:,b_sel);
    base_name='MC_{wat}';
elseif strcmp(base_type,'AAA_wat')
    base_dose=AAA_wat(:,:,:,b_sel);
    base_name='AAA_{wat}';
end

% define comparison dose
if strcmp(comp_type,'MC_med')
        comp_dose=MC_med(:,:,:,b_sel);
        comp_name='MC_{med}';
elseif strcmp(comp_type,'MC_wat')
        comp_dose=MC_wat(:,:,:,b_sel);
        comp_name='MC_{wat}';
end

% Extract the MC uncertainty for the current beam and and build an
% uncertainty mask
```

```
MC_unc=MC_unc(:,:,:,end);
unc_mask=zeros(size(MC_unc));
unc_mask(MC_unc*100<unc_mask_lim)=1;

% Apply the uncertainty mask to the dose arrays so that imaging and
% comparisons are easier to view. This will effectively zeros the dose
% arrays in regions of high uncertainty.
comp_dose=comp_dose.*unc_mask;
base_dose=base_dose.*unc_mask;

% Get the ED array.
[ED,~,~,~,~,~]=egsphant_read([patient_dir '/' phant '.' patient '.egsphant']);

% Permute the grid, dose, uncertainty, and ED arrays to the user selected
% view.
dim=[1 2 3];
axis_view='ij';
x_lab='x axis (cm)';
y_lab='y axis (cm)';
k_iso_dim=3;
if ~strcmp(view_type,'transverse')
    switch view_type
        case 'coronal'
            Y_temp=Y;
            Y=Z;
            Z=Y_temp;
            dim=[3 2 1];
            axis_view='xy';
            x_lab='x axis (cm)';
            y_lab='z axis (cm)';
            i_iso_dim=3;
            j_iso_dim=1;
            k_iso_dim=2;
        case 'sagital'
            X_temp=X;
            X=Z;
            Z=X_temp;
            dim=[1 3 2];
            axis_view='ij';
            x_lab='z axis (cm)';
            y_lab='y axis (cm)';
            i_iso_dim=2;
            j_iso_dim=3;
            k_iso_dim=1;
    end
    ED=permute(ED,dim);
    base_dose=permute(base_dose,dim);
    comp_dose=permute(comp_dose,dim);
    MC_unc=permute(MC_unc,dim);
end

%==============================================================================
%                     1,2 and 3D IMAGING and COMPARISONS
%==============================================================================

% Normalize and mask the electron density array. This essentially zeros the
% electron density in air.
ED_mask=zeros(size(ED));
ED_mask(ED>ED_lim)=1;
ED_norm=ED/max(max(max(ED))).*ED_mask;

% Define the bit depth for imaging
depth=256;
```

```
bins=4;
ticks=1:(depth-1)/bins:depth;

% Calculate dose to integer scaling and tick labels.
max_dose=max([max(max(max(comp_dose))) max(max(max(base_dose)))]);
fac=10^floor(log10(max_dose));
dose_lim=ceil(max_dose/fac)*fac;
dose_scaling=(depth)/dose_lim;
dose_tick_labels=(0:dose_lim/bins:dose_lim);

% Calculate uncertainty to integer scaling, and tick labels.
unc_scaling=(depth-1)/unc_lim;
unc_tick_labels=(0:unc_lim/bins:unc_lim);

%============================================================================
% PLANAR IMAGING and GAMMA COMPARISON
if strcmp(image_mode,'planar')

    % Extract the actual isocenter dose planes for the current view and
    % calculate the gamma map.
    base_plane=interp3(X,Y,Z,base_dose,X,Y,iso(k_iso_dim));
    comp_plane=interp3(X,Y,Z,comp_dose,X,Y,iso(k_iso_dim));
    gamma_plane=gamma_calc(base_plane,comp_plane,res,p_lim,d_lim);

    % Extract the ED plane and convert to uint8 rgb for imaging.
    ED_plane=interp3(X,Y,Z,ED_norm,X,Y,iso(k_iso_dim));
    ED_plane_rgb=ind2rgb(uint8(ED_plane*(depth-1)),gray(depth));

    % Figure setup.
    close all
    fig_adjust=[1 1 0.8 1];
    set(0,'defaultAxesFontSize',12)
    set(0,'defaultTextFontSize',12)
    plane_fig=figure('Units','inches','OuterPosition',fig_size.*fig_adjust);
    axes('Position',[0 0 1 1])
    axis off

    % Display the base dose in the top left frame
    axes('position',[0.1 0.65 0.35 0.35])
    base_plane_rgb=ind2rgb(uint8(base_plane*dose_scaling),jet(depth));
    image(X,Y,0.5*(ED_plane_rgb+base_plane_rgb))
    text(0.02,0.85,base_name,'HorizontalAlignment','left',...
        'Color','w','Units','normalized')
    xlabel(x_lab)
    ylabel(y_lab)
    axis(gca,axis_view,'image')

    % Display the comparison dose in the top left frame
    axes('position',[0.5 0.65 0.35 0.35])
    comp_plane_rgb=ind2rgb(uint8(comp_plane*dose_scaling),jet(depth));
    image(X,Y,0.5*(ED_plane_rgb+comp_plane_rgb))
    text(0.02,0.85,comp_name,'HorizontalAlignment','left',...
        'Color','w','Units','normalized')
    xlabel(x_lab)
    set(gca,'YTickLabels','')
    axis(gca,axis_view,'image')

    % Display the color bar in separate axes so images are the same size.
    axes('position',[0.6 0.65 0.35 0.3])
    c=colorbar('YTick',ticks,'YTickLabel',dose_tick_labels);
    ylabel(c,'Dose (Gy)')
    axis off
    % Display the gamma comparison in accross the entire lower region of
    % the figure.
    axes('position',[0.1 0.07 0.85 0.55])
    gamma_plane_rgb=ind2rgb(uint8(gamma_plane*(depth-1)),jet(depth));
    image(X,Y,0.5*(ED_plane_rgb+gamma_plane_rgb))
    text(0.02,0.90,{'MC vs. film';'\gamma comparison'},...
        'HorizontalAlignment','left',...
        'Color','w','Units','normalized')
    xlabel(x_lab)
    ylabel(y_lab)
    colormap(jet(depth))
    c=colorbar('YTick',ticks,'YTickLabel',(0:1/bins:1));
    ylabel(c,['\gamma index (' num2str(p_lim) '%/' num2str(d_lim*10) 'mm)'])
    axis(gca,axis_view,'image')

    % Print the plane comparsion figures to file.
    print_file=['/home/alex/Projects/Documents/Dissertation/figs/'...
        patient '.' plan '.' beam '.plane'];
    export_fig(print_file,'-pdf','-nocrop',plane_fig)

%===============================================================================
% PROFILE IMAGING and GAMMA COMPARISON
elseif strcmp(image_mode,'profile')

    % Extract the base and comparison planes in order to calculate their
    % gamma map. Then extract the 6 profiles from these planes.
    base_plane=interp3(X,Y,Z,base_dose,X,Y,iso(k_iso_dim));
    comp_plane=interp3(X,Y,Z,comp_dose,X,Y,iso(k_iso_dim));
    base_xprof=interp2(X,Y,base_plane,X,iso(i_iso_dim))';
    base_yprof=interp2(X,Y,base_plane,iso(j_iso_dim),Y);
    comp_xprof=interp2(X,Y,comp_plane,X,iso(i_iso_dim))';
    comp_yprof=interp2(X,Y,comp_plane,iso(j_iso_dim),Y);
    gamma_plane=gamma_calc(base_plane,comp_plane,res,p_lim,d_lim);
    gamma_2d_xprof=interp2(X,Y,gamma_plane,X,iso(i_iso_dim));
    gamma_2d_yprof=interp2(X,Y,gamma_plane,iso(j_iso_dim),Y);

    % Calculate 1D gamma values for comparison with 2D.
    gamma_1d_xprof=gamma_calc(base_xprof,comp_xprof,res,p_lim,d_lim);
    gamma_1d_yprof=gamma_calc(base_yprof,comp_yprof,res,p_lim,d_lim);

    % Figure setup.
    close all
    prof_fig=figure('Units','inches','OuterPosition',fig_size);
    dose_lims=1.2*[0 1.1*max([max(base_xprof) max(base_yprof)])];
    coord={'x','y','z'};

    % Display x profile in the top left frame.
    axes('Position',[0.1 0.4 0.39 0.5]);
    x_lims=[X(1) X(end)];
    plot(X,base_xprof,'g');
    hold on
    plot(X,comp_xprof,'r');
    hold off
    set(gca,'XLim',x_lims,'YLim',dose_lims,'XTickLabels','')
    ylabel('Dose (Gy)')
    text(0.07,0.93,{[coord{k_iso_dim} ' = ' num2str(iso(k_iso_dim)) ' cm'];...
        [coord{i_iso_dim} ' = ' num2str(iso(i_iso_dim)) ' cm']},...
        'Units','normalized','HorizontalAlignment','left')
    text(0.93,0.93,{view_type;'plane'},...
        'Units','normalized','HorizontalAlignment','right')
    % Display gamma x profiles in the lower left frame
    axes('Position',[0.1 0.1 0.39 0.28]);
    plot(X,gamma_1d_xprof,'-k')
    hold on
    plot(X,gamma_2d_xprof,'-k','LineWidth',2)
    hold off
    set(gca,'XLim',x_lims,'YLim',[0 1])
    xlabel(x_lab)
    ylabel(['MC-film \gamma (' num2str(p_lim,'%0.f') '%/'...
        num2str(d_lim*10) 'mm)'])

    % Display y profiles in the top right frame.
    axes('Position',[0.51 0.4 0.39 0.5]);
    y_lims=[Y(1) Y(end)];
    g=plot(Y,base_yprof,'g');
    hold on
    r=plot(Y,comp_yprof,'r');
    hold off
    set(gca,'XLim',y_lims,'YLim',dose_lims,'YTickLabels','','XTickLabels','')
    text(0.07,0.93,{[coord{k_iso_dim} ' = ' num2str(iso(k_iso_dim)) ' cm'];...
        [coord{j_iso_dim} ' = ' num2str(iso(j_iso_dim)) ' cm']},...
        'Units','normalized','HorizontalAlignment','left')
    legend([g r],{base_name,comp_name},'Location','northeast')

    % Display gamma y profiles in the lower right frame
    axes('Position',[0.51 0.1 0.39 0.28]);
    g1=plot(Y,gamma_1d_yprof,'-k');
    hold on
    g2=plot(Y,gamma_2d_yprof,'-k','LineWidth',2);
    hold off
    set(gca,'XLim',y_lims,'YLim',[0 1],'YTickLabels','')
    xlabel(y_lab)
    legend([g1 g2],{'1D Gamma','2D Gamma'},'Location','northeast')

    % Print the profile comarison figure to file.
    print_file=['/home/alex/Projects/Documents/Dissertation/figs/'...
        patient '.' plan '.' beam '.prof'];
    export_fig(print_file,'-pdf','-nocrop',prof_fig)

%================================================================================
% 3D IMAGING
elseif strcmp(image_mode,'3D')

    % Calculate the selected deviation, scaling, and tick labels.
    if strcmp(dev_type,'percent')
        dev_lim=pcnt_dev_lim;
        dev_map=(comp_dose-base_dose)./base_dose*100;
        dev_map(isnan(dev_map))=0;
        dev_label='Percent deviation (%)';
        dev_tick_labels=(-dev_lim:2*dev_lim/bins:dev_lim);
        dev_unit='(%)';
    elseif strcmp(dev_type,'absolute')
        dev_lim=abs_dev_lim;
        dev_map=base_dose-comp_dose;
        dev_label='Absolute deviation (Gy)';
        dev_title=[comp_name '-' base_name];
        dev_unit='(Gy)';
    end
    dev_scaling=(depth-1)/(2*dev_lim);

    % Scale all arrays and convert to uint8.
    ED_int=uint8(ED_norm*(depth-1));
    base_dose_int=uint8(base_dose*dose_scaling);
    comp_dose_int=uint8(comp_dose*dose_scaling);
    MC_unc_int=uint8(MC_unc*unc_scaling);
    dev_dose_int=uint8(depth/2+dev_map*dev_scaling);

    % Figure setup.
    figure('Units','inches','OuterPosition',fig_size);
    axes('Position',[0 0 1 1])
    axis off
    t1=text(0.1,0.05,'');
    text(0.5,0.95,['Plan comparison for ' patient '.' plan '.' beams{b_sel}],...
        'Interpreter','none','HorizontalAlignment','Center',...
        'FontSize',20,'FontWeight','bold')

    H=[axes('position',[0.05 0.57 0.43 0.3]);
        axes('position',[0.05 0.15 0.43 0.3]);
        axes('position',[0.55 0.57 0.43 0.3]);
        axes('position',[0.55 0.15 0.43 0.3])];
    im_int={base_dose_int,comp_dose_int,MC_unc_int,dev_dose_int};
    image_type={comp_name,base_name,'MC uncertainty',dev_title};
    A=zeros(size(H));
    cbar_tick_labels={dose_tick_labels,dose_tick_labels,unc_tick_labels,...
        dev_tick_labels};
    cbar_label={'Dose (Gy)','Dose (Gy)','Stat. Unc. (%)',['Deviation ' dev_unit]};

    % Calculate a limited view span as moving through images is a bit slow.
    view_span=0.75*Z(end)/2*[-1 1];
    K=find(Z>=iso(k_iso_dim)+view_span(1) and Z<=iso(k_iso_dim)+view_span(2))';

    % Display the selected slices.
    for k=K

        % Extract the ED plane for the current slice and convert to rgb. Only
        % one plane at a time can be converted to rgb so conversion occurs
        % inside the viewing loop.
        ED_plane_rgb=ind2rgb(ED_int(:,:,k),gray(depth));

        % Plot each plane. Comparison dose in upper left, base dose in
        % lower left, MC uncertainty in upper right, and deviation in lower
        % right.
        for h=1:numel(H)
            rgb_image=ind2rgb(im_int{h}(:,:,k),jet(depth));
            image(X,Y,0.5*(ED_image_rgb+rgb_image),'Parent',H(h))
            title(H(h),image_type{h},'FontSize',14,'FontWeight','bold')
            xlabel(H(h),x_lab)
            ylabel(H(h),y_lab)
            colormap(jet(depth))
            c1=colorbar('peer',H(h),'YTick',ticks,'YTickLabel',...
                cbar_tick_labels{h});
            ylabel(c1,cbar_label{h})
        end
        axis(H,axis_view,'image')
        set(t1,'String',{['Slice number: ' num2str(k)];...
            ['Slice location: ' sprintf('%6.2f cm',Z(k))]})

        % pause setting
        if pause_val==0
            pause
        else
            pause(pause_val);
        end
    end
end
```

dose2water.m

```
% This function uses the values provided by Seibers et al. (1999) to
% calculate dose to water from dose t0 medium.

function [MC_dpp_dtw]=dose2water(MC_file,phant)

patient_dir=regexprep(MC_file,'(.*)/.*$','$1');
patient=regexprep(patient_dir,'.*/(.*)$','$1');

% water to medium average stopping power ratios
air=1.117; % media index 1
lung=0.999; % media index 2
tissue=1.010; % media index 3
% soft_bone=1.035;
cort_bone=1.116; % media index 4

[~,media,~,~,~,~]=egsphant_read([patient_dir '/' phant '.' patient '.egsphant']);
[MC_dpp,~,X,Y,Z]=MC_read(MC_file);

SP_factor=zeros(numel(Y),numel(X),numel(Z));

SP_factor(media==1)=air;
SP_factor(media==2)=lung;
SP_factor(media==3)=tissue;
SP_factor(media==4)=cort_bone;

MC_dpp_dtw=MC_dpp.*SP_factor;
```

Appendix D Error Prediction Code

Error Index Calculation Scripts

error_predict.m

```
% Parent script for calculating error indices. This starts by calculating
% the LDI as this is control point independent. It then loops through each
% control point and calls the field size index function (FSI_calc) and the
% heterogeneity scatter index functon (HSI_calc) for each control point.
% Note that output_analyze must be run first so the dose arrays are
% available in the patient .mat file.

close all
clearvars

% Select patient, plan, and beam to calculate error maps for.
patient='IMAT_HN1';
patient_list={'IMAT_L1','IMAT_HN1','IMAT_E1','IMAT_P1'};
plan_list={'RA_2_ARC','HN_RA-1','RA_Esophagus','Prostate'};
plan=plan_list{strcmp(patient_list,patient)};
beam='total'; % set to 'total' for total dose
phant='CT_phant';

% Set to one to calculate new error maps, otherwise load them from file.
recalc=1;

% Define error map scaling parameters.
FSI_scale_factor=0.11;
HSI_wt=1;
HSI_kern_len=4; % (cm) width at zero depth;
HSI_kern_sigma=0.5; % (cm) width at zero depth;

% Define imaging and analysis parameters.
image_mode='3D'; % Define the type of imaging {'profile','planar','3D'}.
error_type='FSI'; % Type of error to view {'LDI','FSI','HSI'}.
view_type='transverse'; % Select the type of planes to view.
view_span_factor=0.75; % Factor by which to reduce the display window.
unc_mask_lim=5; % Set the allowed percent uncertainty for the uncertainty mask.
pause_val=0.1; % Set to zero for user stepping.
frac_dev_lim=0.1; % Set the deviation limits.
base_type='MC_wat';
comp_type='AAA_wat';

% Define fig size for the current machine.
global fig_size

% Some constant directories and names.
recalc_dir='/home/alex/Projects/Plan Recalculation/Patients';
patient_dir=[recalc_dir '/' patient];
save_file=[patient_dir '/' patient '.mat'];

%================================================================================
%                    DATA EXTRACTION
```

```
%================================================================================

% Load the grid (X,Y,Z), MC_wat, MC_med, MC_unc, and AAA_wat arrays which
% where all extracted during a previous run of output_analyze.m.
load(save_file)

% Extact the treatment plan parameters.
RP_file=[patient_dir '/RP.' patient '.' plan '.dcm'];
[beam_lab,JP,iso,TA,CA,MU,MLC,GA]=plan_pars(RP_file);
beams={beam_lab{2,:} 'total'}';

% Find index of selected beam.
b_sel=find(strcmp(beams,beam)==1);

% Get ED array.
[ED,~,~,~,~,vox]=egsphant_read([patient_dir '/' phant '.' patient '.egsphant']);

%================================================================================
%                    ERROR MAP CALCULATION
%================================================================================

% If recalculating the error maps...
if recalc==1

    %============================================================================
    % LDI Calculation

    % This models the deviation (MC/AAA) of dose deposition in low density
    % open field regions. The model is derived from AAA to MC comparisons
    % and it is calculated in LDI_validation.m. The model is a sum of a
    % linear and an exponential and takes the following five paramters. It
    % is applicable to ED<1 only.

    % Parameterize the model.
    ED_0=0.0464; % g/cm^3
    LDI_0=0.9182; % LDI intercept for the linear part
    LDI_1=0.0867; % cm^3/g slope of linear part
    LDI_alpha=-0.0513; % exponential scaling
    LDI_beta=-66.1059; % cm^3/g ED deviation scaling

    % Calculate the LDI.
    ED_low=ED;
    ED_low(ED>1)=0;
    LDI_map=(LDI_0+ED_low*LDI_1+LDI_alpha*exp(LDI_beta*(ED_low-ED_0)))-1;

    %============================================================================
    % FSI and HSI Calculations

    % Crop ED array and interpolate onto a symmetric grid centered on the
    % isocenter so it can be rotated. Reset resolution to 0.25 cm^3 so that
    % there are two pixels per target/isocenter leaf.
    x_max=max([iso(1)-X(1) X(end)-iso(1)]);
    y_max=max([iso(2)-Y(1) Y(end)-iso(2)]);
    z_max=max([iso(3)-Z(1) Z(end)-iso(3)]);
    vox_calc=0.25;
    X_sym=([-fliplr(vox_calc/2:vox_calc:x_max) vox_calc/2:vox_calc:x_max]+iso(1))';
    Y_sym=([-fliplr(vox_calc/2:vox_calc:y_max) vox_calc/2:vox_calc:y_max]+iso(2))';
    Z_sym=([-fliplr(vox_calc/2:vox_calc:z_max) vox_calc/2:vox_calc:z_max]+iso(3))';
    I_sym=numel(Y_sym);
    J_sym=numel(X_sym);
```

```
K_sym=numel(Z_sym);
ED_sym=interp3(X,Y,Z,ED,X_sym',Y_sym,Z_sym,'linear',0);

% Build accumulation arrays.
FSI_map=zeros(numel(Y),numel(X),numel(Z),numel(beams));
HSI_map=zeros(numel(Y),numel(X),numel(Z),numel(beams));

% For each beam
for b=1:numel(beams)-1

    % For each control point...
    MU_last=0;
    C=numel(MU{2,b});
    MC_beam_wt=MU{1,b}/sum([MU{1,:}]);
    for c=1:C

        % Display a progress Message
        disp(['Calculating error maps for beam ' beam ', control point '...
            num2str(c) ' of ' num2str(C) '...'])

        % Calculate MU weight.
        MU_CP_wt=(MU{2,b}(c)-MU_last);
        MU_last=MU{2,b}(c);

        %===================================================================
        % FSI Calculation

        % Calclulate the FSI map for the current control point.
        [FSI_CP_map_calc,beam_map]=FSI_calc(X_sym,Y_sym,Z_sym,iso,...
            MLC{b}(c,:),CA{b},FSI_scale_factor);

        % Rotate the FSI map to coincide with the beam directions
        % (gantry position), and interpolate back onto original grid.
        % Here the negative sign forces a clockwise rotation to match
        % the gantry's direction of positive rotation.
        FSI_CP_map=interp3(X_sym,Y_sym,Z_sym,...
            imrotate(FSI_CP_map_calc,-GA{b}(c),'crop'),X',Y,Z,'linear',0);

        % Add control point FSI map, weighted by MU fraction, to the
        % accumulator array.
        FSI_map(:,:,:,b)=FSI_map(:,:,:,b)+MU_CP_wt*FSI_CP_map;

        %===================================================================
        % FSI Calculation (not currently functional)

        % Calculate the HSI map for the current control point.
        % HSI_CP_map_calc=HSI_calc(ED_sym,beam_map,X_sym,Y_sym,Z_sym,iso,...
        %     GA{b}(c),HSI_wt,HSI_kern_len,HSI_kern_sigma);

        % Rotate the FSI map to coincide with the beam directions
        % (gantry position), and interpolate back onto original grid.
        % Here the negative sign forces a clockwise rotation to match
        % the gantry's direction of positive rotation.
        %HSI_CP_map=interp3(X_sym,Y_sym,Z_sym,...
        %    imrotate(HSI_CP_map_calc,-GA{b}(c),'crop'),X',Y,Z);

        % Add control point HSI map, weighted by MU fraction, to the
        % accumulator array.
        %HSI_map(:,:,:,b)=HSI_map(:,:,:,b)+MU_CP_wt*HSI_CP_map;
```

```
            %==================================================================
        end

        % Accumulate error maps into the last index of the beam dimension.
        FSI_map(:,:,:,end)=FSI_map(:,:,:,end)+MC_beam_wt*FSI_map(:,:,:,b);
        HSI_map(:,:,:,end)=HSI_map(:,:,:,end)+MC_beam_wt*HSI_map(:,:,:,b);
    end

    % Save error maps.
    save(save_file,'FSI_map','HSI_map','LDI_map','-append');
end
```

FSI_calc.m

```
% This script calculates the field size error index (FSI).  The
% outputted FSI map will be in the symmetric calculation grid
% as well as aligned with gantry zero so it will have to be interpolated
% and rotated accordingly in the main program.

function [FSI_map,beam_map]=FSI_calc(X_sym,Y_sym,Z_sym,iso,MLC,CA,FSI_scale_factor)

% Build binary image of MLC at iso  Eclipse lists leaves in the y positive
% (Y1->Y2) direction, first from bank B (X1), then from bank A (X2).
vox=Y_sym(2)-Y_sym(1);
MLC_lim=20;
X_MLC=((-MLC_lim+vox/2):vox:(MLC_lim-vox/2))';
Y_MLC=X_MLC;
MLC_bin=zeros(numel(Y_MLC),numel(X_MLC));
L=numel(MLC);
X1_pos=MLC(1:L/2)';
X2_pos=MLC(L/2+1:end)';
skip=1/vox;
i=1;
for l=1:L/2
    if l<11 || l>50
        MLC_bin(i:i+skip-1,(X_MLC>X1_pos(l) and X_MLC<X2_pos(l)))=1;
        i=i+skip;
    else
        MLC_bin(i:i+skip/2-1,(X_MLC>X1_pos(l) and X_MLC<X2_pos(l)))=1;
        i=i+skip/2;
    end
end

% Small field effects are identified by regions of high dose gradient which
% will induce volume ageraging artifacts.
[grad_x,grad_z]=gradient(MLC_bin);
MLC_grad=sqrt(grad_x.^2+grad_z.^2);

% This MLC field dose gradient is transformed into a FSI (or
% correction) by scaling up the open regions by the FSI_scale_factor, and
% scaling down the blocked regions by its negative value.
FSI_map_2d=MLC_bin.*MLC_grad*FSI_scale_factor-...
    abs(MLC_bin-1).*MLC_grad*FSI_scale_factor;

% Apply collimator rotation to the MLC and FSI images.  In the BEV, the
% collimator rotates CCW in x-y (high x to high y), imrotate rotates CCW in
% i-j (high j to low i),so the CA must be negated.
FSI_map_2d_rot=imrotate(FSI_map_2d,-CA,'bilinear','crop');
```

```
MLC_bin_2d_rot=imrotate(MLC_bin,-CA,'bilinear','crop');

% Interpolate the binary and gradient MLC images back onto the calc grid
% and then project them along the diverging beam lines onto the
% calc grid.
FSI_map_2d_sym=interp2(X_MLC+iso(1),Y_MLC+iso(3),FSI_map_2d_rot,...
    X_sym',Z_sym,'linear',1);
MLC_bin_2d_sym=interp2(X_MLC+iso(1),Y_MLC+iso(3),MLC_bin_2d_rot,...
    X_sym',Z_sym,'linear',0);
div_factor=(100+(Y_sym-iso(2)))/100;
X_div=div_factor*(X_sym'-iso(1))+iso(1);
Z_div=div_factor*(Z_sym'-iso(3))+iso(3);
FSI_map=zeros(numel(Y_sym),numel(X_sym),numel(Z_sym));
beam_map=zeros(numel(Y_sym),numel(X_sym),numel(Z_sym));
for i=1:numel(Y_sym)
    FSI_map(i,:,:)=interp2(X_div(i,:),Z_div(i,:),...
        FSI_map_2d_sym,X_sym',Z_sym,'linear',1)';
    beam_map(i,:,:)=interp2(X_div(i,:),Z_div(i,:),...
        MLC_bin_2d_sym,X_sym',Z_sym,'linear',0)';
End
```

HSI_calc.m

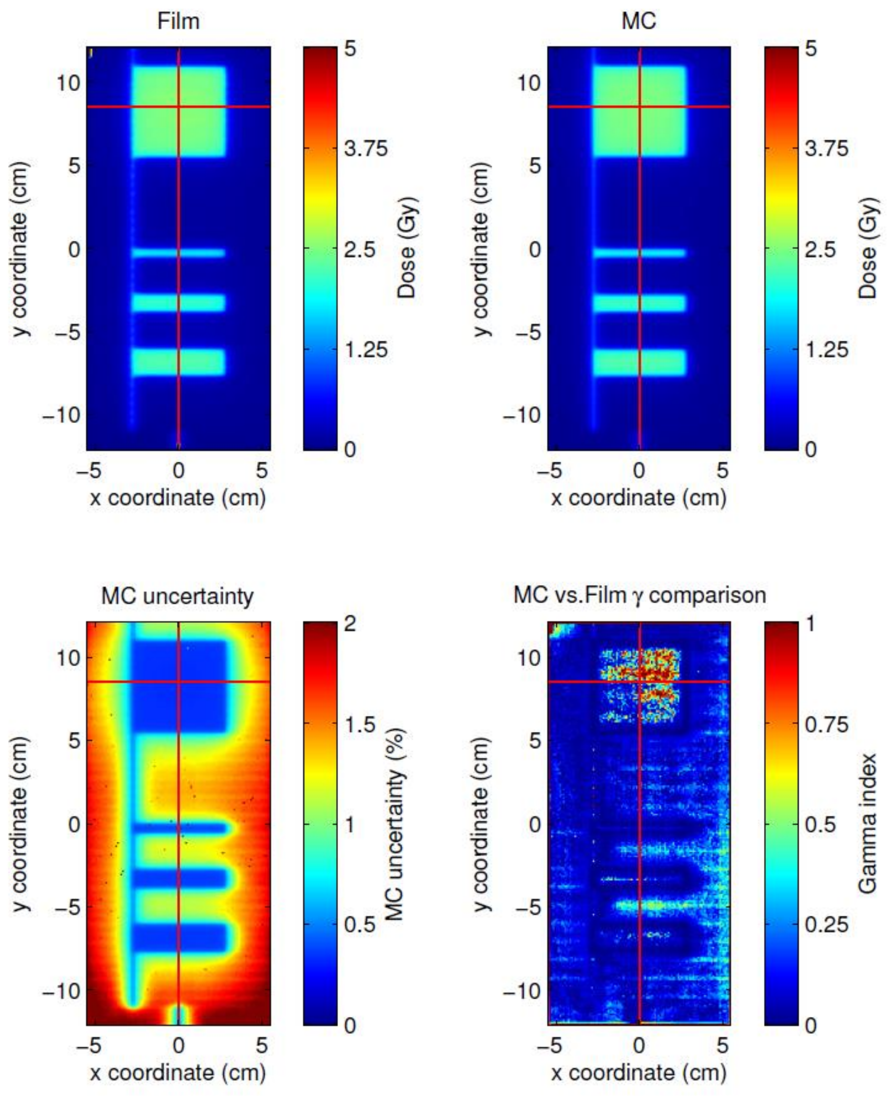
```
% This script calculates the heterogeneous scatter error index (HSI).  The
% outputted HSI map will be in the symmetric calculation grid as well as
% aligned with gantry zero so it will have to be interpolated and rotated
% accordingly in the main program.

function [HSI_map_sym]=HSI_calc(ED_sym,beam_map,X_sym,Y_sym,Z_sym,iso,GA)

% Define calculation parameters
HSI_wt=1;
mu_0=0.6;
sigma_0=0.5;

% Reduce grid resolution and pad grid by the max kernal length on the
% bottom and the max width in the x and z dimensions.
xz_grid_factor=2;
y_grid_factor=2;
y_vox=(Y_sym(2)-Y_sym(1))*y_grid_factor;
xz_vox=(X_sym(2)-X_sym(1))*xz_grid_factor;
max_kern_len=round(1/mu_0/y_vox); % voxels
max_kern_wid=round(2*sigma_0/xz_vox); % voxels
Y_calc=(0:numel(Y_sym)/y_grid_factor+max_kern_len-1)'*y_vox+y_vox/2+iso(2);
X_side=(0:numel(X_sym)/2/xz_grid_factor-1+max_kern_wid)*xz_vox+xz_vox/2;
X_calc=[-fliplr(X_side) X_side]'+iso(1);
Z_side=(0:numel(Z_sym)/2/xz_grid_factor-1+max_kern_wid)*xz_vox+xz_vox/2;
Z_calc=[-fliplr(Z_side) Z_side]'+iso(3);

% Interpolate the ED and beam maps onto the new grid.
ED_calc=interp3(X_sym,Y_sym,Z_sym,ED_sym,X_calc',Y_calc,Z_calc,'linear',0);
% beam_map_calc=interp3(X_sym,Y_sym,Z_sym,beam_map,X_calc',Y_calc,Z_calc,'linear',0);

% Rotate the ED array such that the direction of the beam is perpendicular
% to the x-z plane.  This allows the HSI of the current control point to be
% calculated in a rectilinear grid. Here GA stays positive as this rotation
% moves opposite the gantry's direction of positive rotation.
ED_calc_rot=imrotate(ED_calc,-GA,'crop');

% Calculate the beam divergence factor array based on source to voxel (SVD)
% distance.
[X_calc_dev,Z_calc_dev]=meshgrid(...
    X_calc-iso(1),Y_calc-iso(1)+100,Z_calc-iso(3));
SVD_beamlet=sqrt(X_calc_dev.^2+Y_calc_dev.^2+Z_calc_dev.^2);
beam_div_factor=SVD_beamlet./Y_calc_dev;

% Interpolate the ED array onto the divergent beamlet grid. This results in
% a grid the same size, but each y increment will have a different y and z
% arrays. This essentially skews the ED array such that columns of voxels
% will represent ED values along the divergent rays.
div_factor=(100+(Y_calc-iso(2)))/100;
X_div=div_factor*(X_calc'-iso(1))+iso(1);
Z_div=div_factor*(Z_calc'-iso(3))+iso(3);
ED_calc_div=zeros(size(ED_calc));
temp_div_factor=zeros(size(ED_calc));
for i=1:numel(Y_calc);
    ED_calc_div(i,:,:)=interp2(X_calc,Z_calc,squeeze(ED_calc_rot(i,:,:))',...
        X_div(i,:)',Z_div(i,:),'linear',0)';
end

% temp view
% temp_view(beam_map_calc,X_calc,Y_calc,Z_calc,iso,'transverse')

% Calculate a scatter kernel weight based on the attenuation along each
% beamlet weighted by the beam divergence factor.  Here we assume no
% attenuation at the first y voxel and then use the average of the first
% two voxels to calculate the attenuation in the second voxel, and so on.
rho_beamlet=zeros(size(ED_calc));
rho_beamlet(2:end,:,:)=(ED_calc_div(1:end-1,:,:)+ED_calc_div(2:end,:,:))/2;
t_beamlet=y_vox*beam_div_factor.*cumsum(rho_beamlet,1);
A_wt=exp(-mu_0*t_beamlet).*beam_div_factor.^-2*HSI_wt;

% Calculate an attenuation weighted mu and sigma for the HSI kernel. These
% should eventually be depth dependent.  Note: mu=0.0575; % (cm^2/g) for
% water and Eave=1.5
kern_mu=mu_0./A_wt;
kern_sigma=sigma_0./A_wt;

% temp view A_div=HSI_wt; A=zeros(size(A_div)); for i=1:I_calc
%     A(i,:,:)=interp2(squeeze(X_div(i,:)),squeeze(Z_div(i,:)),...
%         squeeze(A_div(i,:,:))',X_calc',Z_calc,'linear',0)';
% end lims=[min(min(min(A_div))) max(max(max(A_div)))]; close all;
% figure('Units','inches','OuterPosition',[2 2 8 8]); for
%     k=1:K_calc subplot(1,2,1)
%     imshow(A_div(:,:,k),lims,'InitialMagnification','fit') subplot(1,2,2)
%     imshow(A(:,:,k),lims,'InitialMagnification','fit') colorbar
%     pause(0.1)
% end

% Identify the voxels that a) receive beam, b) are not air.
% HSI_min_density=0.01; %g/cm^3
% [I_beam,J_beam,K_beam]=ind2sub(size(beam_map_calc),find(beam_map_calc>0 and...
%     ED_calc_div>HSI_min_density));

% Loop through each beam exposed voxel.
HSI_map_div=ones(size(ED_calc_div));
I_beam=(1:numel(Y_calc)-max_kern_len);
J_beam=(max_kern_wid+1:numel(X_calc)-max_kern_wid);
K_beam=(max_kern_wid+1:numel(Z_calc)-max_kern_wid);
% for v=1:numel(I_beam)
for i=[3]% I_beam
    i
    for j=J_beam
        for k=K_beam
            % Extract values common to the current voxel.
            %               i=I_beam(v);
            %               j=J_beam(v);
            %               k=K_beam(v);

            % Extract the scatter box grid.  For now kernel length is
            I_box=(i:i+min([max_kern_len round(1/mu_0/y_vox)]))';
            xz_div_vox=X_div(I_box(end),2)-X_div(I_box(end),1);
            jk_box_ext=min([max_kern_wid round(2*sigma_0/xz_div_vox)]);
            JK_box=(-jk_box_ext:jk_box_ext);
            J_box=JK_box+j;
            K_box=JK_box+k;

            % Calculate the scatter box grid which is divergent from the
            % scatter point.
            Y_box_dev=Y_calc(I_box)-Y_calc(i);
            box_div_factor=Y_box_dev/Y_box_dev(end);
            X_box_dev=(box_div_factor)*...
                (X_div(I_box(end),J_box)-X_div(I_box(end),j));
            Z_box_dev=(box_div_factor)*...
                (Z_div(I_box(end),K_box)-Z_div(I_box(end),k));
            X_box=X_box_dev+repmat(X_div(I_box,j),1,numel(J_box));
            Z_box=Z_box_dev+repmat(Z_div(I_box,k),1,numel(K_box));

            % Extract the ED scatter box (which is already beam divergent)
            % and calculate the HSI for each xz plane.
            ED_box=ED_calc_div(I_box,J_box,K_box);
            ED_box_div=zeros(size(ED_box));
            ED_box_div_dev=zeros(size(ED_box));
            for i_box=1:numel(I_box)

                % Interpolate the ED box onto the divergent scatter box
                % grid.
                ED_box_div(i_box,:,:)=interp2(X_div(i+i_box-1,J_box),...
                    Z_div(i+i_box-1,K_box),squeeze(ED_box(i_box,:,:))',...
                    X_box(i_box,:)',Z_box(i_box,:),'linear',0)';

                % Subtract the ED at each element in the plane from the CAX
                % value.
                ED_box_div_dev(i_box,:,:)=ED_box_div(i_box,:,:)-...
                    ED_box_div(i_box,jk_box_ext+1,jk_box_ext+1);
            end

            % Calculate the Gaussian factor for the scatter box.  Each ray
            % should have the same Gaussian scaling which is derived from
            % the base plane.
            [G_dev_x,~,G_dev_z]=meshgrid(X_box_dev(end,:),ones(numel(I_box),1),...
                Z_box_dev(end,:));
            G_box_div=exp(-(G_dev_x.^2+G_dev_z.^2)/(2*pi*kern_sigma(i,j,k)^2));

            % Calculate the cumulative difference along each ray
            % (column) of the scatter box.
            rho_ray_dev=cumsum(ED_box_div_dev,1);

            % Calculate the HSI for the scatter box.
            div_vox=y_vox*beam_div_factor(i,j,k);
            HSI_box_div=(A_wt(i,j,k)*G_box_div.*...
                (exp(-kern_mu(i,j,k).*div_vox.*rho_ray_dev)-1)+1);

            % Interpolate the divergerent HSI box back onto the beam
            % divergent calc grid.
```

```
            HSI_box=ones(size(ED_box));
            for i_box=2:numel(I_box)
                HSI_box(i_box,:,:)=interp2(X_box(i_box,:),X_box(i_box,:),...
                    squeeze(HSI_box_div(i_box,:,:))',X_div(i+i_box-1,J_box),...
                    Z_div(i+i_box-1,K_box)','linear',1)';
            end

            % Multiply the HSI box into the HSI map.
            HSI_map_div(I_box,J_box,K_box)=HSI_map_div(I_box,J_box,K_box).*HSI_box;
        end
    end
end

% Interpolate the divergent HSI map back onto the symmetric calculation grid.
HSI_map_calc=ones(size(ED_calc));
for i=1:numel(Y_calc)
    HSI_map_calc(i,:,:)=interp2(X_div(i,:),Z_div(i,:),...
        squeeze(HSI_map_div(i,:,:))',X_calc',Z_calc,'linear',1)';
end

% Interpolate back onto original resolution
HSI_map_sym=interp3(X_calc,Y_calc,Z_calc,HSI_map_calc,X_sym,Y_sym',Z_sym,'linear',1);
```

Appendix E Supplemental Figures

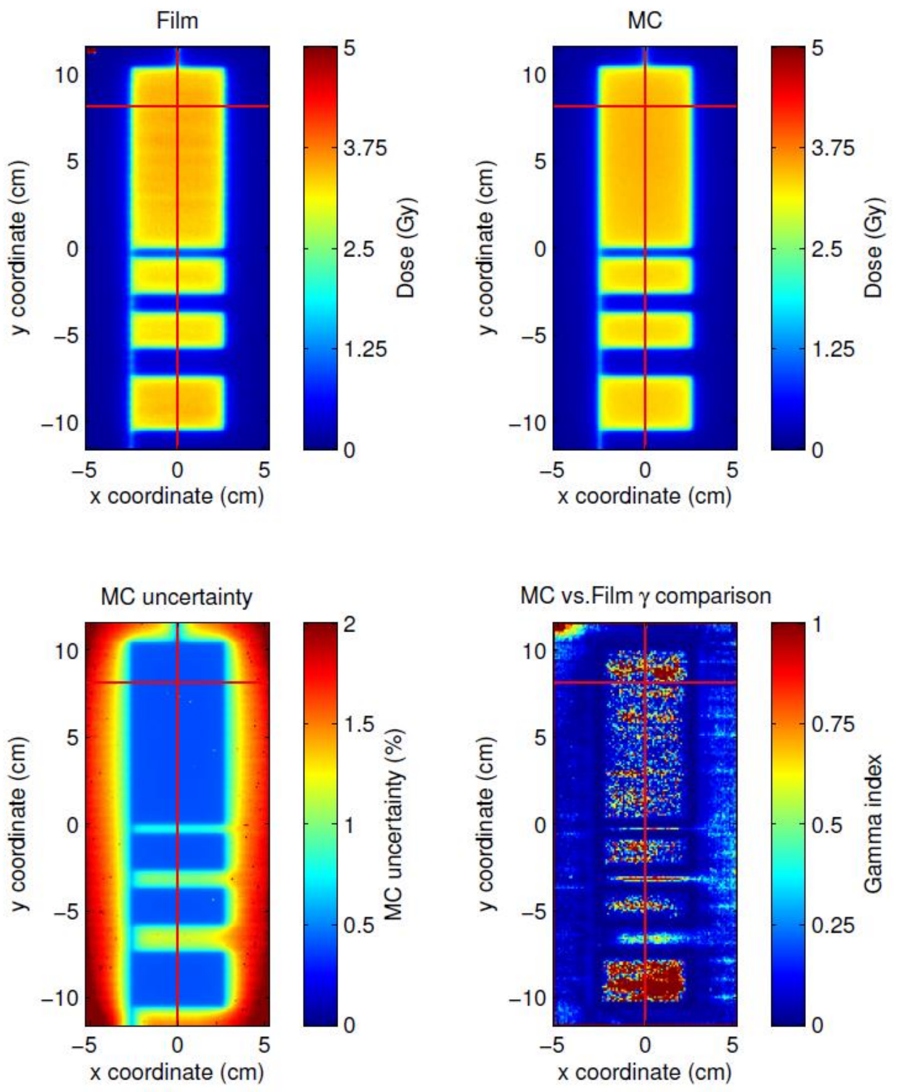
MLC Validation Planar Comparisons

Field 1

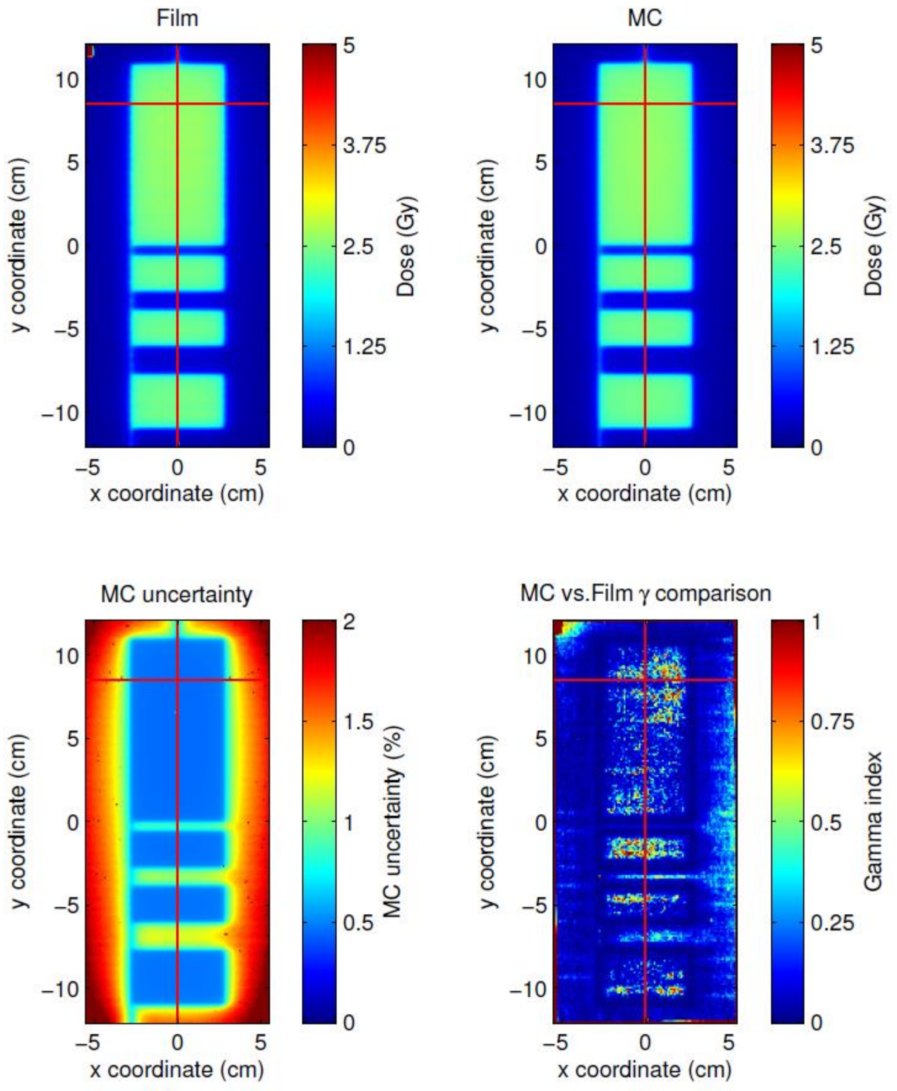


MC dose, MC uncertainty, film dose, and their gamma index (1%/1mm) comparison for field 1, at 10 cm depth.

Field 2

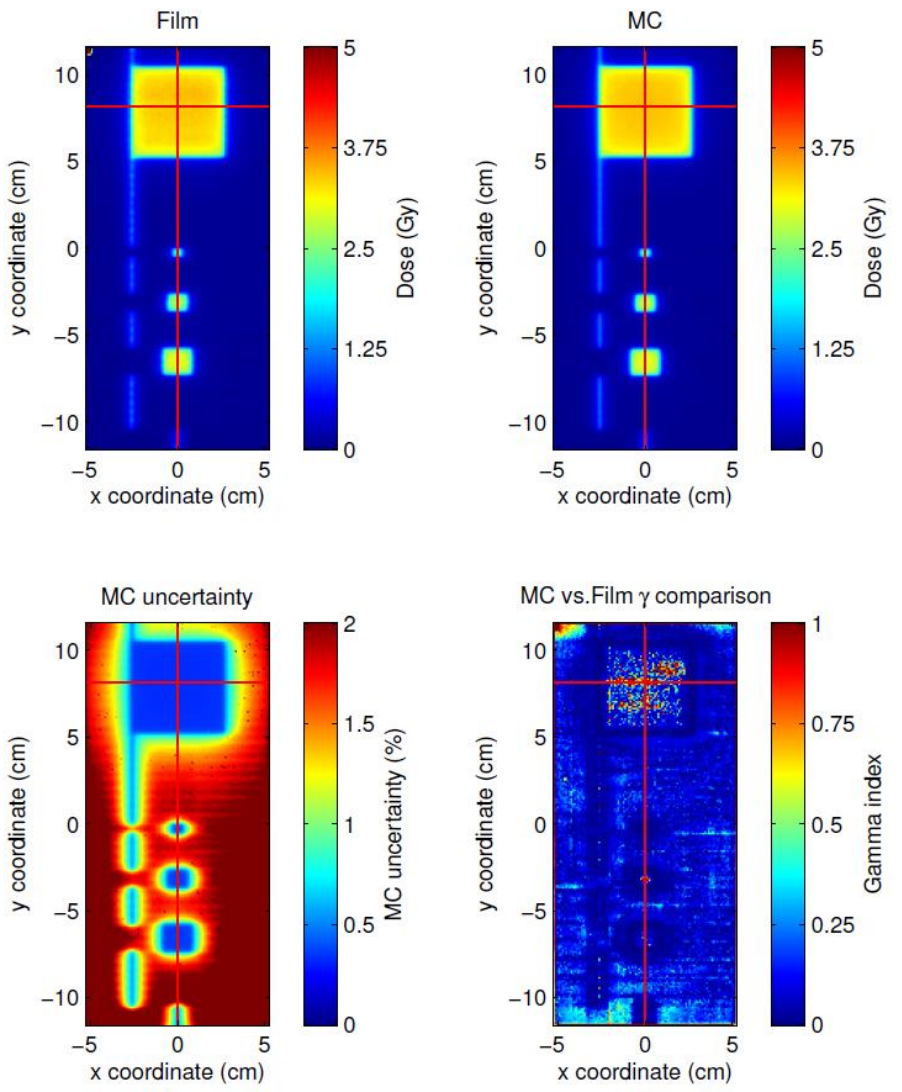


MC dose, MC uncertainty, film dose, and their gamma index (1%/1mm) comparison for field 2, at 5 cm depth.

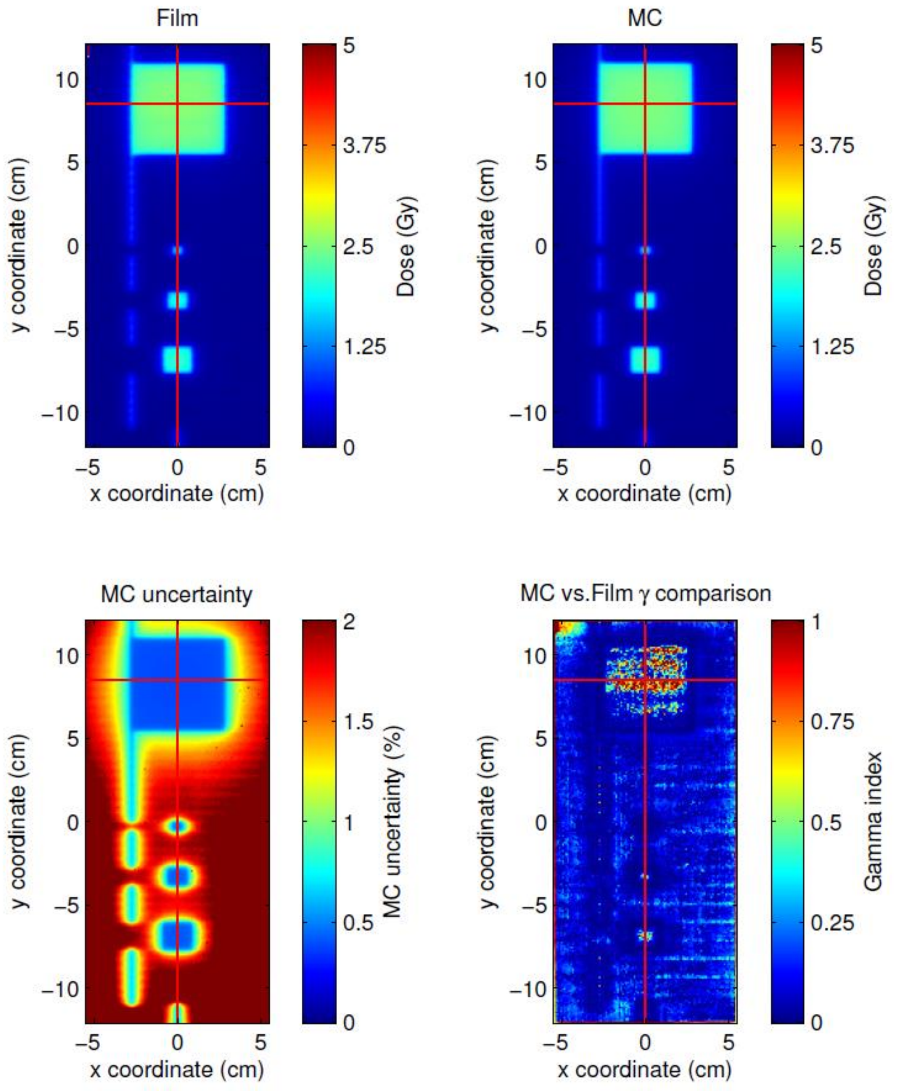


MC dose, MC uncertainty, film dose, and their gamma index (1%/1mm) comparison for field 2, at 10 cm depth.

Field 3

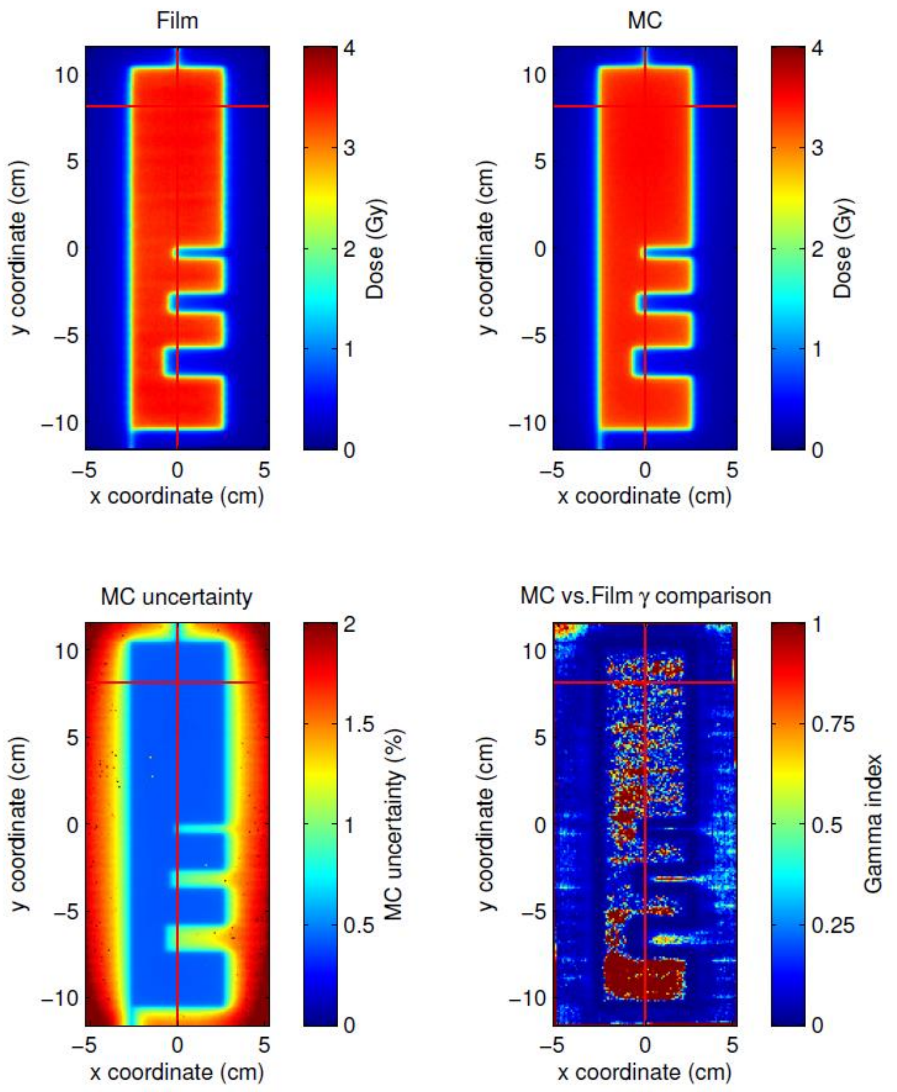


MC dose, MC uncertainty, film dose, and their gamma index (1%/1mm) comparison for field 3, at 5 cm depth.



MC dose, MC uncertainty, film dose, and their gamma index (1%/1mm) comparison for field 3, at 10 cm depth.

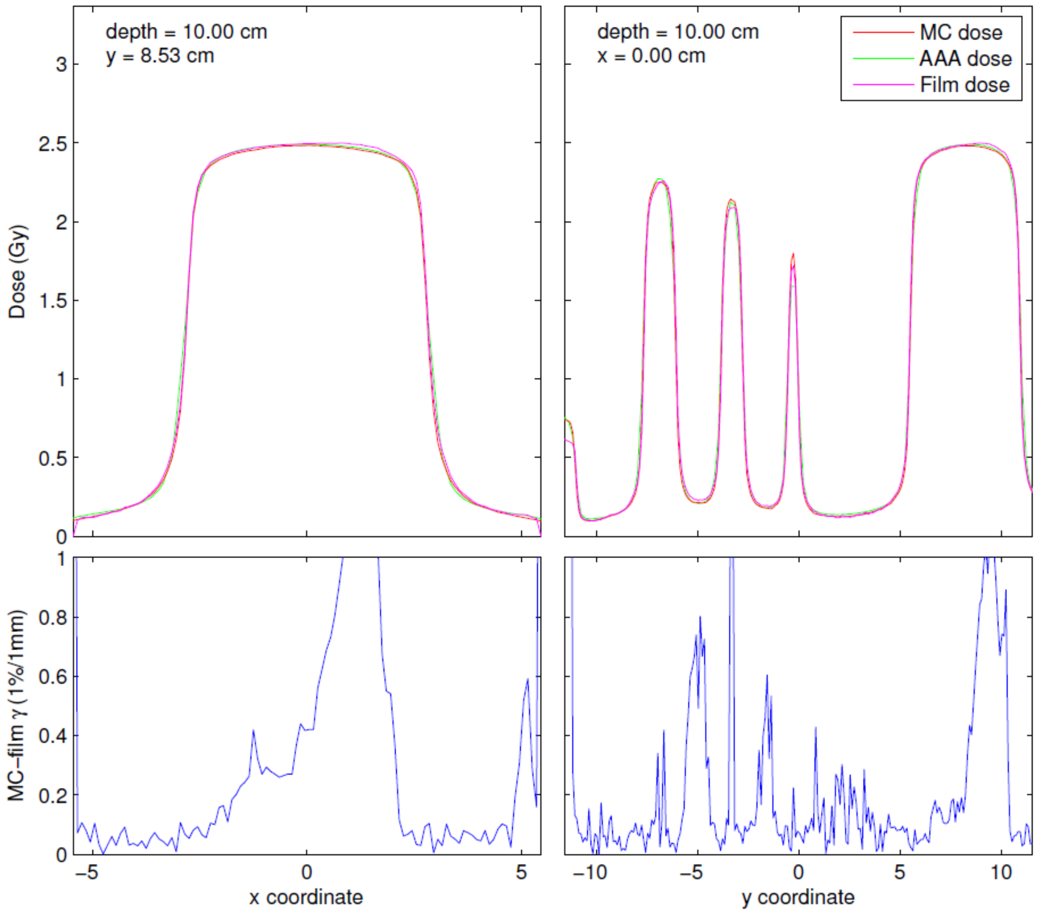
Field 4



MC dose, MC uncertainty, film dose, and their gamma index (1%/1mm) comparison for field 4, at 5 cm depth.

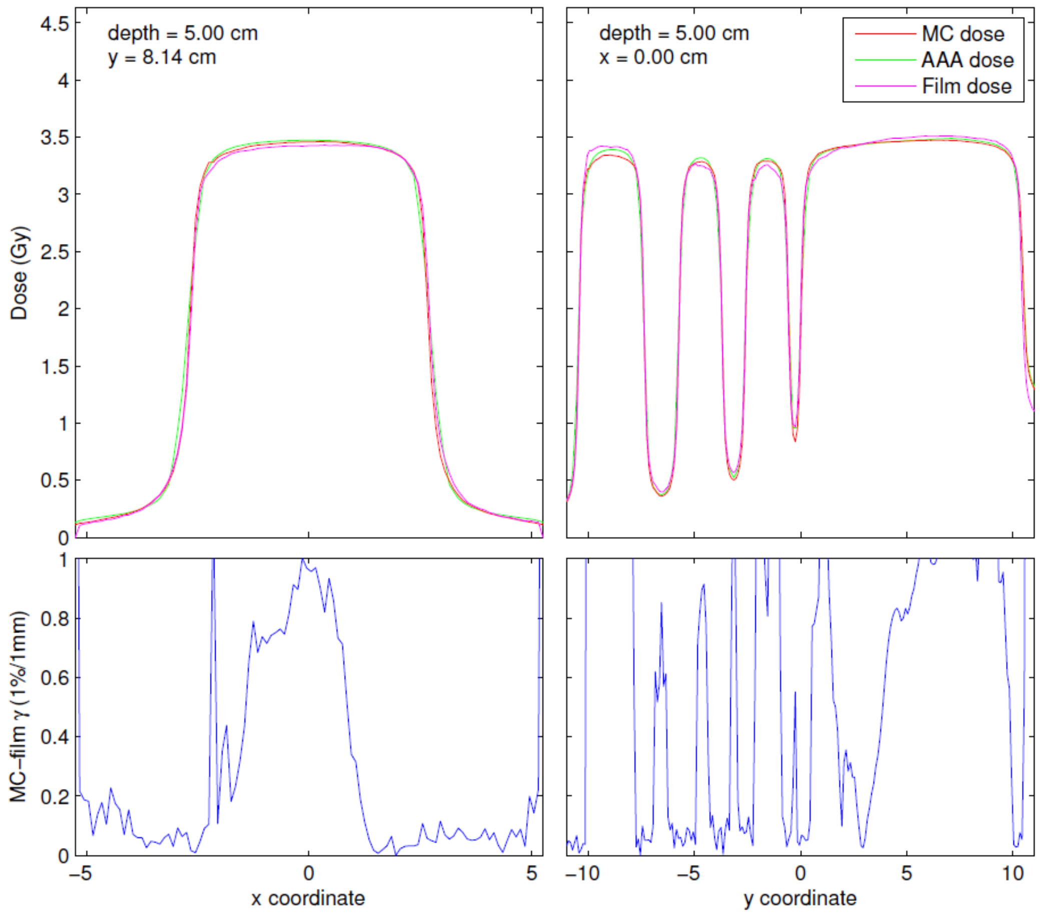
MLC Validation Profile Comparisons

Field 1

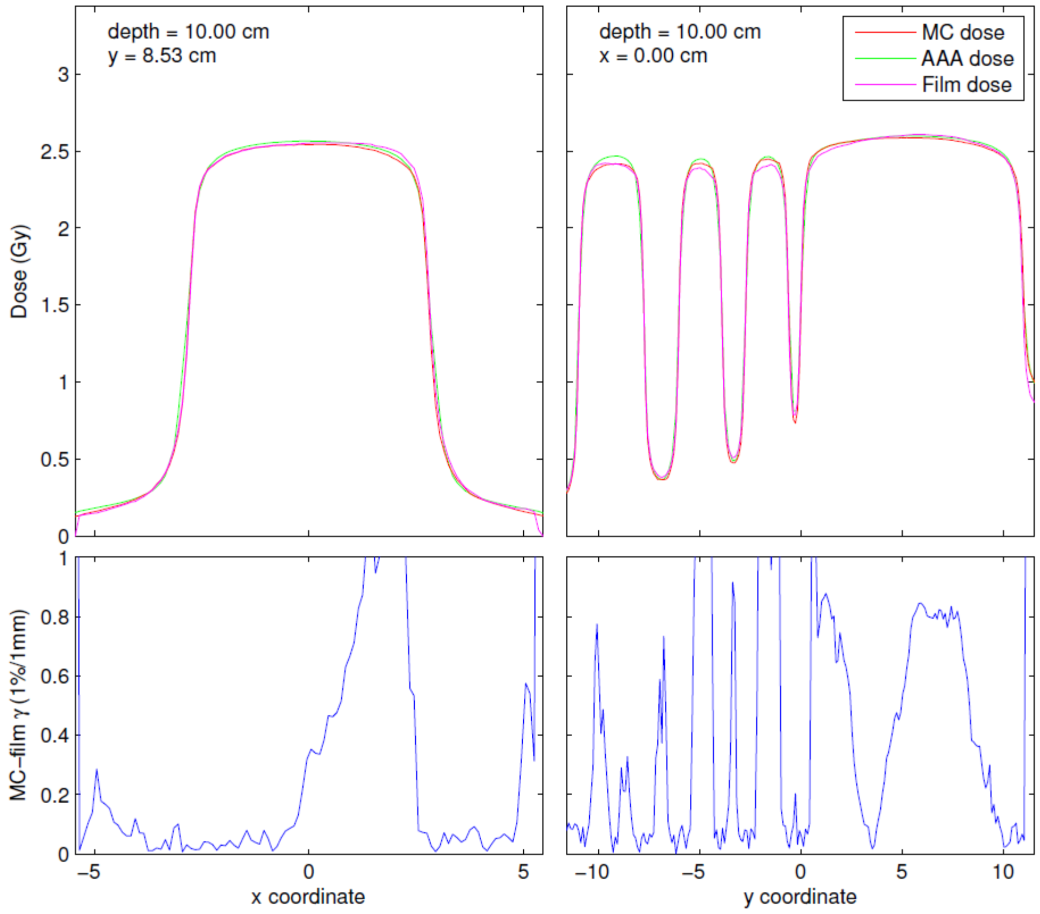


MC, film and AAA profiles for field 1 at 10 cm depth along the red lines delineated in the corresponding planar comparison.

Field 2

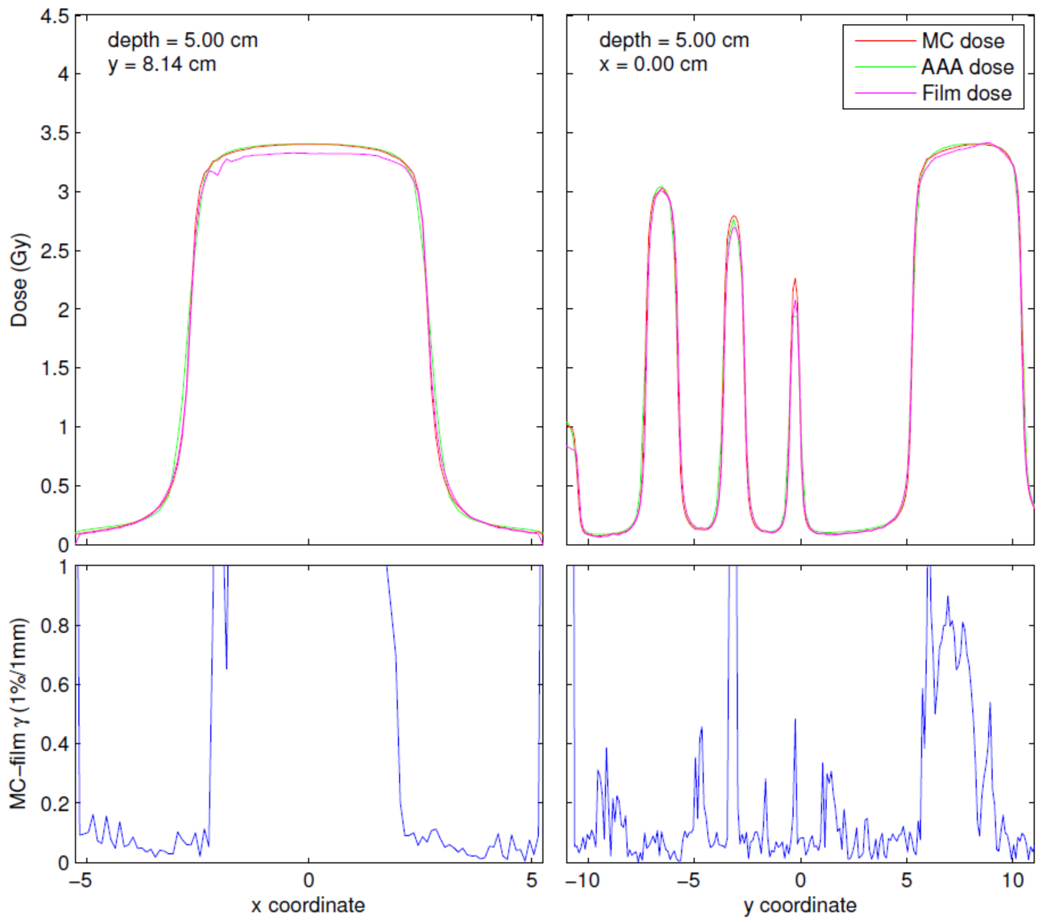


MC, film and AAA profiles for field 2 at 5 cm depth along the red lines delineated in the corresponding planar comparison.

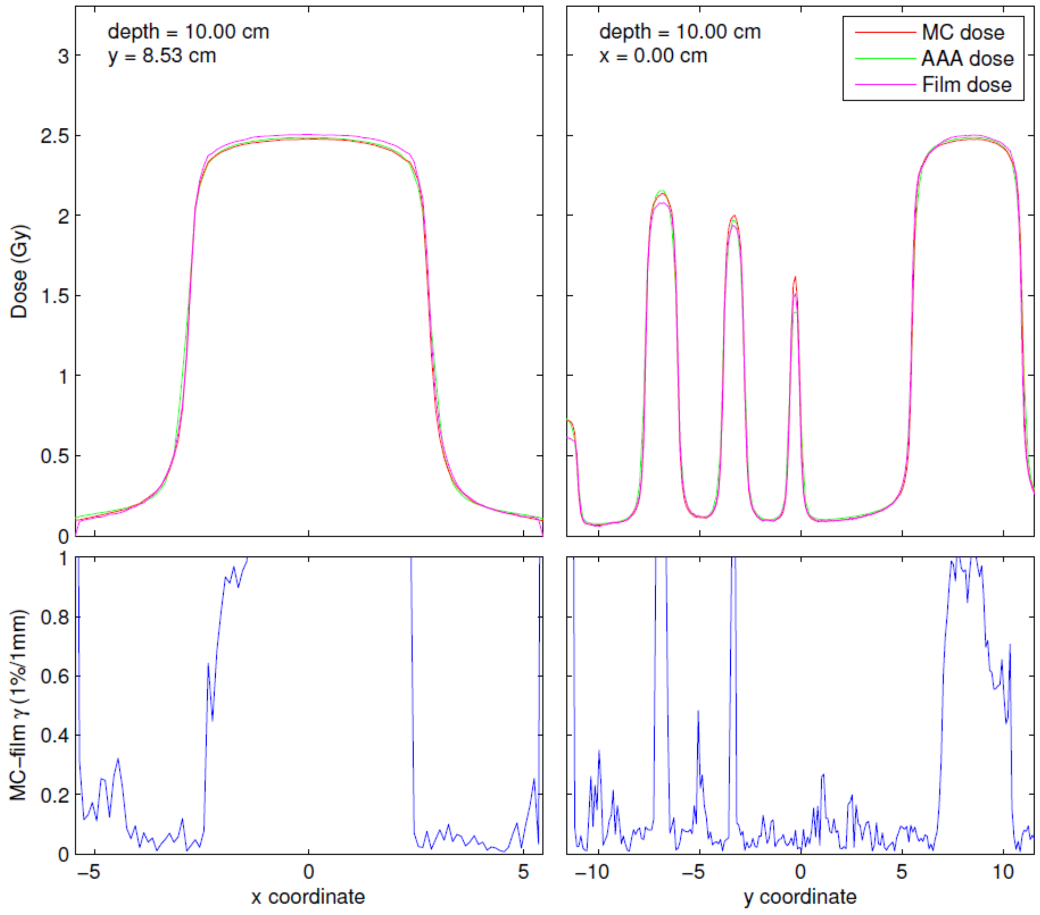


MC, film and AAA profiles for field 2 at 10 cm depth along the red lines delineated in the corresponding planar comparison.

Field 3

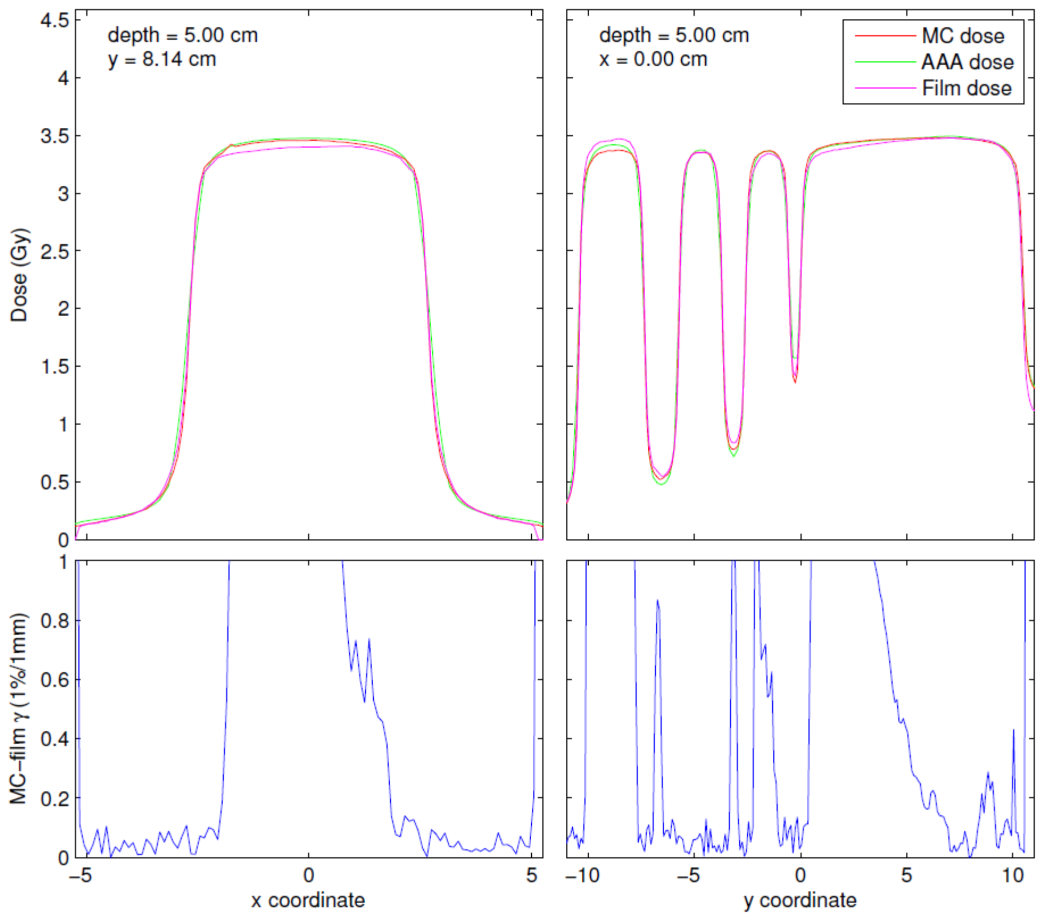


MC, film and AAA profiles for field 3 at 5 cm depth along the red lines delineated in the corresponding planar comparison.



MC, film and AAA profiles for field 3 at 10 cm depth along the red lines delineated in the corresponding planar comparison.

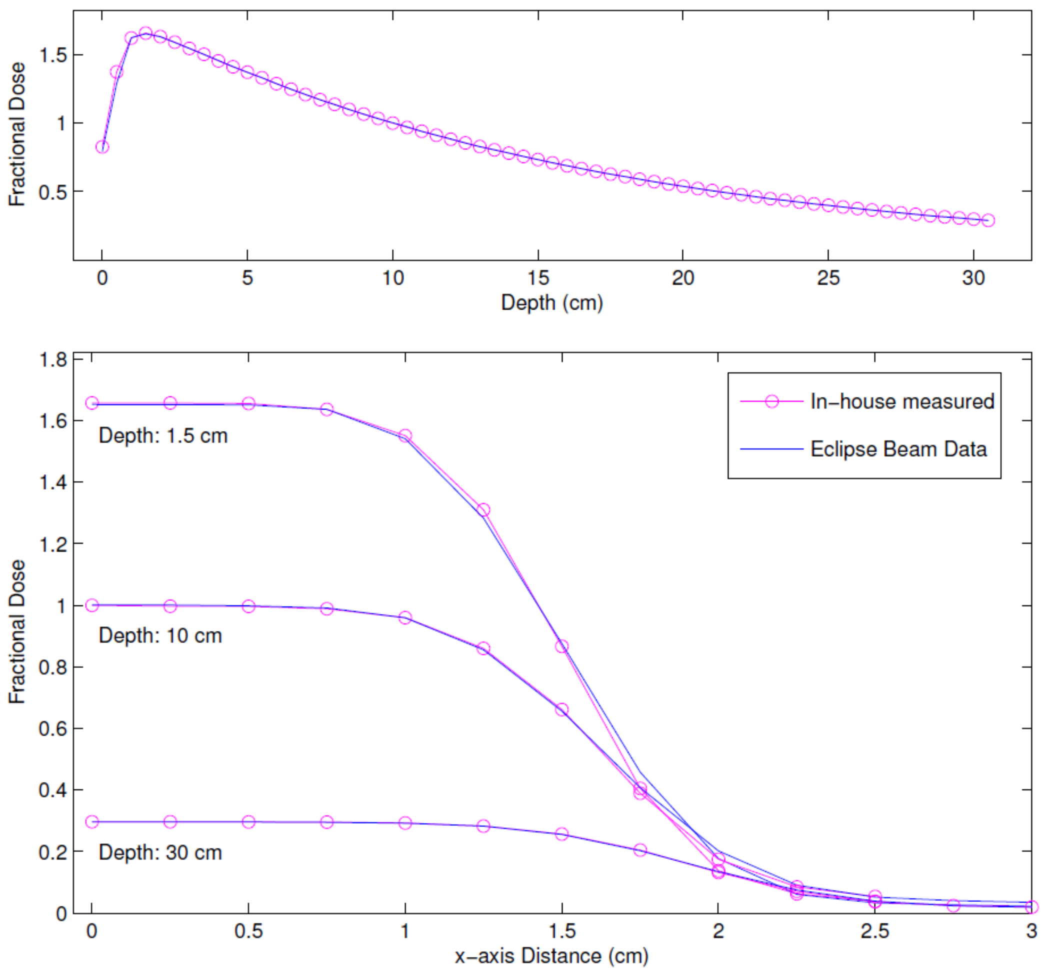
Field 4



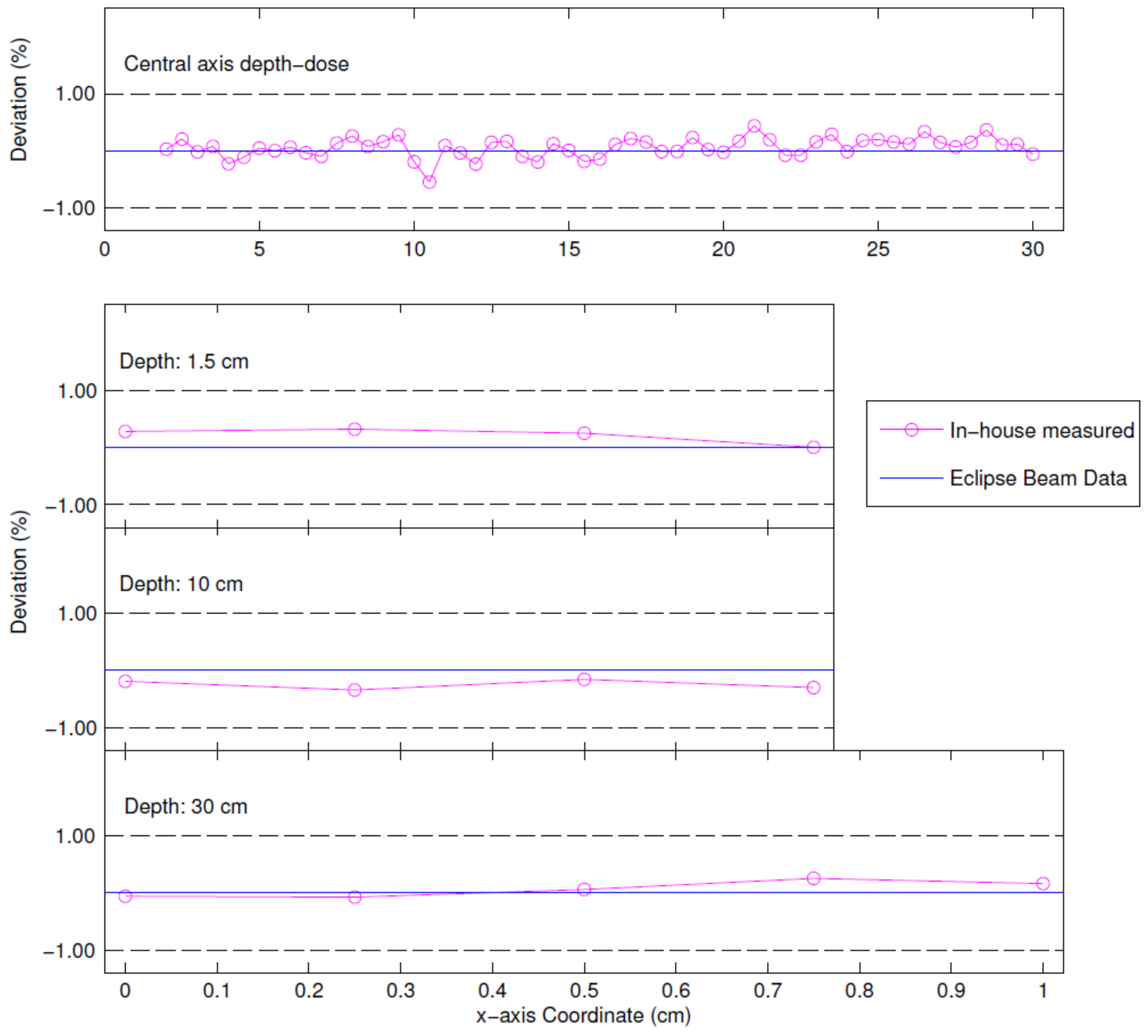
MC, film and AAA profiles for field 4 at 5 cm depth along the red lines delineated in the corresponding planar comparison.

In-house Measured to EBD Comparison

3×3 cm^2 Field Size

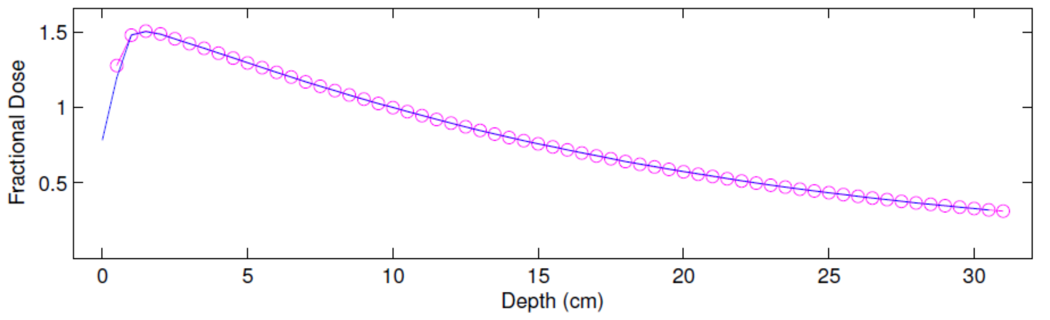


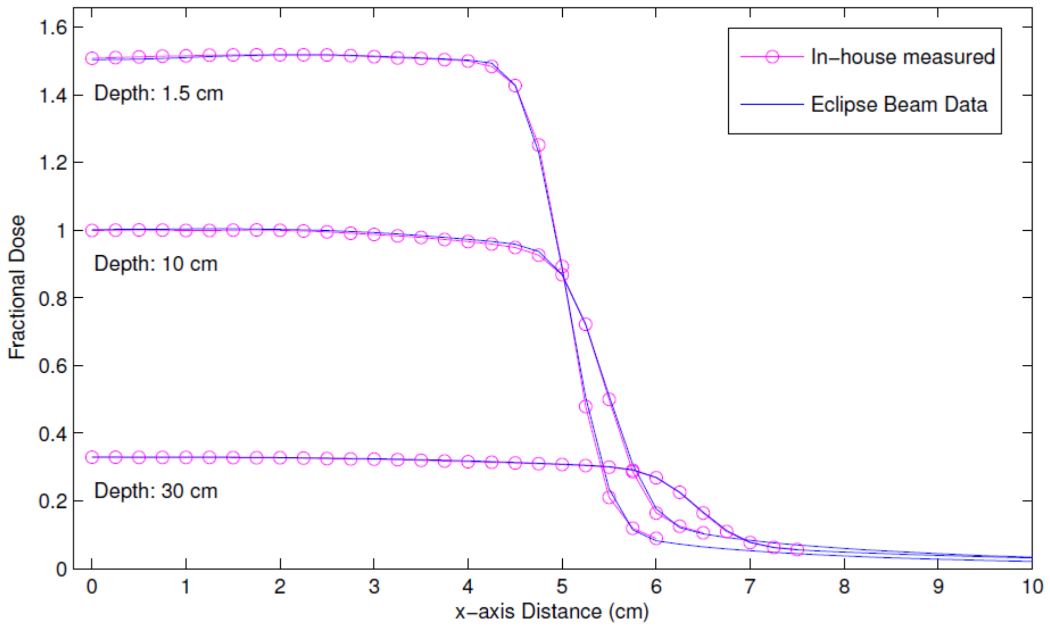
Fractional depth-dose and profile doses for in-house measured and EBD data for the 3×3 cm^2 field size.



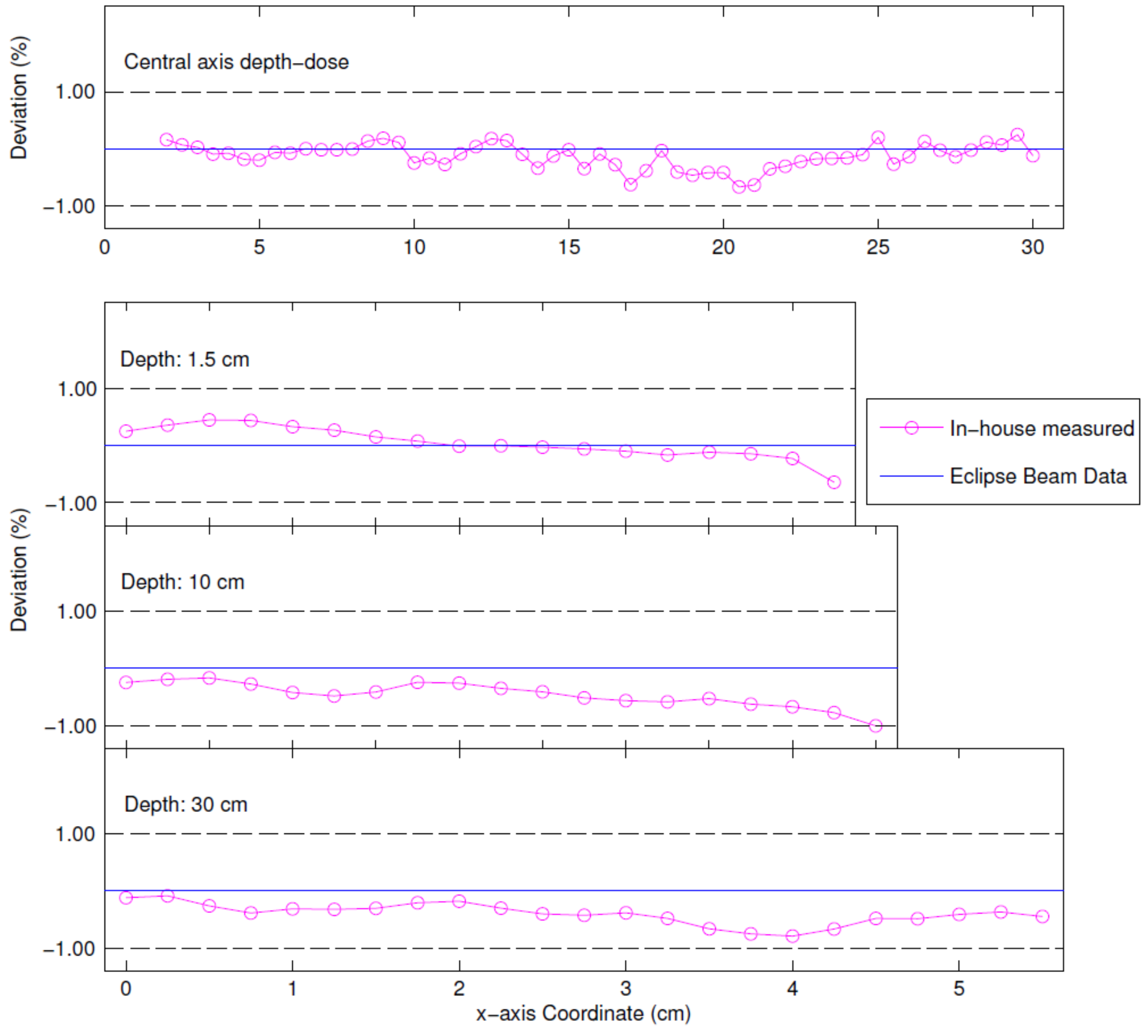
Locally weighted fractional deviation of in-house measured and EBD data for the 3×3 cm^2 field size.

10×10 cm^2 Field Size





Fractional depth-dose and profile doses for in-house measured and EBD data for the 10×10 cm^2 field size.



Locally weighted fractional deviation of in-house measured and EBD data for the 10×10 cm^2 field size.